Natural Language Processing and Text Mining with Graph-Structured Representations

by

Bang Liu

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Engineering

Department of Electrical and Computer Engineering

University of Alberta

Abstract

Natural Language Processing (NLP) and understanding aims to read from unformatted text to accomplish different tasks. As a first step, it is necessary to represent text as a simplified model. Traditionally, Vector Space Model (VSM) is most commonly used, in which text is represented as a bag of words. Recent years, word vectors learned by deep neural networks are also widely used. However, the underlying linguistic and semantic structures of text pieces cannot be expressed and exploited in these representations.

Graph is a natural way to capture the connections between different text pieces, such as entities, sentences, and documents. To overcome the limits in vector space models, we combine deep learning models with graph-structured representations for various tasks in NLP and text mining. Such combinations help to make full use of both the structural information in text and the representation learning ability of deep neural networks. Specifically, we make contributions to the following NLP tasks:

First, we introduce tree-based/graph-based sentence/document decomposition techniques to align sentence/document pairs, and combine them with Siamese neural network and graph convolutional networks (GCN) to perform fine-grained semantic relevance estimation. Based on them, we propose Story Forest system to automatically cluster streaming documents into fine-grained events, while connecting related events in growing trees to tell evolving stories. Story Forest has been deployed into Tencent QQ Browser for hot event discovery.

Second, we propose ConcepT and GIANT systems to construct a user-centered, web-scale ontology, containing a large number of heterogeneous phrases conforming

to user attentions at various granularities, mined from the vast volume of web documents and search click logs. We introduce novel graphical representation and combine it with Relational-GCN to perform heterogeneous phrase mining and relation identification. GIANT system has been deployed into Tencent QQ Browser for news feeds recommendation and searching, serving more than 110 million daily active users. It also offers document tagging service to WeChat.

Third, we propose Answer-Clue-Style-aware Question Generation to automatically generate diverse and high-quality question-answer pairs from unlabeled text corpus at scale by mimicking the way a human asks questions. Our algorithms combine sentence structure parsing with GCN and Seq2Seq-based generative model to make the "one-to-many" question generation close to "one-to-one" mapping problem.

A major part of our work has been deployed into real world applications in Tencent and serves billions of users.

To my parents, Jinzhi Cheng and Shangping Liu.
To my grandparents, Chunhua Hu and Jiafa Cheng.

.

Where there's a will, there is a way.
事在人为

Acknowledgements

This work is not mine alone. I learned a lot and got support from different people for the past six years. Firstly, I am immensely grateful to my advisor, Professor Di Niu, for teaching me how to become a professional researcher. I joined Di's group since September, 2013. During the last 6 years, Di not only created a great research environment for me and all his other students, but also helped me by providing a lot of valuable experiences and suggestions in how to develop a professional career. More importantly, Di is not only a kind and supportive advisor, but also an older friend of mine. He always believes in me even though I am not always that confident about myself. I learned a lot from Di and I am very grateful to him.

I learned a great deal from my talented collaborators and mentors: Professor Linglong Kong and Professor Zongpeng Li. Professor Linglong Kong is my co-supervisor. He is a very nice supervisor as well as friend. He is quite smart and can see the nature of research problems. I learned a lot from him in multiple research projects. Professor Zongpeng Li is one of the coauthors in my first paper. He is an amiable professor with great enthusiasm in research. Thanks for his support in my early research works. I would like to thanks Professor H. Vicky Zhao, who is my co-supervisor when I was pursuing my Master's degree. I am also very grateful to Professor Jian Pei, Davood Rafiei and Cheng Li for being the members in my PhD supervisor committee. Moreover, I would like to thank Professor Denilson Barbosa, James Miller and Scott Dick for being the members in the committee of my PhD candidacy exam.

My friends have made my time over the last six years much more enjoyable. Thanks to my friends, including but not limited to Yan Liu, Yaochen Hu, Rui Zhu, Wuhua Zhang, Wanru Liu, Xu Zhang, Haolan Chen, Dashun Wang, Zhuangzhi Li, Lingju Meng, Qiuyang Xiong, Ting Zhao, Ting Zhang, Fred X. Han, Chenglin Li, Mingjun Zhao, Chi Feng, Lu Qian, Yuanyuan, Ruitong Huang, Jing Cao, Eren, Shuai Zhou, Zhaoyang Shao, Kai Zhou, Yushi Wang, etc. You are my family in Canada. Thank you for everything we have experience together.

I am very grateful to Tencent for their support. I met a lot of friends and brilliant colleagues there. Thanks to my friends Jinghong Lin, Xiao Bao, Yuhao Zhang, Litao

Hong, Weifeng Yang, Shishi Duan, Guangyi Chen, Chun Wu, Chaoyue Wang, Jinwen Luo, Nan Wang, Dong Liu, Chenglin Wu, Mengyi Li, Lin Ma, Xiaohui Han, Haojie Wei, Binfeng Luo, Di Chen, Zutong Li, Jiaosheng Zhao, Shengli Yan, Shunnan Xu, Ruicong Xu and so on. Life is a lot more fun with all of you. Thanks to Weidong Guo, Kunfeng Lai, Yu Xu, Yancheng He, and Bowei Long, for the full support to my research works in Tencent. Thanks to my friend Qun Li and my sister Xiaoqin Zhao for all the accompanying time.

Thanks to my parents, Jinzhi Cheng and Shangping Liu, and my little sister Jia Liu. Your love is what makes me strong. Thanks to all my family members, I love all of you. Lastly, thanks to my grandparents, Chunhua Hu and Jiafa Cheng, who raised me. I will always miss you, grandma.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Building a machine that can understand human language and communicate with people is a long-time dream of researchers. In order to realize this dream, there have been many studies about natural language processing and understanding, computational linguistics, machine learning, or more generally, artificial intelligence.

In the early stage of natural language processing, researchers developed symbolic approaches and expert-designed rule-based systems to capture the meaning of text, but such approaches are unable to deal with unexpected inputs and too restrictive to capture the intricacy of natural language [Winograd, 1972; Ruder, 2019]. As experts cannot write down every possible rules for different NLP tasks, how to learn rules automatically becomes a key problem.

Statistical approaches were proposed in the last 20 years [Manning *et al.*, 1999]. They learn rules automatically by combining statistical models with engineering features of text. However, engineering features is a time-consuming job as features are generally task-specific and require domain expertise. Therefore, the new challenge is how to learn features automatically from raw input text.

As a category of representation learning approaches, deep learning achieves great success in the past seven years [Krizhevsky *et al.*, 2012; Goodfellow *et al.*, 2016; LeCun *et al.*, 2015]. Deep neural network-based models automatically learn a multi-layered hierarchy of features from large amount of data. They greatly reduced the need for feature engineering. However, current deep neural models requires large amount of data and computational resources. Besides, currently, it is still difficult to achieve satisfying performance in NLP tasks that require reasoning based on deep learning.

Reasoning is about making connections between things and form inferences about the world. To understand and reason over text, we need to represent unstructured

text with a simplified model and capture the connections between different text pieces. Most existing approaches utilize Vector Space Models (VSMs) and represent words or text pieces as a sparse one-hot-encoding vectors or dense encoding vectors, where the vector representations are either learned from statistical approaches or trained with deep neural networks. However, natural language text pieces have rich linguistic and semantic structures. Such structures are hard to capture by VSMs. To exploit the underneath structure of text, graph representations of text and graph neural networks that learn over graphs are promising directions to overcome the limits in current natural language processing and understanding models.

Different graph approaches and representations have been proposed to connect text pieces and improve various NLP tasks. For example, word graphs use words as vertices and construct different types of edges, including syntactic analysis [Leskovec *et al.*, 2004], co-occurrences [Zhang *et al.*, 2018*b*; Rousseau and Vazirgiannis, 2013; Nikolentzos *et al.*, 2017] and so on. There are also text graphs that use sentences, paragraphs or documents as vertices. They establish edges by word co-occurrence, location [Mihalcea and Tarau, 2004], text similarities [Putra and Tokunaga, 2017], or hyperlinks between documents [Page *et al.*, 1999]. Besides, hybrid graphs [Rink *et al.*, 2010; Baker and Ellsworth, 2017] consist of different types of vertices and edges.

Aside from text graphs for modeling the relationship between text pieces in a sentence, document or corpus, researchers pay more attention to constructing knowledge graphs and modeling the relations in the world. Large scale graph structured knowledge bases (KBs) store factual information about entities and relationships between them. Until now, a large number of concept and knowledge graphs have been created, including YAGO [Suchanek *et al.*, 2007], DBpedia [Lehmann *et al.*, 2015], Probase [Wu *et al.*, 2012], Freebase [Bollacker *et al.*, 2008], NELL [Carlson *et al.*, 2010], Google Knowledge Graph [Singhal, 2012] and so on. They contain millions of nodes and billions of edges. With these graphs, we can better understand both short queries and long documents, as well as link text with the real world entities and concepts.

In this dissertation, we argue that graph-based approaches can greatly benefit different natural language tasks by explicitly introducing relations between text pieces and reforming unstructured text into structured representations. Our work focus on solving NLP tasks with structured representations and approaches. To this end, we develop novel models for a variety of tasks and demonstrate that our models outperform existing methods, as well as deploy our models into real world applications.

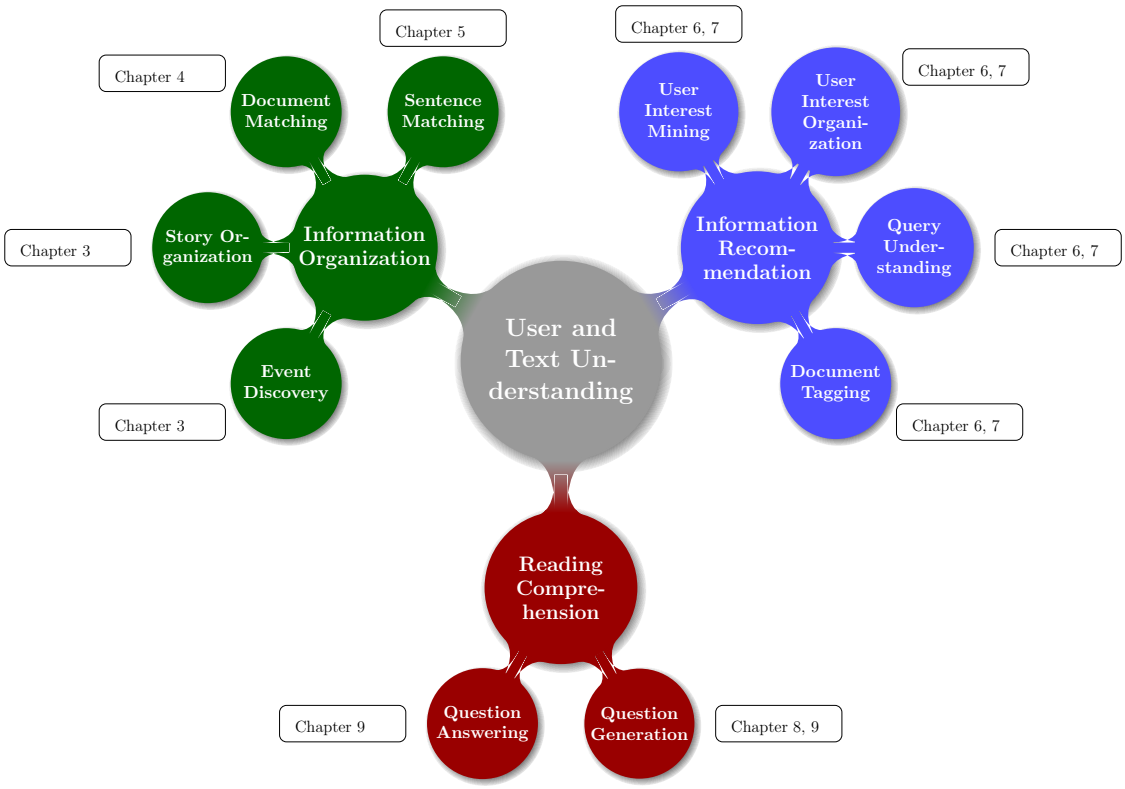


Figure 1.1: The framework of the components in this thesis.

1.2 User and Text Understanding: a Graph Approach

In this thesis, we focus on various natural language processing and text mining tasks which aim to understand users and natural language. Figure 1.1 illustrates the main topics covered in our work, as well as the relationships between them. To improve user and text understanding, our work investigate a variety of research problems on three topics: information organization, information recommendation, and reading comprehension.

Information organization aims to cluster and organize information, such as news articles, to help users retrieve and track useful and non-redundant information easily in the era of information explosion. Our work mainly focus on how to cluster documents into fine-grained events, as well as how to organized related events into structured stories to show their connections. Furthermore, we investigate the problem of document matching and sentence matching, which are core problems in fine-grained document clustering and many other NLP applications.

Information recommendation aims to infer the users' interests based on his/her historical behaviors and recommend information to users that they may be interested

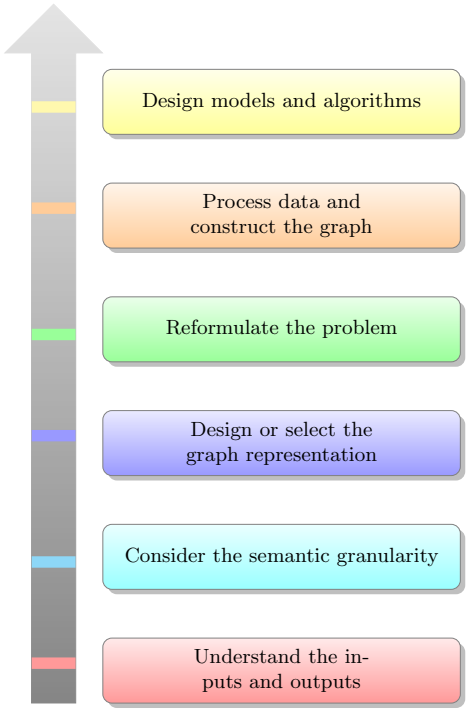


Figure 1.2: The principled methodology used through the different tasks in the thesis.

in. In our work, we focus on mining user interests/attentions from user search click graphs and creating user interests ontology to reveal their relationships. We further perform query understanding and document tagging based on the constructed user interest ontology.

Machine reading comprehension is a core research problem for text understanding. A core challenge in machine reading comprehension is that the creation of high-quality training dataset is quite expensive and time-consuming, as well as requires a lot human efforts. To solve this challenge, we investigate the problem of question generation to automatically generate large amount of high-quality question-answer pairs from unlabeled corpus. We also analyze the performance of question answering models trained with our generated dataset.

Although our thesis contains the discussion about different NLP and text mining tasks, we extensively exploit a unified methodology to analyze and solve different tasks. Figure 1.2 shows the principle methodology used through the different tasks in the thesis. Given a problem, our first step is understand the problem and be clear about the inputs and outputs. Second, we will consider what is the most appropriate semantic granularity, i.e., word, sentence, document, or corpus, to characterize a specific problem. Third, based on our consideration, we will design or select a suitable

graph representation to represent our input data. The key is designing appropriate nodes and edges which are useful in solving the problem. The extraction of node features, edge features, as well as graph attributes are also critical in solving the problem. Fourth, we can reformulate our problem based on the graph representation. For example, we can reformulate the problem of sentence matching as tree matching, or we can reformulate the problem of phrase mining as node selection and ordering. After designed the graph representation and reformulated the problem, we will come up with strategies to process the raw input data and construct the graphs. Finally, we will design models and algorithm for the given problem based on our graph representation.

Use the task of document matching as an example. First, the input is a pair of documents, and the output is a score or a label to indicate the semantic relevance or relationship between the two documents. Second, as a trade of between performance and computational speed, it is most suitable to factorize a document into a set of sentences and compare the document pair in the granularity of sentence. Third, we can group sentences by their sub-topics. Therefore, a node can be a set of correlated sentences discussing the same sub-topic. To show how closely these different sub-topics are related, we can measure the relevance by the text similarities between the sentence sets, and use these similarities as edge features or weights. Fourth, after constructing such a document graph, the problem of text matching turns into a local matching problem on different nodes, and a graph scoring/classification problem based on local matching results. We then implement specific strategies to turn raw article pairs into a document graph, and then design models to estimate the relevance between two articles with our graph representation.

1.3 Contributions

We make extensive contributions to a variety of NLP applications throughout this thesis. Figure 1.3 shows an overview of our works described in this thesis. We can see that our work combines graphical representations of data with machine learning to fully utilize the structural and relational information in different datasets to improve the performance of a variety of tasks. In this tasks, the problems are modeled as tree or graph matching (sentence or document matching), community detection (event/story discovery), node classification and ordering (phrase mining), relationship identification (ontology creation) or node selection (question generation). Our work extensively focused on natural language processing and text mining based on graph-structured representations, and demonstrated the effectiveness of exploiting

Task	Input	Graph	Node	Edge	Operation	Output
sentence matching	two sentences	transformed semantic parsing tree (transformed AMR)	semantic units (word or phrases) in different granularities	compositional relationship	matching trees	relevance score (weighted or binary)
document matching	two documents	concept interaction graph	concepts (keyword or keyword set) with related sentences	similarities	matching graphs	relevance score (weighted or binary)
event/story discovery	documents	keyword graph document graph story tree	keywords documents events	co-occurrence similarities similarities	community detection community detection growing trees	tree structured stories
phrase mining	queries and document titles	query-title interaction graph	words or phrases	sequential relationship, syntactical dependencies, correlations	node classification and node ordering	user attention phrases
ontology creation	user search click graphs	user attention ontology	categories, topic phrases, event phrases, concept phrases, entities	isA relationship, involve relationship, correlate relationship	relationship extraction or identification	user attention ontology
question generation	sentences	dependency tree	words	syntactical dependencies	node selection node classification	questions or question-answer pairs

Figure 1.3: An overview of our works based on graph-structured representations.

the structural information in diverse tasks. Furthermore, our work shows a unified framework to model text data as various graphs and solve the problems in terms of graph computation.

More specifically, for information organization, our contributions include the following:

- We propose the *Story Forest* system for news articles clustering and organization. In this system, we explore a *tree-of-events* representation for organizing news events into logically well-organized trees for better user experience. Besides, we propose the *EventX* algorithm to extract fine-grained events and stories from massive news articles.

- The task of document matching is critical to *Story Forest* system. Therefore, we propose the task of long document matching and apply the divide-and-conquer philosophy to matching a pair of long documents. For long document matching, we propose the so-called *Concept Interaction Graph* (CIG) to represent one or a pair of documents as a weighted graph of concepts, and combines this representation with graph convolutional networks to classify the relationships between two documents.

- Our *Concept Interaction Graph* turns the problem of long document matching into short text matching over vertices. We present a technique named *Hierarchical Sentence Factorization* (or *Sentence Factorization* in short), which is

able to represent a sentence in a hierarchical semantic tree. Based on this representation, we further propose *Ordered Word Mover's Distance* for unsupervised sentence matching, and extend the existing Siamese network architectures to multi-scaled models.

For information recommendation, to help understand user interests and document contents, we make the following contributions:

- We design and implement a concept and event mining system which extract large-scale user-centered concepts and hot topics/events from vast query logs to model user interests and improve query and document understanding. This system constructs and maintains a graph-structured user interest ontology to depict user interests and text topics in different granularities. The nodes in the taxonomy can be tagged to short queries or long documents to improve understanding. The edges in the taxonomy also helps with reasoning and inference over different user interests.

- We implemented and deployed multiple systems (Story Forest, ConcepT, and GIANT) into Tencent QQ Browser. The systems are serving billions of users from Tencent QQ Browser and other applications such as WeChat.

We shall note that our techniques proposed in the thesis are general and can be easily adapted to other languages and other products. They do not rely on any specific features that only Tencent can provide. To adapt our approaches to another language or application, the changes we need to make are mostly data sources, computational and service platforms, off-the-shelf tools for data preprocessing, and hyper-parameters for algorithms.

For machine reading comprehension, we propose efficient systems to generate high-quality training dataset from unlabeled corpus. Specifically:

- We propose a novel Answer-Clue-Style aware question generation system which generates questions based on both given answers and the predicted/sampled clue words. This helps to alleviates the one-to-many mapping problem in text generation. To predict the potential clue words in input, we have designed a novel clue prediction model that combines the syntactic dependency tree of an input with Graph Convolutional Networks. To generate large-scale and high-quality questions, we also propose efficient sampling strategies to sample answer, clue and question types from unlabeled text pieces.

Finally, we open-source our codes and new created datasets in different works for research purpose [1].

1.4 Thesis Outline

Based on the applications of our works, we can divide this thesis into three parts: text clustering and matching for information organization, text mining for information recommendation, and text generation for reading comprehension.

In Chapter 2, we review prior research works that are related to our thesis. Specifically, we will introduce the prior works related to the problems we discussed in information organization, information recommendation, as well as reading comprehension.

In Chapter 3, we focus on the problem of fine-grained event clustering and organization for massive breaking news. In this work, we first propose the design of *Story Forest* system for online news articles organization and visualization, as well as the key *EventX* algorithm for fine-grained event clustering. We then describe our created *Chinese News Events* dataset for evaluating our event extraction algorithms. Based on the dataset, we compare our algorithms with existing approaches and discuss the experimental results.

In Chapter 4 and 5, we present our works on long document matching and short sentence matching, respectively. For document matching, we propose the *Concept Interaction Graph* (CIG) to represent an article as a graph of concepts. We then present a model which combines CIG with graph convolutional networks for semantic matching between a pair of documents. We have created two datasets, each consisting of about 30K pairs of *breaking news* articles covering diverse topics in the open domain, to evaluate our algorithms. For sentence matching, we propose *Hierarchical Sentence Factorization*—a technique that is able to factorize a sentence into a hierarchical representation, with the components at each different scale reordered into a "predicate-argument" form. Based on this technique, we further propose an unsupervised semantic distance metric, as well as multi-scale deep learning models for semantic matching of natural language sentences. We apply our techniques to text-pair similarity estimation and text-pair relationship classification tasks, and show that the proposed hierarchical sentence factorization can be used to significantly improve the performance of existing unsupervised distance-based metrics as well as multiple supervised deep learning models.

In Chapter 6 and 7, we describe our experience of implementing and deploying *ConcepT* and *GIANT* in Tencent QQ Browser. They are concept/event mining sys-

[1] https://github.com/BangLiu/

tems which discover user-centered concepts and hot topics/events at the right granularity conforming to user interests, from vast amount of queries and search logs. We present our techniques for concept mining, document tagging, taxonomy construction, and various applications. Besides, we introduce our experience in deploying ConcepT and GIANT into real world applications, and show their superior ability on user interest modeling as well as query and document understanding.

In Chapter 8, we propose Clue Guided Copy Network for Question Generation (CGC-QG), which is a sequence-to-sequence generative model with copying mechanism, yet employing a variety of novel components and techniques to boost the performance of question generation. We first introduce the problem of one-to-many mapping in text generation. After that, we introduce the concept of clue words for question generation, and propose to predict the clue words in context to alleviate the problem of one-to-many mapping. We design a clue predictor by combing the syntactic structure of sentences with graph convolutional networks. Our model jointly trains the clue prediction as well as question generation with multi-task learning and a number of practical strategies to reduce the complexity. The proposed new modules and strategies significantly improve the performance of question generation. We further propose Answer-Clue-Style-aware Question Generation (ACS-QG) in chapter 9, a novel system aimed at automatically generating diverse and high-quality question-answer pairs from unlabeled text corpus at scale by mimicking the way a human asks questions. With models trained on a relatively smaller amount of data, we can generate 2.8 million quality-assured question-answer pairs from a million sentences in Wikipedia.

We conclude and provide potential future directions in Chapter 10.

Chapter 2

Related Work

Before we describe our approaches for different user and text understanding tasks, let us first set the context by describing prior work done in this space. Our work is related to several lines of research within the NLP and text mining community: text clustering and information retrieval; text matching; ontology creation; phrase mining; question generation and so on.

2.1 Information Organization

There are mainly four research lines that are highly related to our work about information organization: Text Clustering, Story Structure Generation, Text Matching, and Graphical Document Representation.

2.1.1 Text Clustering

The problem of text clustering has been well studied by researchers [Aggarwal and Zhai, 2012]. Distance based clustering algorithms measure the closeness between text pieces with similarity functions such as cosine similarity. Various representations, such as TF-IDF, BM25 term weighting [Büttcher *et al.*, 2006], can be utilized to represent a text object. After transforming text into features, different strategies can be applied for clustering. Partition-based algorithms such as K-means [Jain, 2010] or K-medoids [Park and Jun, 2009] divide the corpus into pre-defined number of clusters. The Spherical K-means algorithm [Buchta *et al.*, 2012] is especially suitable for text clustering due to its low memory and computational requirement. However, such algorithms are sensitive to variations in parameter values and need to specify the number of clusters. The selection of features also plays a key role in the final performance of clustering [Liu *et al.*, 2005]. Hierarchical algorithms [Fung *et al.*, 2003] recursively find nested clusters and create a tree-like structure, but they still need to

assume the number of clusters or a similarity threshold. Density-based algorithms [Ester *et al.*, 1996] do not need to specify the number of clusters in advance, but they do not scale well to high-dimensional sparse data like text [Jain, 2010].

Word and phrase based algorithms find important clusters of words or phrases. [Beil *et al.*, 2002] clusters documents based on frequent pattern mining. [Slonim and Tishby, 2000] proposes a two-phrase clustering procedure that finds word-clusters such that most of the mutual information between words and documents is preserved, and leverages the word-clusters to perform document clustering. Co-clustering algorithms [Dhillon *et al.*, 2003] simultaneously cluster words and documents, as the problem of clustering words and clustering documents are closely related. There are also approaches which utilize the document keywords co-occurrence information to construct a keyword graph, and clustering documents by applying community detection techniques on the keyword graph [Sayyadi and Raschid, 2013].

Non-negative Matrix Factorization is particularly suitable to clustering as a latent space method [Xu *et al.*, 2003]. It factorizes a term-document matrix, where the vectors in the basis system directly correspond to different clusters. It has been shown that matrix factorization is equivalent to spectral clustering [Ding *et al.*, 2005].

Probabilistic model-based algorithms aim to create a probabilistic generative model for text documents. Topic models such as Latent Dirichlet Allocation (LDA) [Blei *et al.*, 2003] and Probabilistic Latent Semantic Indexing (PLSA) [Hofmann, 1999] assume documents are generated by multiple topics. The Gaussian Mixture Model (GMM) [He *et al.*, 2011] assumes that data points are generated by a mixture of Gaussian distributions. However, such model-based algorithms are computationally intensive and do not produce satisfying results when clustering at a finer granularity.

There are also some works concerning the events described in text objects. [Tanev *et al.*, 2008] presents a news event extraction system to extract violent and disaster events from online news. [Ritter *et al.*, 2012] proposes a system to extract an open-domain calendar of significant events from Twitter. In contrast, our EventX algorithm is specially tailored for event extraction among news documents in the open domain. The length of news articles are relatively long compared to Twitters, and the types of events are not restricted to violent and disaster events.

2.1.2 Story Structure Generation

The Topic Detection and Tracking (TDT) research spot news events and group by topics, and track previously spotted news events by attaching related new events into the same cluster [Allan *et al.*, 1998; Allan, 2012; Yang *et al.*, 2009; Sayyadi and

Raschid, 2013]. However, the associations between related events are not defined or interpreted by TDT techniques. To help users capture the developing structure of events, different approaches have been proposed. [Nallapati *et al.*, 2004] proposed the concept of *Event Threading*, and tried a series of strategies based on similarity measure to capture the dependencies among events. [Yang *et al.*, 2009] combines the similarity measure between events, temporal sequence and distance between events, and document distribution along the timeline to score the relationship between events, and models the event evolution structure by a directed acyclic graph (DAG). [Mei and Zhai, 2005] discover and summarize the evolutionary patterns of themes in a text stream by first generating word clusters for each time period and then use the Kullback-Leibler divergence measure to discover coherent themes over time.

The above research works measure and model the relationship between events in a pairwise manner. However, the overall story consistency is not considered. [Wang *et al.*, 2012] generates story summarization from text and image data by constructing a multi-view graph and solving a dominating set problem, but it omits the consistency of each storyline. The *Metro Map* model proposed in [Shahaf *et al.*, 2013] defines metrics such as coherence and diversity for story quality evaluation, and identifies lines of documents by solving an optimization problem to maximize the topic diversity of storylines while guarantee the coherence of each storyline. [Xu *et al.*, 2013] further summarize documents with key images and sentences, and then extract story lines with different definitions of coherence and diversity. These works consider the problem of discovering story development structure as optimizing problems with given news corpora. However, new documents are being generated all the time, and systems that are able to catch related news and update story structures in an online manner are desired.

As studies based on unsupervised clustering techniques [Yan *et al.*, 2011] perform poorly in distinguishing storylines with overlapped events [Hua *et al.*, 2016], more recent works introduce different Bayesian models to generate storyline. However, they often ignore the intrinsic structure of a story [Huang and Huang, 2013] or fail to properly model the hidden relations [Zhou *et al.*, 2015]. [Hua *et al.*, 2016] proposes a hierarchical Bayesian model for storyline generation, and utilize twitter hashtags to "supervise" the generation process. However, the Gibbs sampling inference of the model is time consuming, and such twitter data is not always available for every news stories.

2.1.3 Text Matching.

The task of text matching has been extensively studied for a long time. In recent years, different neural network architectures have been proposed for text pair matching tasks. For representation-focused models, they usually transform text pairs into context representation vectors through a Siamese neural network, followed by a fully connected network or score function which gives the matching result based on the context vectors [Qiu and Huang, 2015; Wan *et al.*, 2016; Liu *et al.*, 2018*a*; Mueller and Thyagarajan, 2016; Severyn and Moschitti, 2015]. For interaction-focused models, they extract the features of all pair-wise interactions between words in text pairs, and aggregate the interaction features by deep networks to give a matching result [Hu *et al.*, 2014; Pang *et al.*, 2016]. However, the intrinsic structural properties of long text documents are not fully utilized by these neural models. Therefore, they cannot achieve good performance for long text pair matching. Here we review related unsupervised and supervised models for text matching.

Traditional unsupervised metrics for document representation, including bag of words (BOW), term frequency inverse document frequency (TF-IDF) [Wu *et al.*, 2008], Okapi BM25 score [Robertson and Walker, 1994]. However, these representations can not capture the semantic distance between individual words. Topic modeling approaches such as Latent Semantic Indexing (LSI) [Deerwester *et al.*, 1990] and Latent Dirichlet Allocation (LDA) [Blei *et al.*, 2003] attempt to circumvent the problem through learning a latent representation of documents. But when applied to semantic-distance based tasks such as text-pair semantic similarity estimation, these algorithms usually cannot achieve good performance.

Learning distributional representation for words, sentences or documents based on deep learning models have been popular recently. *word2vec* [Mikolov *et al.*, 2013] and *Glove* [Pennington *et al.*, 2014] are two high quality word embeddings that have been extensively used in many NLP tasks. Based on word vector representation, the Word Mover's Distance (WMD) [Kusner *et al.*, 2015] algorithm measures the dissimilarity between two sentences (or documents) as the minimum distance that the embedded words of one sentence need to "travel" to reach the embedded words of another sentence. However, when applying these approaches to sentence pair matching tasks, the interactions between sentence pairs are omitted, also the ordered and hierarchical structure of natural languages is not considered.

Different neural network architectures have been proposed for sentence pair matching tasks. Models based on Siamese architectures [Mueller and Thyagarajan, 2016; Severyn and Moschitti, 2015; Neculoiu *et al.*, 2016; Baudiš *et al.*, 2016] usually trans-

form the word embedding sequences of text pairs into context representation vectors through a multi-layer Long Short-Term Memory (LSTM) [Sundermeyer *et al.*, 2012] network or Convolutional Neural Networks (CNN) [Krizhevsky *et al.*, 2012], followed by a fully connected network or score function which gives the similarity score or classification label based on the context representation vectors. However, Siamese models defer the interaction between two sentences until the hidden representation layer, therefore may lose details of sentence pairs for matching tasks [Hu *et al.*, 2014].

Aside from Siamese architectures, [Wang *et al.*, 2017*b*] introduced a matching layer into Siamese network to compare the contextual embedding of one sentence with another. [Hu *et al.*, 2014; Pang *et al.*, 2016] proposed convolutional matching models that consider all pair-wise interactions between words in sentence pairs. [He and Lin, 2016] propose to explicitly model pairwise word interactions with a pairwise word interaction similarity cube and a similarity focus layer to identify important word interactions.

There are also research works which utilize knowledge [Wu *et al.*, 2018], hierarchical property [Jiang *et al.*, 2019] or graph structure [Nikolentzos *et al.*, 2017; Paul *et al.*, 2016] for long text matching. In contrast, our method represents documents by a novel graph representation and combines the representation with GCN.

Finally, pre-training models such as BERT [Devlin *et al.*, 2018] can also be utilized for text matching. However, the model is of high complexity and is hard to satisfy the speed requirement in real-world applications.

2.1.4 Graphical Document Representation

A various of graph representations have been proposed for document modeling. Based on the different types of graph nodes, a majority of existing works can be generalized into four categories: word graph, text graph, concept graph, and hybrid graph.

For word graphs, the graph nodes represent different non-stop words in a document. [Leskovec *et al.*, 2004] extracts subject-predicate-object triples from text based on syntactic analysis, and merge them to form a directed graph. The graph is further normalized by utilizing WordNet [Miller, 1995] to merge triples belonging to the same semantic pattern. [Rousseau and Vazirgiannis, 2013; Rousseau *et al.*, 2015] represent a document as graph-of-word, where nodes represent unique terms and directed edges represent co-occurrences between the terms within a fixed-size sliding window. [Wang *et al.*, 2011] connect terms with syntactic dependencies. [Schenker *et al.*, 2003] connects two words by directed edge if one word is immediately precedes another word in document title, body or link. The edges are categorized by the three different types

of linking.

Text graphs use sentences, paragraphs or documents as vertices, and establish edges by word co-occurrence, location or text similarities. [Balinsky *et al.*, 2011; Mihalcea and Tarau, 2004; Erkan and Radev, 2004] connect sentences if they near to each other, share at least one common keyword, or sentence similarity is above a threshold. [Page *et al.*, 1999] connects web documents by hyperlinks. [Putra and Tokunaga, 2017] constructs directed weighted graphs of sentences for evaluating text coherence. It using sentence similarities as weights and connect sentences with various constraints about sentence similarity or location.

For concept graphs, they link terms in a document to real world entities or concepts based on resources such as DBpedia [Auer *et al.*, 2007], WordNet [Miller, 1995], VerbNet [Schuler, 2005] and so forth. [Schuhmacher and Ponzetto, 2014] identifies the set of concepts contained in a document using DBpedia. Using these concepts as initial seeds, it performs a depth-first search along the DBpedia with a maximum depth of two, and adds all outgoing relational edges and concepts along the paths to form a semantic graph. [Hensman, 2004] identifies the semantic roles in a sentence using WordNet and VerbNet, and combines these semantic roles with a set of syntactic/semantic rules to construct a concept graph.

Hybrid graphs consists of different types of vertices and edges. [Rink *et al.*, 2010] builds a graph representation of sentences that encodes lexical, syntactic, and semantic relations. [Jiang *et al.*, 2010] extract tokens, syntactic structure nodes, part of speech nodes, and semantic nodes from each sentence, and link them by different types of edges that representing different relationships. [Baker and Ellsworth, 2017] combines Frame Semantics and Construction Grammar to construct a Frame Semantic Graph of a sentence.

2.2 Information Recommendation

Our work is mainly related to the following research lines.

2.2.1 Concept Mining

Existing approaches on concept mining are closely related to research works on named entity recognition [Nadeau and Sekine, 2007; Ritter *et al.*, 2011; Lample *et al.*, 2016], term recognition [Frantzi *et al.*, 2000; Park *et al.*, 2002; Zhang *et al.*, 2008], keyphrase extraction [Witten *et al.*, 2005; El-Kishky *et al.*, 2014] or quality phrase mining [Liu *et al.*, 2015; Shang *et al.*, 2018; Liu *et al.*, 2019*c*]. Traditional algorithms utilize

pre-defined part-of-speech (POS) templates and dependency parsing to identify noun phrases as term candidates [Koo *et al.*, 2008; Shang *et al.*, 2018]. Supervised noun phrase chunking techniques [Chen and Chen, 1994; Punyakanok and Roth, 2001] automatically learn rules for identifying noun phrase boundaries. There are also methods that utilize resources such as knowledge graph to further enhance the precision [Witten and Medelyan, 2006; Ren *et al.*, 2017]. Data-driven approaches do not rely on complex linguistic features or rules. Instead, they make use of frequency statistics in the corpus to generate candidate terms and evaluate their quality [Parameswaran *et al.*, 2010; El-Kishky *et al.*, 2014; Liu *et al.*, 2015]. Phrase quality-based approaches exploit statistical features to measure phrase quality, and learn a quality scoring function by using knowledge base entity names as training labels [Liu *et al.*, 2015; Shang *et al.*, 2018]. Neural network-based approaches consider the problem as sequence tagging. They utilize large-scale labeled training data to train complex deep neural models based on CNN or LSTM-CRF [Huang *et al.*, 2015].

2.2.2 Event Extraction

Existing research works on event extraction aim to identify different types of event triggers and their arguments from unstructured text data. They combine supervised or semi-supervised learning with features derived from training data to classify event types, triggers and arguments [Ji and Grishman, 2008; Chen *et al.*, 2017; Liu *et al.*, 2016*b*; Nguyen *et al.*, 2016; Huang and Riloff, 2012]. However, these approaches cannot be applied to new types of events without additional annotation effort. The ACE2005 corpus [Grishman *et al.*, 2005] includes event annotations for 33 types of events. However, such small hand-labeled data is hard to train a model to extract maybe thousands of event types in real-world scenarios. There are also works using neural networks such as RNNs [Nguyen *et al.*, 2016; Sha *et al.*, 2018], CNNs [Chen *et al.*, 2015; Nguyen and Grishman, 2016] or GCNs [Liu *et al.*, 2018*b*] to extract events from text. Open domain event extraction [Valenzuela-Escárcega *et al.*, 2015; Ritter *et al.*, 2012] extracts news-worthy clusters of words, segments and frames from social media data such as Twitter [Atefeh and Khreich, 2015], usually under unsupervised or semi-supervised settings and exploits information redundancy.

2.2.3 Relation Extraction

Relation Extraction (RE) identifies the relationships between different elements such as concepts and entities. A comprehensive introduction can be found in [Pawar *et al.*, 2017]. Most existing techniques for relation extraction can be classified into

the following classes. First, supervised learning techniques, such as features-based [GuoDong *et al.*, 2005] and kernel based [Culotta and Sorensen, 2004] approaches, require entity pairs that labeled with one of the pre-defined relation types as the training dataset. Second, semi-supervised approaches, including bootstrapping [Brin, 1998], active learning [Liu *et al.*, 2016*a*; Settles, 2009] and label propagation [Chen *et al.*, 2006], exploit the unlabeled data to reduce the manual efforts of creating large-scale labeled dataset. Third, unsupervised methods [Yan *et al.*, 2009] utilize techniques such as clustering and named entity recognition to discover relationships between entities. Fourth, Open Information Extraction [Fader *et al.*, 2011] construct comprehensive systems to automatically discover possible relations of interest using text corpus. Last, distant supervision based techniques leverage pre-existing structured or semi-structured data or knowledge to guide the extraction process [Zeng *et al.*, 2015; Smirnova and Cudré-Mauroux, 2018].

2.2.4 Taxonomy and Knowledge Base Construction

Most existing taxonomy or knowledge bases, such as Probase [Wu *et al.*, 2012], DBPedia [Lehmann *et al.*, 2015], YAGO [Suchanek *et al.*, 2007], extract concepts and construct graphs or taxonomies based on Wikipedia or formal documents. To construct domain-specific taxonomies or knowledge bases, they usually select a text corpus as its input, and then extract ontological relationships from the corpus [Poon and Domingos, 2010; Navigli *et al.*, 2011; Zhang *et al.*, 2018*a*; De Sa *et al.*, 2016]. There are also works that construct a taxonomy from keywords [Liu *et al.*, 2012]. [Liu *et al.*, 2019*c*] constructs a three-layered taxonomy from search logs.

2.2.5 Text Conceptualization

Conceptualization seeks to map a word or a phrase to a set of concepts as a mechanism of understanding short text such as search queries. Since short text usually lack of context, conceptualization helps better make sense of text data by extending the text with categorical or topical information, and therefore facilitates many applications. [Li *et al.*, 2007] performs query expansion by utilizing Wikipedia as external corpus to understand query for improving ad-hoc retrieval performance. [Song *et al.*, 2011] groups instances by their conceptual similarity, and develop a Bayesian inference mechanism to conceptualize each group. To make further use of context information, [Wang *et al.*, 2015*b*] utilize a knowledge base that maps instances to their concepts, and build a knowledge base that maps non-instance words, including verbs and adjectives, to concepts.

2.3 Reading Comprehension

Our work about reading comprehension mainly focus on generating question-answer pairs for machine reading comprehension. In this section, we review related works on question generation and the related techniques we utilized.

Rule-Based Question Generation. The rule-based approaches rely on well-designed rules manually created by human to transform a given text to questions [Heilman and Smith, 2010; Heilman, 2011; Chali and Hasan, 2015]. The major steps include preprocessing the given text to choose targets to ask about, and generate questions based on rules or templates [Sun *et al.*, 2018]. However, they require creating rules and templates by experts which is extremely expensive. Also, rules and templates have a lack of diversity and are hard to generalize to different domains.

Answer-Aware Question Generation. Neural question generation models are trained end-to-end and do not rely on hand-crafted rules or templates. The problem is usually formulated as answer-aware question generation, where the position of answer is provided as input. Most of them take advantage of the encoder-decoder framework with attention mechanism [Serban *et al.*, 2016; Du *et al.*, 2017; Liu *et al.*, 2019*b*; Zhou *et al.*, 2017; Song *et al.*, 2018*a*; Hu *et al.*, 2018; Du and Cardie, 2018]. Different approaches incorporate the answer information into generation model by different strategies, such as answer position indicator [Zhou *et al.*, 2017; Liu *et al.*, 2019*b*], separated answer encoding [Kim *et al.*, 2019], embedding the relative distance between the context words and the answer [Sun *et al.*, 2018] and so on. However, with context and answer information as input, the problem of question generation is still a one-to-many mapping problem, as we can ask different questions with the same input.

Auxiliary-Information-Enhanced Question Generation. To improve the quality of generated questions, researchers try to feed the encoder with extra information. [Gao *et al.*, 2018] aims to generate questions on different difficulty levels. It learns a difficulty estimator to get training data, and feeds difficulty as input into the generation model. [Krishna and Iyyer, 2019] learns to generate "general" or "specific" questions about a document, and they utilize templates and train classifier to get question type labels for existing datasets. [Hu *et al.*, 2018] identifies the content shared by a given question and answer pair as an aspect, and learns an aspect-based question generation model. [Gupta *et al.*, 2019] incorporates knowledge base information to ask questions. Compared with these works, our work doesn't require extra labeling or training overhead to get the training dataset. Besides, our settings for question generation dramatically reduce the difficulty of the task, and achieve much better performance.

Multi-task Question Generation. Another strategy is enhancing question generation models with correlated tasks. Joint training of question generation and answering models has improved the performance of individual tasks [Tang *et al.*, 2017; Tang *et al.*, 2018; Wang *et al.*, 2017*a*; Sachan and Xing, 2018]. [Liu *et al.*, 2019*b*] jointly predicts the words in input that is related to the aspect of the targeting output question and will be copied to the question. [Zhou *et al.*, 2019*b*] predicts the question type based on the input answer and context. [Zhou *et al.*, 2019*a*] incorporates language modeling task to help question generation. [Zhang and Bansal, 2019] utilizes question paraphrasing and question answering tasks to regularize the QG model to generate semantically valid questions.

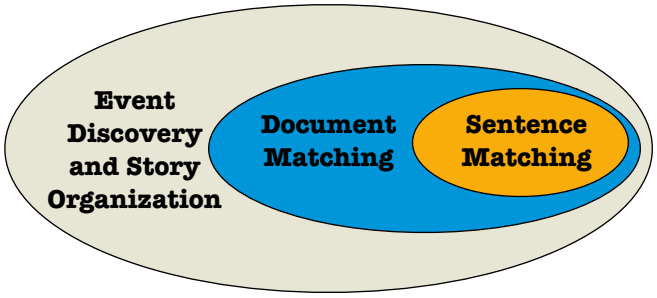
Graph Convolutional Networks. Graph Convolutional Networks generalize Convolutional Neural Networks to graph-structured data, and have been developed and and grown rapidly in scope and popularity in recent years [Kipf and Welling, 2016; Defferrard *et al.*, 2016; Liu *et al.*, 2018*a*; Marcheggiani and Titov, 2017; Battaglia *et al.*, 2018]. Here we focus on the applications of GCNs on natural language. [Marcheggiani and Titov, 2017] applies GCNs over syntactic dependency trees as sentence encoders, and produces latent feature representations of words in a sentence for semantic role labeling. [Liu *et al.*, 2018*a*] matches long document pairs using graph structures, and classify the relationships of two documents by GCN. [Zhang *et al.*, 2018*c*] proposes an extension of graph convolutional networks that is tailored for relation extraction. It pools information over dependency trees efficiently in parallel.

Sequence-to-Sequence Models. Sequence-to-sequence model has been widely used in natural language generation. [Sutskever *et al.*, 2014] proposes a sequence-to-sequence model for the task of machine translation. [Bahdanau *et al.*, 2014] further improves the model performance of machine translation by introducing attention mechanism to the sequence-to-sequence model. To deal with the out-of-vocabulary issue, the copy mechanism is incorporated into sequence-to-sequence models to copy words from source text [Cao *et al.*, 2017; Gu *et al.*, 2016]. In our work, we apply copy mechanism to learn to copy potential clue chunks from the input text, instead of restricting it to out-of-vocabulary words.

Pretrained Language Models. Pre-trained large-scale language models, such as BERT [Devlin *et al.*, 2018] and GPT2 [Radford *et al.*, 2019], have dramatically improved the performance over a series of NLP task [Sun *et al.*, 2019; Yang *et al.*, 2019; Lample and Conneau, 2019]. These pre-trained language modeling have been shown to capture many facets of language relevant for downstream tasks [Clark *et al.*, 2019]. As sequence-to-sequence models often outputs sentences that contain repeated words, we also fine-tuned a GPT2-based question generation model to avoid this problem.

Part I

Text Clustering and Matching: Growing Story Trees to Solve Information Explosion



In the era of information explosion, it is not easy for users to retrieve and track high-quality, well-organized, and non-redundant information that they are interested in from huge amount of resources. In chapter 3, we will introduce our Story Forest system for intelligent news articles organization. Our system contains a set of online schemes that automatically clusters streaming documents into events, while connecting related events in growing trees to tell evolving stories. A core novelty of our Story Forest system is EventX, a semi-supervised scheme to extract events from massive Internet news corpora.

EventX relies on a two-layered, graph-based clustering procedure to group documents into fine-grained events. A key step in the second layer clustering procedure is classifying whether two documents are talking about the same event. This is a problem of document matching. In chapter 4, we propose the Concept Interaction Graph to represent an article as a graph of concepts. We then match a pair of articles by comparing the sentences that enclose the same concept vertex through a series of encoding techniques, and aggregate the matching signals over each vertex and get a final matching result.

Concept interaction graph turns the problem of long document matching into short sentence matching over different vertices. In chapter 5, we propose Hierarchical Sentence Factorization—a technique to factorize a sentence into a hierarchical representation, with the components at each different scale reordered into a "predicate-argument" form. We then apply our techniques to text-pair similarity estimation and text-pair relationship classification tasks.

Chapter 3

Story Forest: Extracting Events and Telling Stories from Breaking News

Extracting events accurately from vast news corpora and organize events logically is critical for news apps and search engines, which aim to organize news information collected from the Internet and present it to users in the most sensible forms. Intuitively speaking, an event is a group of news documents that report the same news incident possibly in different ways. In this chapter, we describe our experience of implementing a news content organization system at Tencent to discover events from vast streams of breaking news and to evolve news story structures in an online fashion. Our real-world system faces unique challenges in contrast to previous studies on topic detection and tracking (TDT) and event timeline or graph generation, in that we 1) need to accurately and quickly extract distinguishable events from massive streams of long text documents, and 2) must develop the structures of event stories in an online manner, in order to guarantee a consistent user viewing experience. In solving these challenges, we propose *Story Forest*, a set of online schemes that automatically clusters streaming documents into events, while connecting related events in growing trees to tell evolving stories. A core novelty of our *Story Forest* system is *EventX*, a semi-supervised scheme to extract events from massive Internet news corpora. *EventX* relies on a two-layered, graph-based clustering procedure to group documents into fine-grained events. We conducted extensive evaluation based on 1) 60 GB of real-world Chinese news data, 2) a large Chinese Internet news dataset that contains $11,748$ news articles with ground truth event labels, and 3) the 20 News Groups English dataset, through detailed pilot user experience studies. The results demonstrate the superior capabilities of *Story Forest* to accurately identify events and organize news text into a logical structure that is appealing to human readers.

3.1 Introduction

With information explosion in a fast-paced modern society, tremendous volumes of news articles are constantly being generated on the Internet by different media providers, e.g., Yahoo! News, Tencent News, CNN, BBC, etc. In the meantime, it becomes increasingly difficult for average readers to digest the huge volumes of daily news articles, which may cover diverse topics and contain redundant or overlapping information. Many news app users have the common experience that they are overwhelmed by highly redundant information about a number of ongoing hot events, while still being unable to get information about the events they are truly interested in. Furthermore, search engines perform document retrieval from large corpora based on user-entered queries. However, they do not provide a natural way for users to view trending topics or breaking news.

An emerging alternative way to visualize news corpora without pre-specified queries is to organize and present news articles through event timelines [Yan *et al.*, 2011; Wang *et al.*, 2016], event threads [Nallapati *et al.*, 2004], event evolution graphs [Yang *et al.*, 2009], or information maps [Shahaf *et al.*, 2012; Shahaf *et al.*, 2013; Xu *et al.*, 2013]. All of these approaches require the extraction of conceptually clean events from a large number of messy news documents, which involves automated event extraction and visualization as a crucial step toward intelligent news systems. However, few existing news information organization techniques successfully achieve this goal due to several reasons:

First of all, prior research on Topic Detection and Tracking (TDT) [Allan, 2012] as well as text clustering [Aggarwal and Zhai, 2012; Jain, 2010] mainly focused on grouping related documents into topics—it is much harder to cluster articles by events, where articles depicting the same event should be grouped together, since the number of events that occur daily in the real world is unpredictable. As a result, we cannot use some of the popular clustering algorithms, e.g., K-means, that require predefining the number of clusters, to extract events. In addition, the sizes of event clusters are highly skewed, because hot events may be extensively discussed by tens or even hundreds of news articles on the Internet. In contrast, regular events will be reported by only a few or even one article. These single-document events, however, constitute the majority of daily news collections, and should also be accurately discovered to appeal to the diverse interests of readers.

Second, many recently proposed event graphs or information maps try to link events in an evolution graph [Yang *et al.*, 2009] or permitting intertwining branches in the information map [Shahaf *et al.*, 2013]. However, we would like to argue that such

overly complex graph structures do not make it easy for users to quickly visualize and understand news data. In fact, most *breaking news* follows a much simpler storyline. Using complex graphs to represent breaking news may complicate and even blur the story structure.

Third, most existing event time-line or event graph generation schemes are based on *offline* optimization over the entire news corpora. However, for an automated event extraction system that aids the visualization of breaking news, it is desirable to "grow" the stories in an online fashion as news articles are published, without disrupting or restructuring the previously generated storylines. On one hand, given the vast amount of daily news data, incremental and online computation will incur less computation overhead by avoiding repeated processing of older documents. On the other hand, an online scheme can deliver a consistent story development structure to users, so that users can quickly follow newly trending events.

In this chapter, we propose the *Story Forest*, a novel news organization system that addresses the aforementioned challenges. To extract conceptually clean events, each of which is essentially a cluster of news documents describing the same physical breaking news event, Story Forest incorporates a novel, semi-supervised, two-layered document clustering procedure that leverages a wide range of feature engineering and machine learning techniques, including keyword extraction, community detection, and graph-based clustering. We call this clustering procedure *EventX*. To the best of our knowledge, it is the first document clustering scheme specially tailored for event extraction among breaking news documents in the open domain.

We start with the observation that documents focusing on the same topic usually contain overlapping keywords. Therefore, in the first layer of the clustering procedure in *EventX*, we utilize a classifier trained on over $10,000$ news articles to distinguish keywords from non-keywords for each document. We then apply an existing community detection algorithm onto a keyword co-occurrence graph constructed from news corpora and extract subgraphs [Sayyadi and Raschid, 2013] of keywords to represent topics. Each document is assigned a topic by finding out its most similar keyword subgraph. However, a keyword community or a topic is still coarse-grained and may cover many events. In the second layer of *EventX*, documents within each topic are further clustered into fine-grained events. We construct a document relationship graph within each topic, where the relationship between each pair of documents, i.e., whether they describe the same event, is predicted by a supervised document pair relationship classifier trained on carefully handcrafted features. Finally, we apply the graph-based community detection algorithm again to decompose the document relationship graph of each topic into conceptually separate events.

To enhance event visualization, our Story Forest system further groups the discovered events into stories, where each story is represented by a *tree* of interconnected events. A link between two events indicates the temporal evolution or a causal relationship between the two events. In contrast with existing story generation systems such as StoryGraph [Yang *et al.*, 2009] and MetroMap [Shahaf *et al.*, 2012], we propose an online algorithm to evolve story trees incrementally as breaking news articles arrive. Consequently, each story (called a story tree) is presented in one of several easy-to-view structures, i.e., either a linear timeline, a flat structure, or a tree with possibly multiple branches, which we believe are succinct and sufficient to represent story structures of most breaking news.

Currently, access to the related public data for event extraction and organization is extremely limited. Therefore, to facilitate evaluation and further research on the problem of event clustering and story formation for breaking news, we have created multiple datasets, with the effort of dedicated editors. First, we have created the *Chinese News Corpus* dataset which contains 60 GB of Chinese news documents collected from all major Internet news providers in China (including Tencent, Sina, WeChat, Sohu, etc.) in a 3-month period from October 1, 2016 to December 31, 2016, covering very much diversified topics in the open domain. Second, we further created the *Chinese News Events* dataset, where each article is manually labeled with the true event label and story label by editors and product managers at Tencent. It is also, to the best of our knowledge, the first Chinese dataset for event extraction evaluation. The new datasets have been made publicly available for research purposes.[1]

We evaluated the performance of Story Forest based on the Chinese News Corpus dataset, and compared our EventX news document clustering algorithm with other approaches on the Chinese News Events dataset. We also conducted a detailed and extensive pilot user experience study for (long) news document clustering and news story generation to evaluate how our system as well as several baseline schemes appeal to the habit of human readers. According to the pilot user experience study, our system outperforms multiple state-of-the-art news clustering and story generation systems, such as KeyGraph [Sayyadi and Raschid, 2013] and StoryGraph [Yang *et al.*, 2009], in terms of logical validity of the generated story structures, as well as the conceptual cleanness of each identified event/story. Experiments show that the average time for our Java-based system to finish event clustering and story structure generation based on the daily news data is less than 30 seconds on a MacBook Pro

[1]Our Chinese News Events dataset is currently available at:https://pan.baidu.com/s/12vWHHTD8gQLPvVftm6LQdg. For the Chinese News Corpus dataset, we are currently under the process of publishing it to the public for research purposes.

with a 2 GHz Intel Core i7 processor, and 8 GB memory. Therefore, our system proves to be highly efficient and practical.

To summarize, we make the following contributions in this chapter:

- We formally define the problem of event extraction for breaking news articles in the open domain, where the granularity of an event must conform to the physical events described by the articles and can be implicitly guided by the labeled dataset in our semi-supervised algorithms. We will describe it in more details in Sec. 3.2.

- We propose the *EventX* algorithm, which is a two-layered, graph-based document clustering algorithm that can perform fast event extraction from a large volume of news documents in a semi-supervised manner. Note that the main novelty of *EventX* includes a *layered* clustering scheme to separate the problem of topic discovery from that of finer-grained event extraction. Such a two-layered graph-based clustering scheme significantly improves the overall time efficiency and scalability of the algorithm, making it applicable for industry practice.

- We explore a *tree-of-events* representation for visualizing news documents. We also introduce an online algorithm to dynamically incorporate new events into the existing trees. Combining this approach with the *EventX* algorithm, we create the Story Forest system, for intelligent and efficient news story structure formation.

- We have collected and labeled a large amount of data for the study and evaluation of event extraction and story structure organization, since to our best knowledge, there is no publicly available dataset specifically dedicated to news event clustering or extraction and story formation.

Our algorithm has been successfully integrated into the hot event discovery feature of Tencent QQ browser, which is one of the most popular mobile browsers that serves over 100 millions of daily active users.

The remainder of this chapter is organized as follows. Sec. 3.2 formally describes the problem of event extraction and organization from massive news data. In Sec. 3.3, we propose the main design of *Story Forest* system and *EventX* algorithm. In Sec. 3.4, we describe the *Chinese News Events* dataset collected and created specifically for evaluating event extraction algorithms. We then compare and discuss the experimental results of *EventX* and *Story Forest* among other baselines. This chapter is concluded in Sec. 3.5.

3.2 Problem Definition and Notations

In this section, we will first describe key concepts and notations used in this chapter, and formally define our problem. Then, we conduct a case study to clearly illustrate the idea of story trees.

3.2.1 Problem Definition

We first present the definitions of some key concepts in a bottom-up hierarchy, *event* $\rightarrow$ *story tree* $\rightarrow$ *topic*, to used in this chapter.

Definition 1. *Event: an event $\mathcal{E}$ is a set of news documents reporting a same piece of real-world breaking news.*

Definition 2. *Story tree: a story $\mathcal{S}$ is a tree of related events that report a series of evolving real-world breaking news. A story usually revolves around a group of specific persons and happen at certain places during specific times. Each node in the tree is a story, and each unidirectional edge in the tree indicates the time sequence of events. Each branch in the tree represents a subset of events that are talking about the same sub-story. The granularity of a story is subjective and can be implicitly determined by a training dataset.*

Definition 3. *Topic: a topic consists of a set of stories that are highly correlated or similar to each other.*

In this work, we assume that two news articles are describing the same event as long as the incident they cover/report has the same time of occurrence and involves the same participating entities. This assumption is highly justifiable for breaking news articles on the Internet, which unlike fictions, are usually short, succinct and focusing on reporting the incident of a group of specific persons, organizations or other types of entities, taking actions at one or several locations. We do not consider the stance, perspective or the publishing date of an article. Additionally, we do not model relationships between events, other than the weightless temporal evolution required for building story trees. This allows each event to be conceptually cleaner and more specific.

In contrast, a topic is usually determined in a subjective manner, and is usually more vague and broader compared to an event. For example, both *American presidential election* and *2016 U.S. presidential election* can be considered topics, with the second topic being a subset of the first topic. We also make the assumption that each event has a single most appropriate corresponding topic. This ensures that an

event cannot appear under multiple topic clusters, which significantly simplifies the clustering procedure. This assumption is also realistic for most *breaking news* articles, where an article is usually written for timeliness to report only one real-world incident.

Each topic may contain multiple story trees, and each story tree consists of multiple logically connected events. In our work, events (instead of news documents) are the smallest atomic units. Each event is also assumed to belong to a single story and contributes partial information to the overall story. For instance, considering the topic *American presidential election*, *2016 U.S. presidential election* is a story within this topic, and *Trump and Hilary's first television debate* is an event within this story.

The boundaries between events, stories and topics do not have to be explicitly defined. In real-world applications, the labeled training dataset often implicitly captures and reflects their differences. In this way, even documents that are not related to evolving physical events can be clustered in according to a specific granularity, as long as the labeled training dataset contains such kind of samples to help defining the boundary. Therefore, our EventX Event Extraction algorithm defines a general framework to cluster documents into events that implicitly defined by any training dataset.

Note that prior studies on text clustering usually focus on clustering at the granularity of topics [Allan, 2012; Aggarwal and Zhai, 2012], which are clusters of related articles. In contrast, the event extraction problem is much more challenging, because on top of clustering documents that belong to the same topic, we also need to utilize key information within each document to ensure that all documents within an event cluster are reporting the same physical event.

3.2.2 Notations

We now introduce some notations and describe our problem formally. Given a news document stream $D = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_t, \ldots\}$, where $\mathcal{D}_t$ is the set of news documents collected on time period t, our objective is to: a) cluster all news documents D into a set of events $E = \{\mathcal{E}_1, \ldots, \mathcal{E}_{|E|}\}$, and b) connect the extracted events to form a set of stories $S = \{\mathcal{S}_1, ..., \mathcal{S}_{|S|}\}$. Each story $\mathcal{S} = (E, L)$ contains a set of events E and a set of links L, where $L_{i,j} :=< \mathcal{E}_i, \mathcal{E}_j >$ denotes a directed link from event $\mathcal{E}_i$ to $\mathcal{E}_j$, which indicates a temporal evolution or logical connection relationship.

Furthermore, we require the events and story trees to be extracted in an online or incremental manner. For online, it means that we extract events from each $\mathcal{D}_t$ in an online manner as soon as the news corpus $\mathcal{D}_t$ arrives in time period t. For incremental,

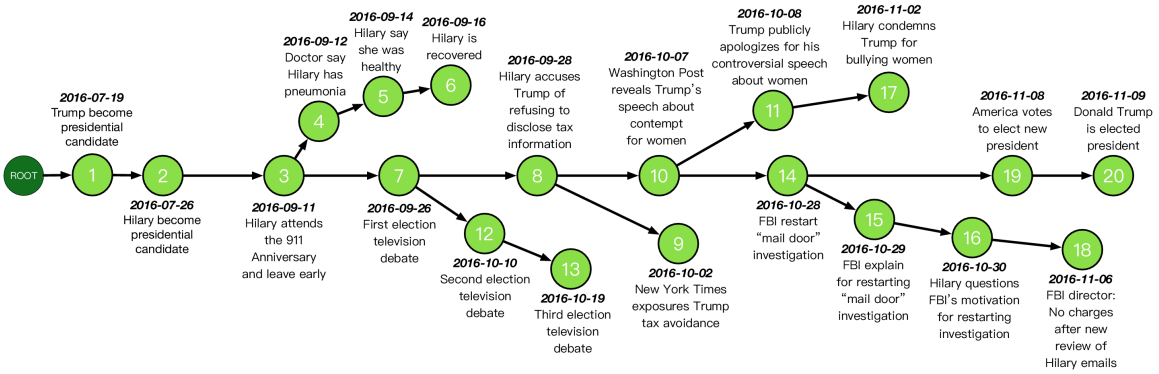


Figure 3.1: The story tree of "2016 U.S. presidential election."

we will *merge* or *insert* the discovered events into the existing story trees that were found at time $t-1$ without changing the existing tree structures. This is a unique strength of our scheme as compared to prior work, since we do not need to repeatedly process older documents and can deliver the complete set of evolving yet logically consistent story trees to users on-demand.

3.2.3 Case Study

To give readers more intuition on what the stories or events look like, here we use an illustrative example to help further clarify the concept of stories vs. events. Fig. 3.1 showcases the story tree of "2016 U.S. presidential election". The story contains 20 nodes, where each node indicates an event in 2016 U.S. election, and each link indicates a temporal evolution or a logical connection between two events. For example, event 19 says America votes to elect new president, and event 20 says Donald Trump is elected president. The index number on each node represents the event sequence over the timeline. There are 6 paths within this story tree, where the path $1 \rightarrow 20$ capture the whole presidential election process, branch $3 \rightarrow 6$ are about Hilary's health conditions, branch $7 \rightarrow 13$ are about television debates, $14 \rightarrow 18$ are related to the "mail door" investigation, etc. As we can see, by modeling the evolutionary and logical structure of a story into a story tree, users can easily grasp the logic of news stories and learn the main information quickly. Let us consider the following 4 events under the topic *2016 U.S. presidential election*: 1) *First election television debate*; 2) *Second election television debate*; 3) *FBI restarts "mail door" investigation*; 4) *America votes to elect the new president.* Intuitively, these 4 events should having no overlapping information between them. A news article about Trump and Hillary's *First election television debate* is conceptually separate from another article that is reporting Trump and Hillary's *Second election television debate.* For news articles,

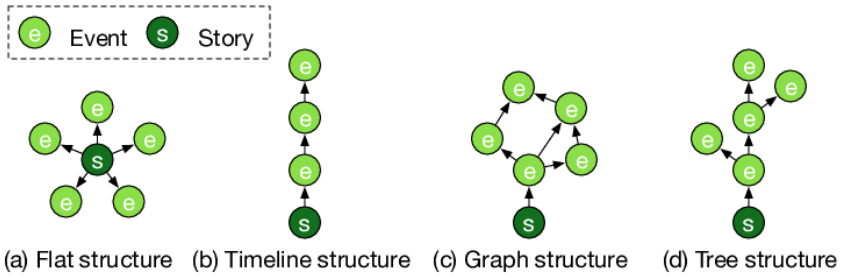


Figure 3.2: Different structures to characterize a story.

different events under the same topic should be clearly distinguishable, because they usually follow the progressing timeline of real-world affairs.

Let us represent each story by an empty root node s from which the story is originated, and denote each event by an event node e. The events in a story can be organized in one of the following four structures shown in Fig. 3.2: a) a flat structure that does not include dependencies between events; b) a timeline structure that organizes events by their timestamps; c) a graph structure that checks the connection between all pairs of events and maintains a subset of most strong connections; d) a tree structure to reflect the structures of evolving events within a story. Compared with a tree structure, sorting events by timestamps omits the logical connection between events, while using directed acyclic graphs to model event dependencies without considering the evolving consistency of the whole story can leads to unnecessary connections between events. Through extensive user experience studies in Sec. 3.4, we show that tree structures are the most effective way to represent breaking news stories.

3.3 The Story Forest System

In this section, we start with an overview of the proposed *Story Forest* system. Then, we separately introduce the detailed procedures of event extraction from news documents, and how we model stories' evolutionary structure by story trees.

An overview of our *Story Forest* system is shown in Fig. 3.3, which mainly consists of four components: preprocessing, keyword graph construction, clustering documents to events, and growing story trees with events. The overall process is divided into 8 steps. First, the input news document stream will be processed by a variety of NLP and machine learning tools, including document filtering and word segmentation. Then we perform keyword extraction, construct/update keyword co-occurrence graph, and split the graph into sub-graphs. After that, we utilize our proposed *EventX*

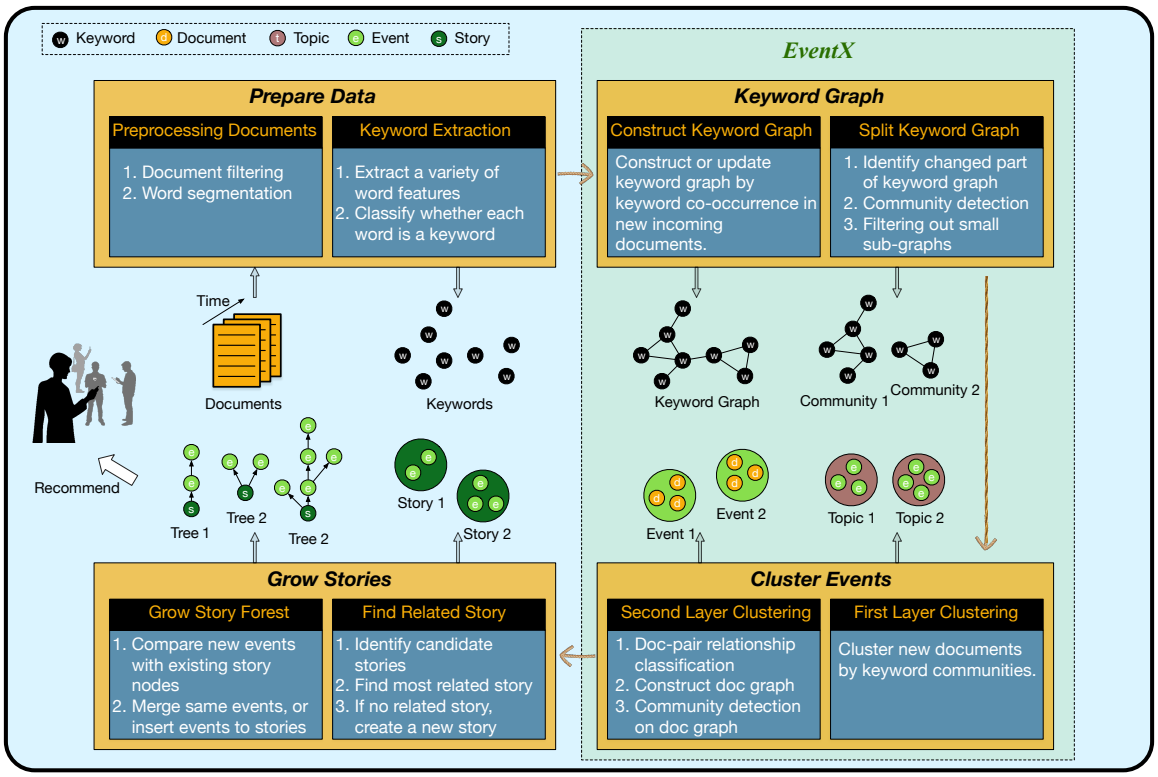


Figure 3.3: An overview of the system architecture of *Story Forest*.

algorithm to cluster documents into fine-grained events. Finally, we update the story trees (formed previously) by either inserting each discovered event into an existing story tree at the right place, or creating a new story tree if the event does not belong to any existing story. Note that each topic may contain multiple story trees and each story tree consists of logically connected events. We will explain the design choices of each component in detail in the following subsections.

3.3.1 Preprocessing

When new documents arrive, the first task *Story Forest* performs is document preprocessing, which includes the following sequential steps:

Document filtering: documents with content length smaller than a threshold (20 characters) will be discarded.

Word segmentation: we segment the title and body of each document using Stanford Chinese Word Segmenter *Version 3.6.0* [Chang *et al.*, 2008], which has proved to yield excellent performance on Chinese word segmentation tasks. Note that for data in a different language, the corresponding word segmentation tool in that language can be used.

Figure 3.4: The structure of keyword classifier.

Type	Features
Word feature	Named entity or not, location name or not, contains angle brackets or not.
Structural feature	TF-IDF, whether appear in title, first occurrence position in document, average occurrence position in document, distance between first and last occurrence positions, average distance between word adjacent occurrences, percentage of sentences that contains the word, TextRank score.
Semantic feature	LDA

Table 3.1: Features for the keyword classifier.

Keyword extraction: extracting appropriate keywords to represent the main ideas of a document is critical to the performance of the system. We have found that traditional keyword extraction approaches, such as TF-IDF based keyword extraction and TextRank [Mihalcea and Tarau, 2004], cannot produce satisfying results on real-world news data. For example, the TF-IDF based method measures a word's importance by its frequency in the document. Therefore, it is unable to extract keywords that have a relatively low frequency. The TextRank algorithm utilizes the word co-occurrence information and is able to handle such cases. However, its computation time increases significantly as document length increases. Another idea is to fine-tune a rule-based system to combine multiple keyword extraction strategies. Still, such type of system heavily relies on the quality of the rules and often generalizes poorly.

To efficiently and accurately extract keywords, we trained a binary classifier to determine whether a word is a keyword to a document. In particular, we manually labeled the keywords of $10,000+$ documents, including $20,000+$ positive keyword samples and $350,000+$ negative samples. Each word is transformed into a multi-view feature vector. Table 3.1 lists the main features that we found to be critical to the binary classifier. For the LDA feature vector, we trained a 1000-dimensional LDA model based on news data collected from January 1, 2016 to May 31, 2016 that

contains 300,000+ documents. The training process costs 30 hours.

After acquiring the features for each word. A straightforward idea is to input them directly to a Logistic Regression (LR) classifier. However, as a linear classifier, Logistic Regression relies on careful feature engineering. To reduce the impact of human bias in handcrafted features, we combine a Gradient Boosting Decision Tree (GBDT) with a LR classifier to get a keyword/non-keyword classification result, as shown in Fig. 3.4. The GBDT component is trained to automatically discover useful cross features or combinations and discretize continuous features. The output of the GBDT will serve as the input to the LR classifier. The LR classifier will determine whether a word is a keyword for the current document. We also tried SVM as the classifier in the second stage instead of LR and observed similar performance. It is also worth mentioning that on another Chinese keyword extraction dataset, our classifier achieved a precision of 0.83 and a recall of 0.76, while they are 0.72 and 0.76 respectively if we exclude the GBDT component. Note that we can still employ off-the-shelf tools, such as RAKE [Rose *et al.*, 2010], to perform keyword extraction. When abundant training data is available, our proposed keyword extraction approach could enhance the performance of event extraction in later stages, but it is not required by *Story Forest* system.

3.3.2 Event Extraction by EventX

After document preprocessing, we need to extract events. Event extraction here is essentially a fine-tuned document clustering procedure to group conceptually similar documents into events. Although clustering studies are often subjective in nature, we show that our carefully designed procedure can significantly improve the accuracy of event clustering, conforming to human understanding, based on a manually labeled news dataset. To handle the high accuracy requirement for long news text clustering, we propose *EventX*, a 2-layer clustering approach based on both keyword graphs and document graphs. The major steps in *EventX* include keyword co-occurrence graph construction, first-layer topic clustering based on keyword co-occurrence graph, and second-layer event extraction based on document relationship graph.

Note that the concept of “event" in this chapter and our proposed dataset is fundamentally different from the “event mentions” in cross-document event coreference [Lee *et al.*, 2012]. In our case, no matter how many event mentions can be found in an article, there is always a single major event that the news article is intended to report and cover or that triggers this report, with its narratives focusing on this major event. The task of *EventX* is to extract events by identify whether a group of documents are

ALGORITHM 1: EventX Event Extraction Algorithm

Input: A set of news documents $\mathcal{D} = \{d_1, d_2, ..., d_{|\mathcal{D}|}\}$, with extracted features described in Sec. 3.3.1; a pre-trained document pair relationship classifier.

Output: A set of events $E = \{\mathcal{E}_1, \mathcal{E}_2, ..., \mathcal{E}_{|E|}\}$.

1: Construct a keyword co-occurrence graph $\mathcal{G}$ of all documents' keywords. Connect w_i and w_j by an undirected edge $e_{i,j}$ if the times that the keywords w_i and w_j co-occur exceed a certain threshold δ_e, and $\Pr\{w_j|w_i\}$, $\Pr\{w_i|w_j\}$ are bigger than another threshold δ_p.

2: Split $\mathcal{G}$ into a set of small and strongly connected keyword communities $C = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_{|C|}\}$, based on the community detection algorithm [Ohsawa *et al.*, 1998]. The algorithm keeps splitting a graph by iteratively delete edges with high betweenness centrality score, until a stop condition is satisfied.

3: **for** each keyword community $\mathcal{C}_i$, $i = 1, \ldots, |C|$ **do**

4: Retrieve a subset of documents $\mathcal{D}_i$ which is highly related to this keyword community by calculating the cosine similarity between the TF-IDF vector of each document and that of the keyword community, and comparing it to a threshold δ.

5: Connect document pairs in $\mathcal{D}_i$ to form a document relationship graph $\mathcal{G}_i^d$ using the document pair relationship classifier.

6: Split $\mathcal{G}_i^d$ into a set of document communities $C_i^d = \{\mathcal{C}_{i,1}^d, \mathcal{C}_{i,2}^d, ..., \mathcal{C}_{i,|C_i^d|}^d\}$, based on the community detection algorithm. Each community represents an event.

7: **end for**

reporting the same breaking news with different narratives or paraphrasing, instead of identifying what event mentions a document contains. In the following, we provide a detailed description on the inner workings of *EventX*.

Construct Keyword Co-occurrence Graph

Algorithm 1 shows the detailed steps of *EventX*. We show that our two-layered approach significantly improves the accuracy of event clustering on long news articles.

Given a news corpus $\mathcal{D}$, we construct a keyword co-occurrence graph [Sayyadi and Raschid, 2013] $\mathcal{G}$. Each node in $\mathcal{G}$ is a keyword w extracted by the scheme described in Sec. 3.3.1, and each undirected edge $e_{i,j}$ indicates that w_i and w_j have co-occurred in a same document. Edges that satisfy two conditions will remain and other edges will be pruned: the times of co-occurrence shall be above a minimum threshold δ_e (we set $\delta_e = 2$ in our experiments), and the conditional probabilities of the occurrence $\Pr\{w_j|w_i\}$ and $\Pr\{w_i|w_j\}$ also need to be greater than a predefined threshold δ_p (we use 0.15). The conditional probability $\Pr\{w_j|w_i\}$ is calculated as

$$\Pr\{w_i|w_j\} = \frac{DF_{i,j}}{DF_j}, \tag{3.1}$$

where $DF_{i,j}$ represents the number of documents that contain both keyword w_i and

w_j, and DF_j is the document frequency of keyword w_j. It represents the probability that w_i occurs in a document if that document contains w_j.

Topic Clustering on Keyword Co-occurrence Graph

Next, we perform community detection on the constructed keyword co-occurrence graph. The goal is to split the whole keyword graph $\mathcal{G}$ into communities $C = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_{|C|}\}$, where each community (or subgraph) $\mathcal{C}_i$ contains the keywords for a certain topic i (to which multiple events may be associated). The intuition is that keywords related to a common topic usually will appear frequently in documents belonging to that topic. For example, documents belonging to the topic "2016 U.S. presidential election" will frequently mention keywords such as "Donald Trump", "Hillary Clinton", "election" and so on. Such highly related keywords will be connected to each other in the keyword graph and form dense subgraphs, while keywords that are not highly related will have sparse connections or no connection. Our objective here is to extract dense keyword subgraphs associated with different topics.

The benefit of using community detection in the keyword graph is that it allows each keyword to appear in multiple communities. In reality it is not unusual to have one keyword appearing under multiple topics. We also tried another method of clustering keywords by *Word2Vec*. But the performance is worse than community detection based on co-occurrence graphs. The main reason is that when clustering with pre-trained word vectors, words with similar semantic meanings are more likely be grouped together. However, unlike articles in a specialized domain, news topics from the open domain often contain keywords with diverse semantic meanings.

To detect keyword communities, we utilize the *betweenness centrality score* [Sayyadi and Raschid, 2013] of edges to measure the strength of each edge in the keyword graph. The betweenness score of an edge is defined as the number of shortest paths between all pairs of nodes that pass through it. An edge between two communities is expected to have a high betweenness score. Edges with high betweenness score will be removed iteratively to divide the communities. If two edges have the same betweenness score, the one with lower conditional probability will be removed. The conditional probability of edge $e_{i,j}$ is calculated as $(\Pr\{w_i|w_j\} + \Pr\{w_j|w_i\})/2$. We calculate the betweenness score of edges using breadth first search (BFS) and iteratively remove edges with the highest betweenness score. Once a keyword graph is no longer fully connected, we continue the same process recursively on each connected component of the keyword graph. The splitting process ends if the number of nodes in each subgraph is smaller than a predefined threshold δ_g (we use 3), or the maximum betweenness score of all edges in the subgraph is smaller than a threshold based on the subgraph's size. We

Feature type	Description
Keyword	Number of common keywords in titles or contents, percentage of common keywords in titles or contents.
Textual similarity	TF-IDF (and TF) similarities of titles or contents, TF-IDF (and TF) similarities of the first N (N=1,2,3) sentences.
Semantic	LDA cosine similarity of two documents, the absolute value of the difference between the LDA vectors of the two documents.

Table 3.2: Features for document pair relationship classification.

refer interested readers to [Sayyadi and Raschid, 2013] for more details about community detection. We notice that the betweenness-based community detection algorithm is used as an off-the-shelf tool in *EventX* algorithm. There exist other community detection algorithms that are more efficient in terms of time complexity [Radicchi *et al.*, 2004], and we can easily switch to these algorithms if time efficiency is a concern.

After obtaining the keyword communities, we calculate the cosine similarities between each document and each keyword community. The documents are represented as TF-IDF vectors. Given a keyword community is essentially a bag of words, it can also be considered as a document. We assign each document to the keyword community with the highest similarity, as long as the similarity is also above a predefined threshold δ. At this point, we have finished clustering in the first layer, i.e., the documents are now grouped by topics.

Event Extraction based on Document Relationship Graph

After we partition documents into topics, we perform the second-layer document clustering within each topic to obtain fine-grained events. We also call this process *event clustering*. An event cluster only contains documents that talk about the same event. As mentioned before, since event clusters could vary dramatically in sizes, traditional unsupervised learning based algorithms, such as K-means, Non-negative Matrix Factorization, Hierarchical clustering, are not appropriate. Instead, we adopt a supervised-learning-guided clustering procedure in the second layer.

Specifically, in contrast with the keyword graph in the first layer, now we consider each document as a graph node, and try to connect document pairs that discuss the

same event [Liu *et al.*, 2019*a*]. In keyword co-occurrence graph, judging whether two keywords are related is achieved through the co-occurrence of keywords in documents. This is feasible because the granularity of a topic is relatively coarse and subjective compared with an event. However, simply combining unsupervised strategies to judge whether two documents are talking about the same event cannot produce satisfying results, because events are highly diverse and the clustering granularity should be adjusted accordingly.

To address the above challenges, we propose a semi-supervised clustering scheme which incorporates a document pair relationship classifier to construct a document relationship graph, and performs event extraction on that graph. Given a set of documents $\mathcal{D} = \{d_1, d_2, ..., d_{|\mathcal{D}|}\}$ within a topic cluster, for each pair of documents $\langle d_i, d_j \rangle$ that belongs to $\mathcal{D}$, we add an edge $e^d_{i,j}$ if they are talking about the same event. We trained a SVM classifier to determine whether a pair of documents are talking about the same event using multiple document pair features, such as cosine similarity of TF-IDF vectors, number of common keywords, LDA cosine similarity, as the input vector. Table 3.2 lists the main features we utilized to train the document pair relationship classifier. For each pair of documents within a same topic, we decide whether to draw an edge between them based on the prediction made by the document pair relationship classifier. Hence, documents in each topic $\mathcal{D}$ will form a document relationship graph $\mathcal{G}^d$. We then apply the same community detection algorithm mentioned above to such graphs. Naturally, each community within a topic cluster now represents an event. Note that the graph-based clustering on the second layer is highly efficient, since the number of documents contained in each topic is significantly less after the first-layer document clustering.

In a nutshell, our two-layered scheme first groups documents into topics based on keyword community detection and then further groups the documents within each topic into fine-grained events. It has multiple advantages compared with a number of existing clustering algorithms:

1. It does not require a pre-defined number of event clusters. This is critical in real-world applications, as for real-world data such as daily news, the exact number of events is almost always unknown and varies dramatically. Most existing clustering algorithms require the number of clusters as a parameter, or need to carefully choose parameters that control the clustering granularity. For EventX, even though we still need to specify a stopping criteria in the community detection algorithm, the threshold hyper parameters in our algorithm are much more stable and insensitive to different sizes of news documents. There-

fore, these thresholds can be selected using grid search, and do not require to be updated frequently for news articles from a different date.

2. Our algorithm can adaptively determine the granularity of different events based on the pre-trained document pair relationship classifier, as well as community detection on the document relationship graph.

3. The performance of our algorithm is less sensitive to the parameter settings in the first layer, because the granularity of a topic is usually subjective. We can set relatively low thresholds in the first layer to ensure that documents belonging to the same event are grouped together, then extract the fine-grained events in the second layer.

4. The performance of event clustering can be improved by adding more features and training data to the document pair relationship classifier, without affecting keyword extraction and topic clustering.

5. The two-layered scheme of our algorithm makes event clustering highly efficient. Directly performing relationship classification within a document corpora is not feasible, as the time complexity is at least $\mathcal{O}(n_d^2)$ where n_d is the number of documents. By utilizing the two-layered scheme, the documents will be split into multiple topics (each topic usually contains 1 to 100 documents). Therefore, the time complexity for the second-layer event extraction is approximately $\mathcal{O}(n_d)$. The total time complexity that taking both two layers of clustering into account is still much smaller than $\mathcal{O}(n_d^2)$, as we will discuss in Sec. 3.4.4

3.3.3 Growing Story Trees Online

Given the set of extracted events for a particular topic, we further organize these events into multiple stories under this topic in an online manner. Each story is represented by a *Story Tree* to characterize the evolving structure of that story. Upon the arrival of a new event and given an existing story forest, our online algorithm to grow the story forest mainly involves two steps: a) identifying the story tree to which the event belongs; b) updating the found story tree by inserting the new event at the right place. If this event does not belong to any existing story, we create a new story tree.

a) Identifying the related story tree. Given a set of new events $E_t = \{\mathcal{E}_1, \mathcal{E}_2, ..., \mathcal{E}_{|E_t|}\}$ at time period t and an existing story forest $\mathcal{F}_{t-1} = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_{|\mathcal{F}_{t-1}|}\}$ that has been formed during previous $t-1$ time periods, our objective is to assign

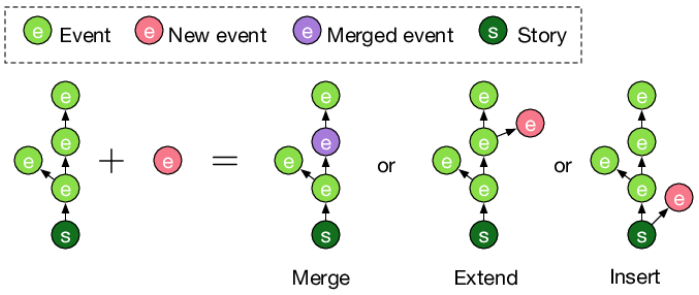


Figure 3.5: Three types of operations to place a new event into its related story tree.

each new event $\mathcal{E} \in E_t$ to an existing story tree $\mathcal{S} \in \mathcal{F}_{t-1}$. If no story in the current story forest matches that event, a new story tree will be created and added to the story forest.

We apply a two-step strategy to decide whether a new event $\mathcal{E}$ belongs to an existing story tree $\mathcal{S}$ formed previously. First, as described at the end of Sec. 3.3.2, event $\mathcal{E}$ has its own keyword set $\mathcal{C}_\mathcal{E}$. Similarly, for the existing story tree $\mathcal{S}$, there is an associated keyword set $\mathcal{C}_\mathcal{S}$ that is a union of all the keyword sets of the events in that tree.

Then, we can calculate the compatibility between event $\mathcal{E}$ and story tree $\mathcal{S}$ as the Jaccard similarity coefficient between $\mathcal{C}_\mathcal{S}$ and $\mathcal{C}_\mathcal{E}$: $\text{compatibility}(\mathcal{C}_\mathcal{S}, \mathcal{C}_\mathcal{E}) = \frac{|\mathcal{C}_\mathcal{S} \cap \mathcal{C}_\mathcal{E}|}{|\mathcal{C}_\mathcal{S} \cup \mathcal{C}_\mathcal{E}|}$. If the compatibility is bigger than a threshold, we further check whether at least a document in event $\mathcal{E}$ and at least a document in story tree $\mathcal{S}$ share n or more common words in their titles (with stop words removed). If yes, we assign event $\mathcal{E}$ to story tree $\mathcal{S}$. Otherwise, they are not related. In our experiments, we set $n = 1$. If the event $\mathcal{E}$ is not related to any existing story tree, a new story tree will be created.

b) Updating the related story tree. After a related story tree $\mathcal{S}$ has been identified for the incoming event $\mathcal{E}$, we perform one of the 3 types of operations to place event $\mathcal{E}$ in the tree: *merge*, *extend* or *insert*, as shown in Fig. 3.5. The *merge* operation merges the new event $\mathcal{E}$ into an existing event node in the tree. The *extend* operation will append event $\mathcal{E}$ as a child node to an existing event node in the tree. Finally, the *insert* operation directly appends event $\mathcal{E}$ to the root node of story tree $\mathcal{S}$. Our system chooses the most appropriate operation to process the incoming event based on the following procedures.

Merge: we merge $\mathcal{E}$ with an existing event in the tree, if they essentially talk about the same event. This can be achieved by checking whether the centroid documents of the two events are talking about the same thing using the document-pair relationship classifier described in Sec. 3.3.2. The centroid document of an event is simply the concatenation of all the documents in the event.

Extend *and* ***Insert***: if event $\mathcal{E}$ does not overlap with any existing event, we will find the parent event node in $\mathcal{S}$ to which it should be appended. We calculate the *connection strength* between the new event $\mathcal{E}$ and each existing event $\mathcal{E}_j \in \mathcal{S}$ based on three factors: 1) the time distance between $\mathcal{E}$ and $\mathcal{E}_j$, 2) the compatibility of the two events, and 3) the *storyline coherence* if $\mathcal{E}$ is appended to $\mathcal{E}_j$ in the tree, i.e.,

$$\begin{aligned}\text{ConnectionStrength}(\mathcal{E}_j, \mathcal{E}) := \ &\text{compatibility}(\mathcal{E}_j, \mathcal{E}) \times \\ &\text{coherence}(\mathcal{L}_{\mathcal{S}\to\mathcal{E}_j\to\mathcal{E}}) \times \text{timePenalty}(\mathcal{E}_j, \mathcal{E}).\end{aligned} \tag{3.2}$$

Now we explain the three components in the above equation one by one. *First*, the compatibility between two events $\mathcal{E}_i$ and $\mathcal{E}_j$ is given by

$$\text{compatibility}(\mathcal{E}_i, \mathcal{E}_j) = \frac{\text{TF}(d_{c_i}) \cdot \text{TF}(d_{c_j})}{\|\text{TF}(d_{c_i})\| \cdot \|\text{TF}(d_{c_j})\|}, \tag{3.3}$$

where d_{c_i} is the centroid document of event $\mathcal{E}_i$.

Furthermore, the storyline of $\mathcal{E}_j$ is defined as the path in $\mathcal{S}$ starting from the root node of $\mathcal{S}$ ending at $\mathcal{E}_j$ itself, denoted by $\mathcal{L}_{\mathcal{S}\to\mathcal{E}_j}$. Similarly, the storyline of $\mathcal{E}$ appended to $\mathcal{E}_j$ is denoted by $\mathcal{L}_{\mathcal{S}\to\mathcal{E}_j\to\mathcal{E}}$. For a storyline $\mathcal{L}$ represented by a path $\mathcal{E}^0 \to \ldots \to \mathcal{E}^{|\mathcal{L}|}$, where $\mathcal{E}^0 := \mathcal{S}$, its *coherence* [Xu *et al.*, 2013] measures the theme consistency along the storyline, and is defined as

$$\text{coherence}(\mathcal{L}) = \frac{1}{|\mathcal{L}|} \sum_{i=0}^{|\mathcal{L}|-1} \text{compatibility}(\mathcal{E}^i, \mathcal{E}^{i+1}), \tag{3.4}$$

Finally, the bigger the time gap between two events, the less possible that the two events are connected. We thus calculate time penalty by

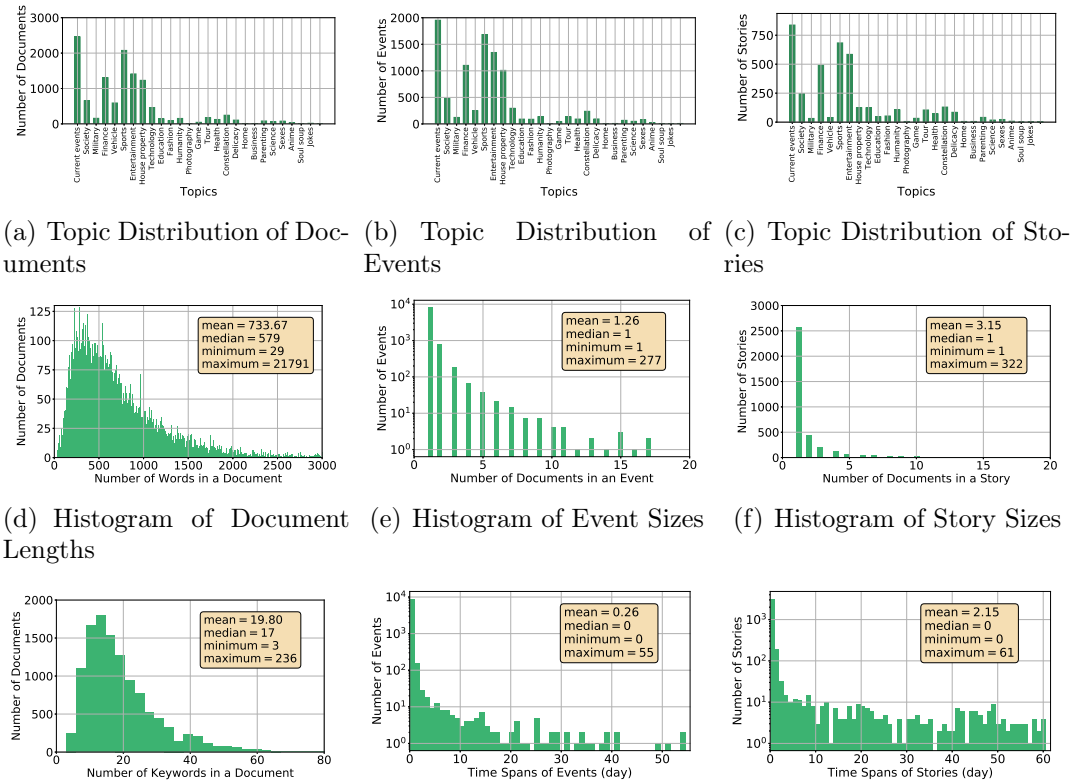
$$\text{timePenalty}(\mathcal{E}_j, \mathcal{E}) = \begin{cases} e^{\delta \cdot (t_{\mathcal{E}_j} - t_{\mathcal{E}})} & \text{if } t_{\mathcal{E}_j} - t_{\mathcal{E}} < 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

where $t_{\mathcal{E}_j}$ and $t_{\mathcal{E}}$ are the timestamps of event $\mathcal{E}_j$ and $\mathcal{E}$ respectively. The timestamp of an event is the minimum timestamp of all the documents in the event.

We calculate the connection strength between the new event $\mathcal{E}$ and every event node $\mathcal{E}_j \in \mathcal{S}$ using (3.2), and append event $\mathcal{E}$ to the existing $\mathcal{E}_j$ that leads to the maximum connection strength. If the maximum connection strength is lower than a threshold value, we *insert* $\mathcal{E}$ into story tree $\mathcal{S}$ by directly appending it to the root node of $\mathcal{S}$. In other words, *insert* is a special case of *extend*.

3.4 Performance Evaluation

We conduct two groups of experiments. The first group independently evaluates *EventX*, since it is a core novelty within our Story Forest system. We report the



(a) Topic Distribution of Documents

(b) Topic Distribution of Events

(c) Topic Distribution of Stories

(d) Histogram of Document Lengths

(e) Histogram of Event Sizes

(f) Histogram of Story Sizes

(g) Histogram of Document Keyword Numbers

(h) Histogram of Event Time Spans by Day

(i) Histogram of Story Time Spans by Day

Figure 3.6: The characteristics of the introduced Chinese News Event and Story dataset.

performance of *EventX* on a manually labeled Chinese News Events dataset and the openly available 20 Newsgroups dataset. We then compare *EventX* against several clustering baselines. In the second group, we add the story visualization structures and examine the overall performance of Story Forest on a much larger news documents dataset, i.e., the Chinese News Corpus dataset. We also investigate the effects of various hyper-parameters in *EventX* as well as the algorithm complexity of the entire system.

3.4.1 News Datasets

To help evaluate Story Forest, we create the *Chinese News Corpus* dataset by collecting 60 GB of Chinese news documents (approximately 1.5 millions of news articles) from major Internet news providers in China, such as Tencent and Sina, in a three-month period from October 1, 2016 to December 31, 2016 covering different topics in the open domain. The entire Story Forest system is tested on the full 60 GB of Chinese news documents and we employ human evaluators to analyze its overall

performance.

We then sample a smaller subset from these documents to create the *Chinese News Events* dataset, which is manually labeled with the truth events and stories, by editors and product managers at Tencent. It is also, to the best of our knowledge, the first Chinese dataset for event extraction evaluation. In the creation of *Chinese News Events* dataset, we assume that 1) each article only has one event ID, i.e., associated with the breaking news that leads to this article; 2) different events from the same story are clearly distinguishable, partly because they usually follow the progressing timeline of real-world affairs; 3) each article only belongs to one news story indicated by a unique story ID (we ignore the complex storylines and entangled plots such as those in a fiction). We train and evaluate the *EventX* algorithm on this labeled dataset of 11,748 Chinese news articles. When evaluating the whole Story Forest system on the Chinese News Corpus dataset, documents belonging to the Chinese News Events dataset are removed.

Fig. 3.6 shows some statistics of the new dataset. Fig. 3.6(a), Fig. 3.6(b) and Fig. 3.6(c) demonstrate how many documents, events and stories each topic contains. We can see that the distributions are highly skewed. The top 3 largest topics are "entertainment", "sports" and "current events". Fig. 3.6(d) shows the histogram of word counts in documents. The average number of words in a document is 733.67 and the longest document has 21,791 words. From Fig. 3.6(g), we can see that the average number of keywords extracted from each document is 19.8. In terms of the sizes of events and stories, the histograms of the number of documents in each event and that in each story are shown in Fig. 3.6(e) and Fig. 3.6(f), respectively. We can easily observe that for real-world news data, a majority of events only contain a single document: the average size of an event is 1.26 (documents), while the largest event contains 277 documents. The average size of a story is 3.15 (documents) and the largest story contains 322 documents. Finally, Fig. 3.6(h) and Fig. 3.6(i) plot the distributions of the time spans of each event and story. The average time span of an event is 0.26 days, and that of a story is 2.15 days. These statistics help to justify the assumptions we made above when labeling the dataset—most breaking news events and stories have relatively short lifespans with simple narratives (in contrast to fictions or scientific articles).

3.4.2 Evaluation of EventX

Evaluation Datasets

We utilize the following datasets for evaluating *EventX*:

- **Chinese News Events (CNE)**. This is the full dataset that contains 11,748 news articles belonging to 9,327 events. The average number of words in documents is 733.67.

- **Chinese News Events Subset 1**. It is created by removing events containing less than 2 documents in the Chinese News Events dataset. It contains 3,557 news articles belonging to 1,136 events. The average number of words in documents is 768.07.

- **Chinese News Events Subset 2**. We further filter events that contain less than 4 documents in the Chinese News Events dataset. This dataset contains 1,456 news articles belonging to 176 events. The average length of documents is 760.04 words.

- **20 Newsgroups**[2]. This is a classic dataset for the evaluation of text clustering algorithms. It consists of 18,821 documents that belong to 20 different news groups. The average length of documents is 137.85 words.

For Chinese datasets, after word segmentation, we extract keywords from each document using the supervised approach mentioned in Sec. 3.3.1. The binary classifier to classify whether a word is a keyword to a document is trained on a dataset that we manually labeled. It contains the keywords of 10,000+ documents, including 20,000+ positive keyword samples and 350,000+ negative samples. On another Chinese keyword extraction evaluation dataset, our classifier achieved a precision of 0.83 and a recall of 0.76, while they are 0.72 and 0.76 respectively if we exclude the GBDT component. The average number of keywords for each document in our proposed Chinese News Events dataset is 19.8. For English documents, we extracted keywords using RAKE [Rose *et al.*, 2010].

Evaluation Metrics

The evaluation metrics we used include: Homogeneity (H), Completeness (C), V-measure score (V) [Rosenberg and Hirschberg, 2007] and Normalized Mutual Information (NMI). Homogeneity and completeness are intuitive evaluation metrics based on conditional entropy analysis. Homogeneity is larger if each cluster contains only members from a single class. The completeness is maximized if all members of a ground truth class are assigned to the same cluster. Homogeneity and completeness

[2] `http://qwone.com/~jason/20Newsgroups/`

scores are defined as:

$$H = 1 - \frac{\sum_{c,k} n_{c,k} \log \frac{n_{c,k}}{n_k}}{\sum_c n_c \log \frac{n_c}{N}}, \tag{3.6}$$

$$C = 1 - \frac{\sum_{c,k} n_{c,k} \log \frac{n_{c,k}}{n_c}}{\sum_k n_k \log \frac{n_k}{N}}, \tag{3.7}$$

where n_c is the number of documents in class c, and n_k is the number of documents in cluster k. $n_{c,k}$ is the number of documents in class c as well as in cluster k, and N is the number of total documents in the dataset. The V-measure is the harmonic mean between homogeneity and completeness:

$$\mathrm{V} = \frac{2 \times \mathrm{H} \times \mathrm{C}}{\mathrm{H} + \mathrm{C}}. \tag{3.8}$$

Normalized mutual information [Estévez *et al.*, 2009] is also widely used for text clustering evaluation. It measures the amount of statistical information shared by the ground truth cluster labels and the cluster assignment results. The NMI is 1 if the cluster results perfectly match the truth labels, and it is close to 0 when the cluster labels are randomly generated. Normalized mutual information is formally defined as:

$$NMI = \frac{\sum_{c,k} n_{c,k} \log \frac{n_{c,k} \cdot N}{n_c \cdot n_k}}{\sqrt{(\sum_c n_c \log \frac{n_c}{N})(\sum_k n_k \log \frac{n_k}{N})}} \tag{3.9}$$

Methods for Comparison

We compare *EventX* to other document clustering methods, as well as methods that utilize both word and document clusters:

- **KeyGraph**: the KeyGraph algorithm [Sayyadi and Raschid, 2013] clusters documents based on a keyword co-occurrence graph, without the second-layer clustering based on document relationship graphs and document pair relationship classifier.

- **Co-clustering**: this algorithm takes in a term-document matrix and simultaneously maps rows to row-clusters and columns to column-clusters. Then, it calculates the mutual information difference between the current row and column clusters and iteratively updates both clusters until this difference is below a threshold [Dhillon *et al.*, 2003].

- **Word-to-Doc Clustering**: this algorithm takes in a joint probability distribution, typically a term-document matrix. It first computes word-clusters to

Table 3.3: Comparing different algorithms on Chinese News Events (CNE) dataset.

Algorithm	CNE ($K = 2331$) H	C	V	NMI	CNE ($K = 4663$) H	C	V	NMI	CNE ($K = 9327$) H	C	V	NMI
EventX	**0.990**	0.954	**0.972**	**0.972**	**0.990**	0.954	**0.972**	**0.972**	**0.990**	0.954	**0.972**	**0.972**
KeyGraph	0.811	**0.959**	0.879	0.882	0.811	**0.959**	0.879	0.882	0.811	0.959	0.879	0.882
Co-Clustering	0.962	0.773	0.857	0.863	0.960	0.839	0.896	0.898	0.958	0.901	0.929	0.929
Word-Doc	0.965	0.820	0.887	0.890	0.964	0.891	0.926	0.927	0.959	**0.974**	0.966	0.966
NMF	0.514	0.954	0.668	0.700	0.403	0.940	0.564	0.615	0.231	0.910	0.368	0.458
K-means+LDA	0.266	0.105	0.151	0.167	0.355	0.126	0.186	0.211	0.466	0.147	0.224	0.262
K-means+TF-IDF	0.445	0.173	0.250	0.278	0.470	0.166	0.245	0.279	0.485	0.154	0.234	0.273

Table 3.4: Comparing different algorithms on Chinese News Events Subset 1.

Algorithm	CNE Subset 1 ($K = 568$) H	C	V	NMI	CNE Subset 1 ($K = 1136$) H	C	V	NMI	CNE Subset 1 ($K = 2272$) H	C	V	NMI
EventX	**0.973**	0.841	**0.902**	**0.905**	0.973	0.841	**0.902**	**0.905**	0.973	0.841	0.902	0.905
KeyGraph	0.837	0.846	0.842	0.842	0.837	0.846	0.842	0.842	0.837	0.846	0.842	0.842
Co-Clustering	0.868	0.777	0.820	0.821	0.860	0.850	0.855	0.855	0.847	0.911	0.878	0.879
Word-to-Doc	0.897	0.822	0.858	0.859	0.876	0.914	0.895	0.895	0.850	**0.979**	**0.910**	**0.912**
NMF	0.491	**0.913**	0.638	0.669	0.351	**0.931**	0.510	0.572	0.156	0.714	0.256	0.334
K-means+LDA	0.886	0.351	0.503	0.558	0.956	0.340	0.502	0.570	0.989	0.314	0.477	0.558
K-means+TF-IDF	0.928	0.363	0.522	0.580	**0.975**	0.338	0.502	0.574	**0.991**	0.313	0.476	0.557

capture the most information about the documents. Relying on this new cluster-document matrix, the algorithm recursively merges the documents into bigger clusters, such that the mutual information loss between document clusters and word clusters is minimized [Slonim and Tishby, 2000].

- **Non-negative Matrix Factorization (NMF)**: this algorithm coverts a corpus into a term-document matrix where each element in the matrix is the TF-IDF value of a word in a document. The factorized basis vectors correspond to different clusters. By examining the largest component of a document along any of the vectors, we can decide the cluster membership of that document [Xu *et al.*, 2003].

- **K-means with LDA**: extract the 1000-dimensional LDA vector of each document, and cluster them by the K-means algorithm [Jain, 2010], which is probably the most widely used method for clustering.

- **K-means with TF-IDF**: represent each document by TF-IDF vector, and cluster by K-means.

For the compared baseline methods, we vary the number of clusters for each dataset. For algorithms that require iterative updates, we set a maximum iteration

Table 3.5: Comparing different algorithms on Chinese News Events Subset 2.

Algorithm	CNE Subset 2 ($K = 88$)				CNE Subset 2 ($K = 176$)				CNE Subset 2 ($K = 352$)			
	H	C	V	NMI	H	C	V	NMI	H	C	V	NMI
EventX	**0.964**	0.680	0.798	0.810	**0.964**	0.680	0.798	0.810	**0.964**	0.680	0.798	0.810
KeyGraph	0.691	0.767	0.717	0.728	0.691	0.767	0.717	0.728	0.691	0.767	0.717	0.728
Co-Clustering	0.783	0.728	0.755	0.755	0.750	0.801	0.775	0.775	0.723	0.867	0.788	0.792
Word-to-Doc	0.886	0.765	**0.821**	**0.823**	0.805	**0.865**	**0.834**	**0.835**	0.731	**0.927**	**0.817**	**0.823**
NMF	0.753	**0.853**	0.800	0.802	0.801	0.809	0.805	0.805	0.745	0.802	0.773	0.773
K-means+LDA	0.783	0.430	0.555	0.581	0.890	0.408	0.559	0.602	0.940	0.380	0.541	0.598
K-means+TF-IDF	0.857	0.469	0.606	0.634	0.928	0.424	0.582	0.627	0.959	0.379	0.543	0.603

Table 3.6: Comparing different algorithms on 20 Newsgroups dataset.

Algorithm	20 Newsgroups ($K = 10$)				20 Newsgroups ($K = 20$)				20 Newsgroups ($K = 40$)			
	H	C	V	NMI	H	C	V	NMI	H	C	V	NMI
EventX	**0.996**	0.340	**0.507**	**0.582**	**0.996**	0.340	**0.507**	**0.582**	**0.996**	0.340	**0.507**	**0.582**
KeyGraph	0.995	0.340	0.507	0.581	0.995	0.340	0.507	0.581	0.995	0.340	0.507	0.581
Co-Clustering	0.421	0.318	0.362	0.366	0.374	0.368	0.371	0.371	0.351	0.423	0.384	0.386
Word-to-Doc	0.578	0.306	0.400	0.421	0.540	0.390	0.453	0.459	0.515	**0.464**	0.488	0.489
NMF	0.315	**0.529**	0.395	0.408	0.396	**0.468**	0.429	0.431	0.456	0.413	0.433	0.434
K-means+LDA	0.245	0.352	0.368	0.293	0.333	0.352	0.342	0.342	0.321	0.279	0.298	0.299
K-means+TF-IDF	0.350	0.282	0.312	0.194	0.368	0.264	0.308	0.159	0.332	0.301	0.316	0.236

of 20 (which proved to be enough for convergence). Notice that our *EventX* algorithm and the KeyGraph algorithm do not require a pre-defined number of clusters.

The *EventX* algorithm needs to pre-train a document relationship classifier. For the Chinese News Events dataset and subsets, we trained a document-pair relationship classifier using $5,000$ labeled news pairs, collected from the web from January 01, 2017 to April 04, 2017. For the 20 Newsgroups dataset, we used the training set[3] to construct a document-pair relationship dataset and trained a classifier, then evaluated the algorithms on the test set.

Comparison with Existing Methods

Table 3.3 shows the performance of different algorithms on the four datasets with different number of clusters K. As we can see, our approach achieves the best V-measure and NMI scores in most cases. And our method achieves overall high homogeneity scores among all datasets. This implies that most event clusters we obtain are pure: each event only contains documents that talk about the same event. In comparison, the homogeneity for other methods varies widely across datasets. The reason is that we adopted two layers of graph-based clustering algorithms to extract events at a

[3]The splitting of training and testing set for the 20 Newsgroups is the same with `http://csmining.org/index.php/id-20-newsgroups.html`.

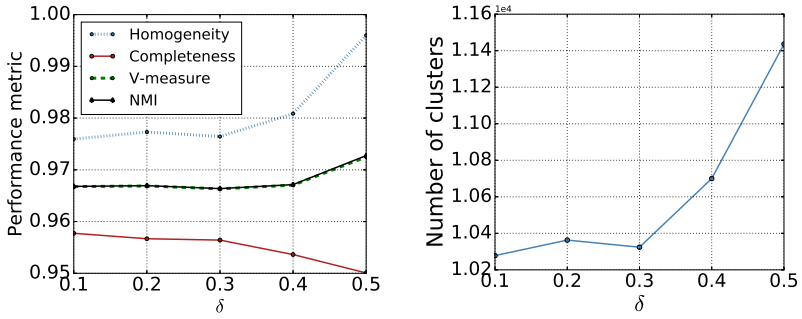
more appropriate granularity. For completeness, even though our approach scored a little bit lower than KeyGraph, it is more or less expected. As we further split the topic clusters in the second layer document relationship graph, it is more likely for documents labeled as one class to appear in different clusters, simply because with a two-layered clustering scheme we have more clusters. Considering the significant improvements of homogeneity, we believe the slightly lower completeness score would not strongly affect the real-world performance of our model.

The performance of Word-to-Doc Clustering algorithm is slightly better than our algorithm on the Chinese News Events Subset 2. However, our algorithm does not need the number of clusters ahead of time, while the performance of the Word-to-Doc Clustering algorithm is highly dependent on this parameter. For real-world news data, the event size distribution is highly skewed like the Chinese News Events full dataset, rather than the subsets.

Considering the results on the 20 Newsgroups dataset. Our algorithm achieves better performance than the other algorithms in terms of the Homogeneity, V-measure score and NMI. We can also observe that the completeness of our algorithm is relatively low on the 20 Newsgroups dataset, while the homogeneity is still high. The reason is that the documents in 20 Newsgroups dataset are grouped according to topics rather than events. Therefore, two documents belonging to the same cluster may have entirely different keywords, and our algorithm will assign them two different cluster labels. For the same reason, we can see that the performance of KeyGraph is similar to *EventX* on the 20 Newsgroups dataset. After the first layer keyword co-occurrence graph based clustering, the clusters are already pure and the second layer document relationship graph based clustering would not further split it.

Influence of Parameters

The main parameter that will influence the final clustering result of *EventX* is the threshold δ that assigns each document to a keyword subgraph in the fist layer clustering step. Fig. 3.7 shows its influence to the clustering performance on the Chinese News Events dataset. As we can see, the performance is quite robust when $\delta \leq 0.3$. If we keep increasing δ, the number of clusters increases, leading to an increase in homogeneity and a decrease in completeness. The reason is that, when δ is within a reasonable range (usually between 0.1 to 0.3), the performance of our algorithm is insensitive to it, as the first layer keyword co-occurrence graph based clustering will split documents into topics, and the value of δ will only influence the granularity of the clustered topics. The events within each topic will not be divided, and they will be extracted by the second layer clustering procedure. If δ keeps increasing, at some



(a) Clustering performance (b) Number of clusters

Figure 3.7: The influence of parameter δ to the clustering performance and number of clusters on the Chinese News Events dataset.

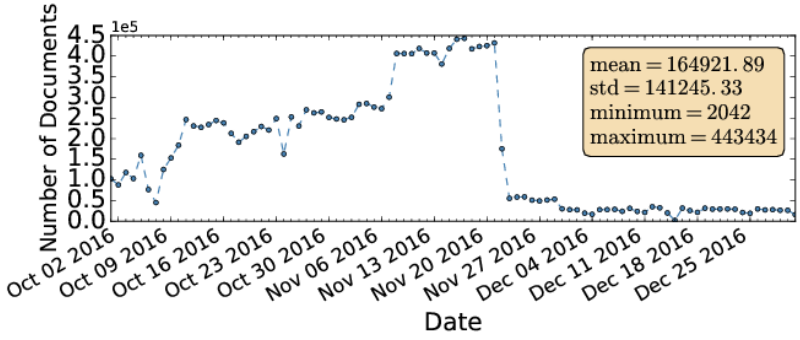


Figure 3.8: The number of documents on different days in the Story Forest evaluation dataset.

point, events will split in the first layer, resulting in an increase in homogeneity and a decrease in completeness. For other parameters, such as the δ_e, δ_p and δ_g we defined in Sec. 3.3.2 and Sec. 3.3.2, they are parameters that influence the construction of keyword co-occurrence graph and subgraphs. Our event extraction algorithm is also insensitive to these parameters as long as they are within a reasonable range. The reason is the same with δ.

3.4.3 Evaluation of Story Forest

Now we evaluate the complete Story Forest system. Fig. 3.8 shows the amounts of documents on different days in the larger 60 GB dataset. The average number of documents in one day during that period is 164, 922. For the following experiments, we use the data in the first 7 days for parameter tuning. The remaining data serves as the test set.

We test different event timeline and story generation algorithms on the large 3-month news dataset through pilot user evaluation. To make fair comparisons, the

	Tree	Flat	Thread	Timeline	Graph
Correct edges	**82.8**%	73.7%	66.8%	58.3%	32.9%
Consistent paths	**77.4**%	–	50.1%	29.9%	–
Best structure	**187**	88	84	52	19

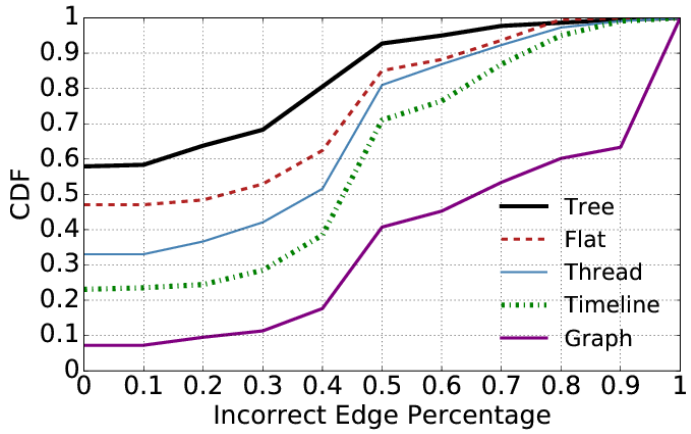
Table 3.7: Comparing different story structure generation algorithms.

same preprocessing and event extraction procedures before developing story structures are adopted for all methods, with 261 stories detected from the dataset. The only difference is how to construct the story structure given a set of event nodes. We compare our online Story Forest system with the following existing algorithms:

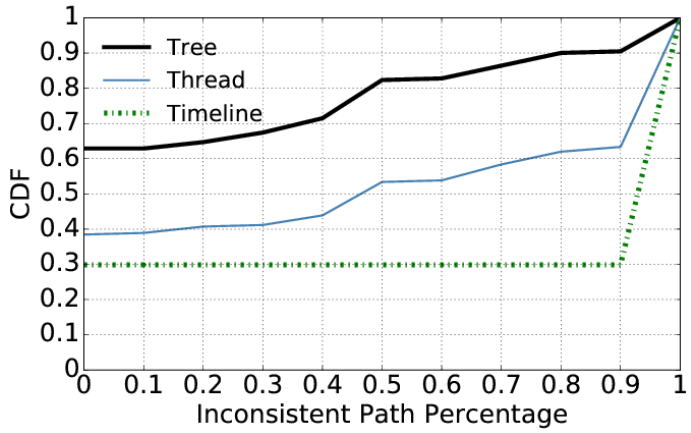
- **Flat Cluster (Flat):** this method clusters related events into a story without revealing the relationships between events, which approximates some previous works in TDT [Yang *et al.*, 2002; Allan *et al.*, 1998].
- **Story Timeline (Timeline):** this method organizes events linearly according to the timestamps of events [Sayyadi *et al.*, 2009; Sayyadi and Raschid, 2013].
- **Story Graph (Graph):** this method calculates a connection strength for every pair of events and connect the pair if the score exceeds a threshold [Yang *et al.*, 2009].
- **Event Threading (Thread):** this algorithm appends each new event to its most similar earlier event [Nallapati *et al.*, 2004]. The similarity between two events is measured by the TF-IDF cosine similarity of the event centroids.

We enlisted 10 human reviewers, including product managers, software engineers and senior undergraduate students, to blindly evaluate the results given by different approaches. Each individual story was reviewed by 3 different reviewers. When the reviewers' opinions are different, they will discuss to give a final result. For each story, the reviewers answered the following questions for each of the 5 different structures generated by different schemes:

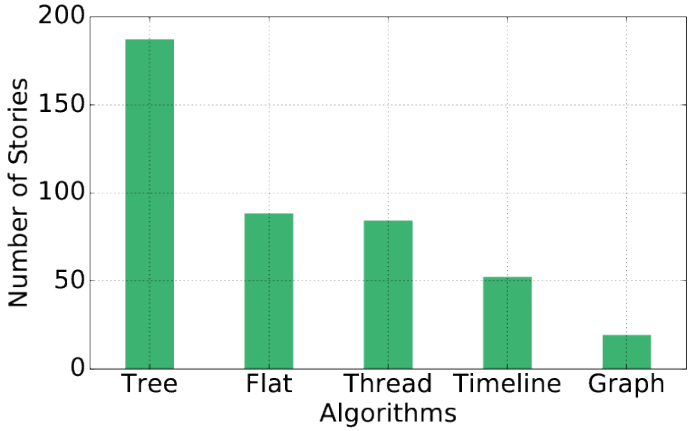
1. Do all the documents in each story cluster truly talk about the same story (*yes* or *no*)? Continue if *yes*.
2. Do all the documents in each event node truly talk about the same event (*yes* or *no*)? Continue if *yes*.



(a) Percentage of incorrect edges

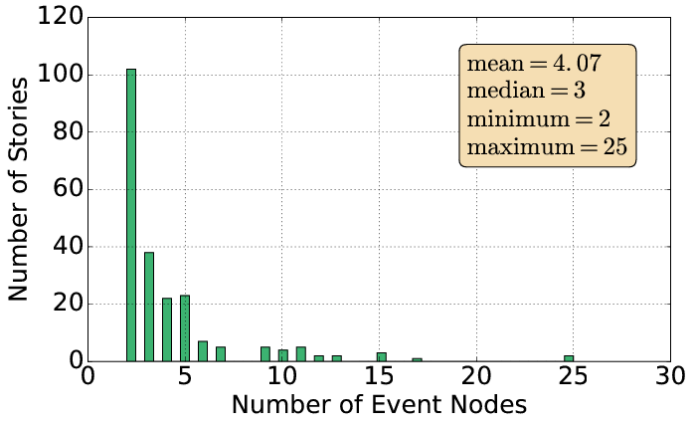


(b) Percentage of inconsistent paths

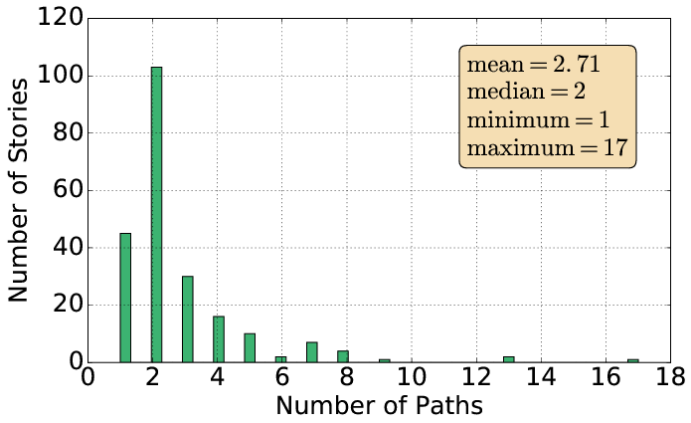


(c) Number of times rated as the most readable structure

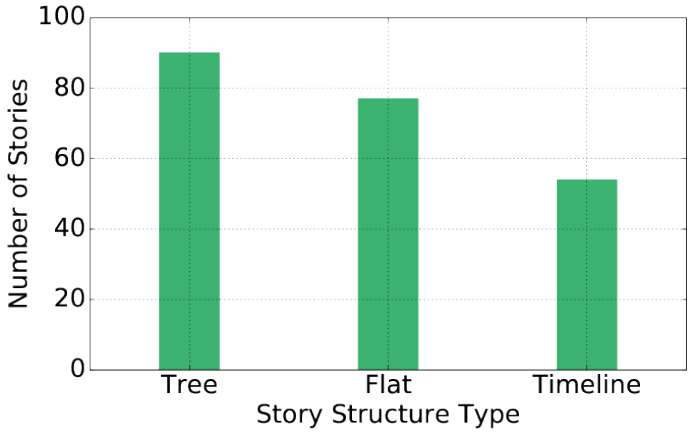
Figure 3.9: Comparing the performance of different story structure generation algorithms.



(a) Histogram of the number of events in each story



(b) Histogram of the number of paths in each story



(c) Numbers of different story structures

Figure 3.10: The characteristics of the story structures generated by the Story Forest system.

3. For each story structure given by different algorithms, how many edges correctly represent the event connections?

4. For each story structure given by story forest, event threading and story timeline, how many paths from ROOT to any leaf node exist in the graph? And how many such paths are logically coherent?

5. Which algorithm generates the structure that is the best in terms of revealing the story's underlying logical structure?

Note that for question (3), the total number of edges for each tree equals to the number of events in that tree. Therefore, to make a fair comparison, for the story graph algorithm, we only retain the n edges with the top scores, where n is the number of events in that story graph.

We first report the clustering effectiveness of our system in the pilot user evaluation on the 3-month dataset. Among the 261 stories, 234 of them are pure story clusters (*yes* to question 1), and furthermore there are 221 stories only contains pure event nodes (*yes* to question 2). Therefore, the final accuracy to extract events (*yes* to both question 1 and 2) is 84.7%.

Next, we compare the output story structures given by different algorithms from three aspects: the correct edges between events, the logical coherence of paths, and the overall readability of different story structures. Fig. 3.9(a) compares the CDFs of incorrect edge percentage under different algorithms. As we can see, Story Forest significantly outperforms the other 4 baseline approaches. As shown in Table 3.7, for 58% story trees, all the edges in each tree are reviewed as correct, and the average percentage of correct edges for all the story trees is 82.8%. In contrast, the average correct edge percentage given by the story graph algorithm is 32.9%.

An interesting observation is that the average percentage of correct edges given by the simple flat structure is 73.7%, which is a special case of our tree structures. This can be explained by the fact that most real-world breaking news that last for a constrained time period are not as complicated as a novel with rich logical structure, and a flat structure is often enough to depict their underlying logic. However, for stories with richer structures and a relatively longer timeline, Story Forest gives better result than other algorithms by comprehensively considering the event similarity, path coherence and time gap, while other algorithms only consider parts of all the factors.

For path coherence, Fig. 3.9(b) shows the CDFs of percentages of inconsistent paths under different algorithms. Story Forest gives significantly more coherent paths: the average percentage of coherent paths is 77.4% for our algorithm, and is 50.1% and

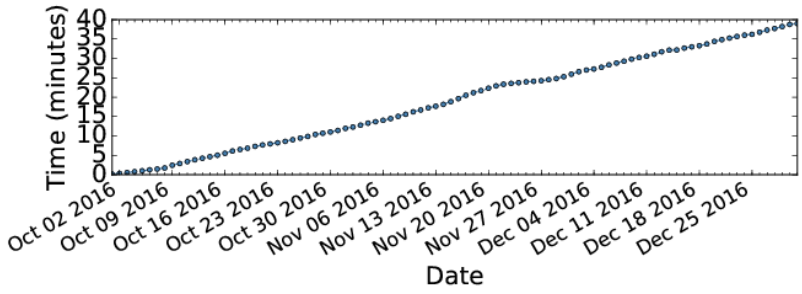


Figure 3.11: The running time of our system on the 3-month news dataset.

29.9%, respectively, for event threading and story timeline. Note that path coherence is meaningless for flat or graph structure.

Fig. 3.9(c) plots overall readability of different story structures. Among the 221 stories, the tree-based Story Forest system gives the best readability on 187 stories, which is much better than all other approaches. Different algorithms can generate the same structure. For example, the Story Forest system can also generate a flat structure, a timeline, or a same structure as the event threading algorithm does. Therefore, the sum of the numbers of best results given by different approaches is bigger than 221. It's worth noting that the flat and timeline algorithms also give 88 and 52 most readable results, which again indicates that the logic structures of a large portion of real-world news stories can be characterized by simple flat or timeline structures, which are special cases of story trees. And complex graphs are often an overkill.

We further inspect the story structures generated by Story Forest. Fig. 3.10(a) and Fig. 3.10(b) plot the distributions of the number of events and the number of paths in each story tree, respectively. The average numbers of events and paths are 4.07 and 2.71, respectively. Although the tree structure includes the flat and timeline structures as special cases, among the 221 stories, Story Forest generates 77 flat structures and 54 timelines, while the remaining 90 structures generated are still story trees. This implies that Story Forest is versatile and can generate diverse structures for real-world news stories, depending on the logical complexity of each story.

3.4.4 Algorithm Complexity and System Overhead

In this section, we discuss the complexity of each step in the Story Forest. For a time slot (e.g., in our case is one day), let N_d be the number of documents, N_w the number of unique words in corpora, note $N_w << N_d$, N_e the number of different events, N_s

the number of different stories, and N_k represents the maximum number of unique keywords in a document.

As discussed in [Sayyadi and Raschid, 2013], building keyword graph requires $\mathcal{O}(N_d N_k + N_w^2)$ complexity, and community detection based on betweenness centrality requires $\mathcal{O}(N_w^3)$. The complexity of assigning documents to keyword communities is $\mathcal{O}(N_d N_k N_e)$. So by far the total complexity is $\mathcal{O}(N_d N_k N_e + N_w^3)$. There exist other community detection algorithms requiring only $\mathcal{O}(N_w^2)$, such as the algorithm in [Radicchi *et al.*, 2004]. Thus we can further improve efficiency by using faster community detection algorithms.

After clustering documents by keyword communities, for each cluster the average number of documents is ${}^{N_d}/_{N_e}$. The pair-wise document relation classification is implemented in $\mathcal{O}(({}^{N_d}/_{N_e})^2)$. The complexity of the next document graph splitting operation is $\mathcal{O}(({}^{N_w}/_{N_e})^3)$. Therefore, the total complexity is $\mathcal{O}(N_e(({}^{N_d}/_{N_e})^2 + ({}^{N_w}/_{N_e})^3))$. Our experiments show that usually $1 \leq {}^{N_d}/_{N_e} \leq 100$. Combining with $N_w << N_d$, the complexity is now approximately $\mathcal{O}(N_e)$.

To grow story trees with new events, the complexity of finding the related story tree for each event is of $\mathcal{O}(N_s T)$, where T is the history length to keep existing stories and delete older stories. If no existing related story, creating a new story requires $\mathcal{O}(1)$ operations. Otherwise, the complexity of updating a story tree is $\mathcal{O}(T^{N_e}/_{N_s})$. In summary, the complexity of growing story trees is $\mathcal{O}(N_e T(N_s + {}^{N_e}/_{N_s})) \approx \mathcal{O}(T N_e N_s)$, as our experience on the Tencent news dataset shows that $1 \leq {}^{N_e}/_{N_s} \leq 200$. Our online algorithm to update story structure requires $\mathcal{O}({}^{N_e}/_{N_s})$ complexity and delivers a consistent story development structure, while most existing offline optimization based story structure algorithms require at least $\mathcal{O}(({}^{N_e}/_{N_s})^2)$ complexity and disrupt the previously generated story structures.

Fig. 3.11 shows the running time of our *Story Forest* system on the 3 months news dataset. The average time of processing each day's news is around 26 seconds, and increases linearly with number of days. For the offline keyword extraction module, the processing efficiency is approximately 50 documents per second. The performance of keyword extraction module is consistent with time and doesn't require frequently retraining. The LDA model is incrementally retrained every day to handle new words. For keyword extraction, the efficiency of event clustering and story structure generation can be further improved by a parallel implementation.

3.5 Concluding Remarks and Future Works

In this chapter, we describe our experience of implementing *Story Forest*, a news content organization system at Tencent, which is designed to discover events from vast streams of trending and breaking news and organize events in sensible story trees in an online manner. Our system is specifically tailored for fast processing massive amounts of breaking news data, whose story structures can most likely be captured by either a tree, a timeline or a flat structure. We proposed *EventX*, a semi-supervised text clustering algorithm designed to efficiently extract events from vast news corpora. *EventX* is a two-layered, graph-based document clustering scheme. In the first layer, we leveraged keyword co-occurrence graphs to cluster documents into topics, while in the second layer, we utilized a novel document relationship graph constructed through supervised learning to further split each topic into events, leading to an overall semi-supervised learning procedure. Our system further organizes the events into story trees with efficient online algorithms upon the arrival of daily news data.

We conducted extensive performance evaluation based on 60 GB of real-world (Chinese) news data, although our ideas are not language-dependent and can easily be extended to other languages, through detailed pilot user experience studies. To stimulate future research in event extraction, we also created a large Chinese News Events dataset that contains 11,748 long news articles collected from all major Internet news portals in China from October 01, 2016 to November 30, 2016, with ground truth event labels. We evaluated *EventX* on the Chinese News Events dataset, as well as the 20 Newsgroups English dataset. Extensive results suggest that our clustering procedure is significantly more effective at accurate event extraction than existing algorithms. For event extraction, the V-measure score and NMI of the clustering results on the Chinese News Events dataset are both 0.972, demonstrating the effectiveness of *EventX* to extract conceptually clean event clusters from vast and unstructured Internet news data. For story generation, 83% of the event links generated by Story Forest are logically correct as compared to an accuracy of 33% generated by more complex story graphs, demonstrating the ability of our system to organize trending news events into a logical structure that appeals to human readers.

In our future work, we will extend our proposed Story Forest and EventX framework to make it applicable to a continuous stream of news documents. Currently, we can use EventX in a batch-based manner: for example, we can extract the events for each batch of documents collected in different hours, and merge a new event with an existing event if they are talking about the same one. We can further improve our

framework by maintaining a dynamic keyword graph, performing dynamic keyword community detection, and run incremental event extraction based on such a dynamic keyword graph. Whenever new document stream comes in, we update the keyword graph to find out new keyword communities, and cluster new documents into new events or assign them to existing events. In this way, our framework can be directly applied to a continuous stream of news documents, rather than processing them in batches.

Chapter 4

Matching Article Pairs with Graphical Decomposition and Convolutions

The EventX algorithm we proposed in the Story Forest system requires us to classify the relationship between a pair of articles and construct a document graph for the second-layer clustering. Identifying the relationship between two articles, e.g., whether two articles published from different sources describe the same breaking news, is critical to many document understanding tasks. However, it is a challenging task. Existing approaches for modeling and matching sentence pairs do not perform well in matching *longer* documents, which embody more complex interactions between the enclosed entities than a sentence does.

To model article pairs, we propose the *Concept Interaction Graph* to represent an article as a graph of concepts. We then match a pair of articles by comparing the sentences that enclose the same concept vertex through a series of encoding techniques, and aggregate the matching signals through a graph convolutional network. To facilitate the evaluation of long article matching, we have created two datasets, each consisting of about 30K pairs of *breaking news* articles covering diverse topics in the open domain. Extensive evaluations of the proposed methods on the two datasets demonstrate significant improvements over a wide range of state-of-the-art methods for natural language matching.

4.1 Introduction

Identifying the relationship between a pair of articles is an essential natural language understanding task, which is critical to news systems and search engines. For example, a news system needs to cluster various articles on the Internet reporting

the same breaking news (probably in different ways of wording and narratives), remove redundancy and form storylines [Shahaf *et al.*, 2013; Liu *et al.*, 2017; Zhou *et al.*, 2015; Vossen *et al.*, 2015; Bruggermann *et al.*, 2016]. The rich semantic and logic structures in longer documents have made it a different and more challenging task to match a pair of articles than to match a pair of sentences or a query-document pair in information retrieval.

Traditional term-based matching approaches estimate the semantic distance between a pair of text objects via unsupervised metrics, e.g., via TF-IDF vectors, BM25 [Robertson *et al.*, 2009], LDA [Blei *et al.*, 2003] and so forth. These methods have achieved success in query-document matching, information retrieval and search. In recent years, a wide variety of deep neural network models have also been proposed for text matching [Hu *et al.*, 2014; Qiu and Huang, 2015; Wan *et al.*, 2016; Pang *et al.*, 2016], which can capture the semantic dependencies (especially sequential dependencies) in natural language through layers of recurrent or convolutional neural networks. However, existing deep models are mainly designed for matching sentence pairs, e.g., for paraphrase identification, answer selection in question-answering, omitting the complex interactions among keywords, entities or sentences that are present in a longer article. Therefore, article pair matching remains under-explored in spite of its importance.

In this chapter, we apply the divide-and-conquer philosophy to matching a pair of articles and bring deep text understanding from the currently dominating sequential modeling of language elements to a new level of graphical document representation, which is more suitable for longer articles. Specifically, we have made the following contributions:

First, we propose the so-called *Concept Interaction Graph* (CIG) to represent a document as a weighted graph of concepts, where each concept vertex is either a keyword or a set of tightly connected keywords. The sentences in the article associated with each concept serve as the features for local comparison to the same concept appearing in another article. Furthermore, two concept vertices in an article are also connected by a weighted edge which indicates their interaction strength. The CIG does not only capture the essential semantic units in a document but also offers a way to perform anchored comparison between two articles along the common concepts found.

Second, we propose a divide-and-conquer framework to match a pair of articles based on the constructed CIGs and graph convolutional networks (GCNs). The idea is that for each concept vertex that appears in both articles, we first obtain the local matching vectors through a range of text pair encoding schemes, including both

neural encoding and term-based encoding. We then aggregate the local matching vectors into the final matching result through graph convolutional layers [Kipf and Welling, 2016; Defferrard *et al.*, 2016]. In contrast to RNN-based sequential modeling, our model factorizes the matching process into local matching sub-problems on a graph, each focusing on a different concept, and by using GCN layers, generates matching results based on a holistic view of the entire graph.

Although there exist many datasets for sentence matching, the semantic matching between longer articles is a largely unexplored area. To the best of our knowledge, to date, there does not exist a labeled public dataset for long document matching. To facilitate evaluation and further research on document and especially news article matching, we have created *two labeled datasets*[1], one annotating whether two news articles found on Internet (from different media sources) report the same breaking news event, while the other annotating whether they belong to the same news story (yet not necessarily reporting the same breaking news event). These articles were collected from major Internet news providers in China, including Tencent, Sina, WeChat, Sohu, etc., covering diverse topics, and were labeled by professional editors. Note that similar to most other natural language matching models, all the approaches proposed in this chapter can easily work on other languages as well.

Through extensive experiments, we show that our proposed algorithms have achieved significant improvements on matching news article pairs, as compared to a wide range of state-of-the-art methods, including both term-based and deep text matching algorithms. With the same encoding or term-based feature representation of a pair of articles, our approach based on graphical decomposition and convolutions can improve the classification accuracy by 17.31% and 23.09% on the two datasets, respectively.

4.2 Concept Interaction Graph

In this section, we present our *Concept Interaction Graph* (CIG) to represent a document as an *undirected weighted graph*, which decomposes a document into subsets of sentences, each subset focusing on a different *concept*. Given a document $\mathcal{D}$, a CIG is a graph G_D, where each vertex in G_D is called a *concept*, which is a keyword or a set of highly correlated keywords in document $\mathcal{D}$. Each sentence in $\mathcal{D}$ will be attached to the *single* concept vertex that it is the most related to, which most frequently is the concept the sentence mentions. Hence, vertices will have their own sentence sets, which are disjoint. The weight of the edge between a pair of concepts denotes how much the two concepts are related to each other and can be determined in various

[1]Our code and datasets are available at: https://github.com/BangLiu/ArticlePairMatching

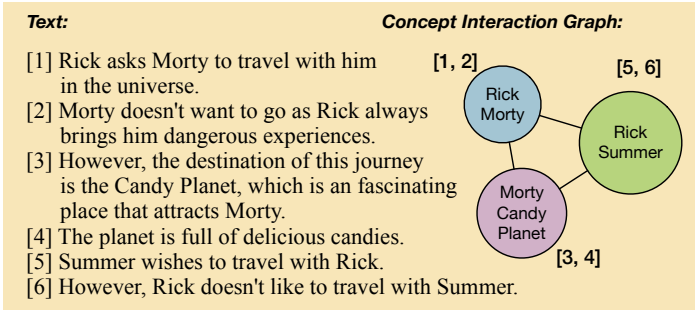


Figure 4.1: An example to show a piece of text and its Concept Interaction Graph representation.

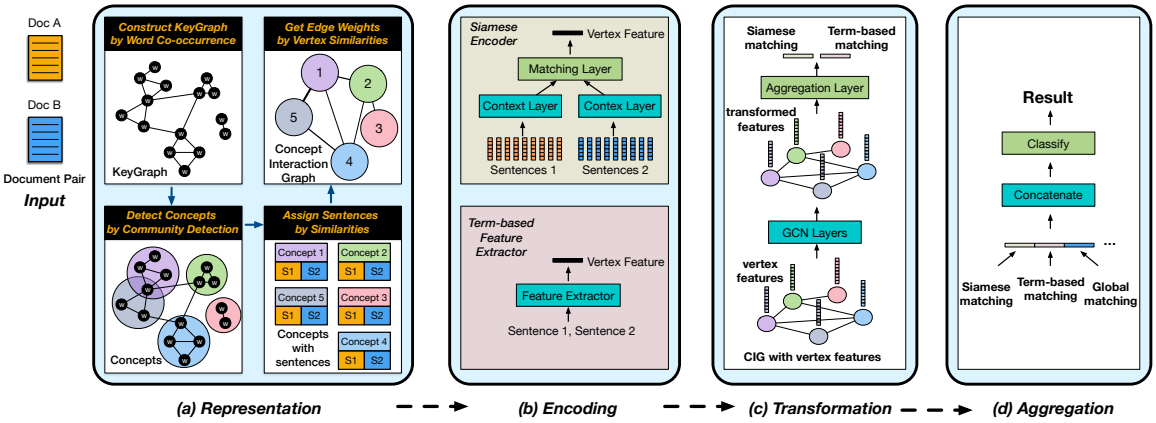


Figure 4.2: An overview of our approach for constructing the *Concept Interaction Graph* (CIG) from a pair of documents and classifying it by Graph Convolutional Networks.

ways.

As an example, Fig. 4.1 illustrates how we convert a document into a Concept Interaction Graph. We can extract keywords *Rick*, *Morty*, *Summer*, and *Candy Planet* from the document using standard keyword extraction algorithms, e.g., TextRank [Mihalcea and Tarau, 2004]. These keywords are further clustered into three concepts, where each concept is a subset of highly correlated keywords. After grouping keywords into concepts, we attach each sentence in the document to its most related concept vertex. For example, in Fig. 4.1, sentences 1 and 2 are mainly talking about the relationship between *Rick* and *Morty*, and are thus attached to the concept (*Rick*, *Morty*). Other sentences are attached to vertices in a similar way. The attachment of sentences to concepts naturally dissects the original document into multiple disjoint sentence subsets. As a result, we have represented the original document with a graph of key concepts, each with a sentence subset, as well as the interaction topology among them.

Fig 4.2 (a) illustrates the construction of CIGs for a pair of documents aligned by the discovered concepts. Here we first describe the detailed steps to construct a CIG for a single document:

KeyGraph Construction. Given a document $\mathcal{D}$, we first extract the named entities and keywords by TextRank [Mihalcea and Tarau, 2004]. After that, we construct a keyword co-occurrence graph, called *KeyGraph*, based on the set of found keywords. Each keyword is a vertex in the KeyGraph. We connect two keywords by an edge if they co-occur in a same sentence.

We can further improve our model by performing co-reference resolution and synonym analysis to merge keywords with the same meaning. However, we do not apply these operations due to the time complexity.

Concept Detection (Optional). The structure of KeyGraph reveals the connections between keywords. If a subset of keywords are highly correlated, they will form a densely connected sub-graph in the KeyGraph, which we call a *concept*. Concepts can be extracted by applying community detection algorithms on the constructed KeyGraph. Community detection is able to split a KeyGraph G_{key} into a set of communities $C = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_{|C|}\}$, where each community $\mathcal{C}_i$ contains the keywords for a certain concept. By using overlapping community detection, each keyword may appear in multiple concepts. As the number of concepts in different documents varies a lot, we utilize the *betweenness centrality score* based algorithm [Sayyadi and Raschid, 2013] to detect keyword communities in KeyGraph.

Note that this step is optional, i.e., we can also use each keyword directly as a concept. The benefit brought by concept detection is that it reduces the number of vertices in a graph and speeds up matching, as will be shown in Sec. 4.4.

Sentence Attachment. After the concepts are discovered, the next step is to group sentences by concepts. We calculate the cosine similarity between each sentence and each concept, where sentences and concepts are represented by TF-IDF vectors. We assign each sentence to the concept which is the most similar to the sentence. Sentences that do not match any concepts in the document will be attached to a *dummy vertex* that does not contain any keywords.

Edge Construction. To construct edges that reveal the correlations between different concepts, for each vertex, we represent its sentence set as a concatenation of the sentences attached to it, and calculate the edge weight between any two vertices as the TF-IDF similarity between their sentence sets. Although edge weights may be decided in other ways, our experience shows that constructing edges by TF-IDF similarity generates a CIG that is more densely connected.

When performing article pair matching, the above steps will be applied to a pair

of documents $\mathcal{D}_A$ and $\mathcal{D}_B$, as is shown in Fig. 4.2 (a). The only additional step is that we align the CIGs of the two articles by the concept vertices, and for each common concept vertex, merge the sentence sets from $\mathcal{D}_A$ and $\mathcal{D}_B$ for local comparison.

4.3 Article Pair Matching through Graph Convolutions

Given the merged CIG G_{AB} of two documents $\mathcal{D}_A$ and $\mathcal{D}_B$ described in Sec. 4.2, we match a pair of articles in a "divide-and-conquer" manner by matching the sentence sets from $\mathcal{D}_A$ and $\mathcal{D}_B$ associated with each concept and aggregating local matching results into a final result through multiple graph convolutional layers. Our approach overcomes the limitation of previous text matching algorithms, by extending text representation from a sequential (or grid) point of view to a graphical view, and can therefore better capture the rich semantic interactions in longer text.

Fig. 4.2 illustrates the overall architecture of our proposed method, which consists of four steps: a) representing a pair of documents by a single merged CIG, b) learning multi-viewed matching features for each concept vertex, c) structurally transforming local matching features by graph convolutional layers, and d) aggregating local matching features to get the final result. Steps (b)-(d) can be trained end-to-end.

Encoding Local Matching Vectors. Given the merged CIG G_{AB}, our first step is to learn an appropriate *matching vector* of a fixed length for each individual concept $v \in G_{AB}$ to express the semantic similarity between $\mathcal{S}_A(v)$ and $\mathcal{S}_B(v)$, the sentence sets of concept v from documents $\mathcal{D}_A$ and $\mathcal{D}_B$, respectively. This way, the matching of two documents is converted to match the pair of sentence sets on each vertex of G_{AB}. Specifically, we generate local matching vectors based on both neural networks and term-based techniques.

Siamese Encoder: we apply a Siamese neural network encoder [Neculoiu *et al.*, 2016] onto each vertex $v \in G_{AB}$ to convert the word embeddings [Mikolov *et al.*, 2013] of $\{\mathcal{S}_A(v), \mathcal{S}_B(v)\}$ into a fixed-sized hidden feature vector $\mathbf{m}_{AB}(v)$, which we call the *match vector*.

We use a Siamese structure to take $\mathcal{S}_A(v)$ and $\mathcal{S}_B(v)\}$ (which are two sequences of word embeddings) as inputs, and encode them into two context vectors through the context layers that share the same weights, as shown in Fig. 4.2 (b). The context layer usually contains one or multiple bi-directional LSTM (BiLSTM) or CNN layers with max pooling layers, aiming to capture the contextual information in $\mathcal{S}_A(v)$ and $\mathcal{S}_B(v)\}$.

Let $\mathbf{c}_A(v)$ and $\mathbf{c}_B(v)$ denote the context vectors obtained for $\mathcal{S}_A(v)$ and $\mathcal{S}_B(v)$, respectively. Then, the matching vector $\mathbf{m}_{AB}(v)$ for vertex v is given by the subsequent aggregation layer, which concatenates the element-wise absolute difference and the element-wise multiplication of the two context vectors, i.e.,

$$\mathbf{m}_{AB}(v) = (|\mathbf{c}_A(v) - \mathbf{c}_B(v)|, \mathbf{c}_A(v) \circ \mathbf{c}_B(v)), \tag{4.1}$$

where $\circ$ denotes Hadamard product.

Term-based Similarities: we also generate another matching vector for each v by directly calculating term-based similarities between $\mathcal{S}_A(v)$ and $\mathcal{S}_B(v)$, based on 5 metrics: the TF-IDF cosine similarity, TF cosine similarity, BM25 cosine similarity, Jaccard similarity of 1-gram, and Ochiai similarity measure. These similarity scores are concatenated into another matching vector $\mathbf{m}'_{AB}(v)$ for v, as shown in Fig. 4.2 (b).

Matching Aggregation via GCN The local matching vectors must be aggregated into a final matching score for the pair of articles. We propose to utilize the ability of the Graph Convolutional Network (GCN) filters [Kipf and Welling, 2016] to capture the patterns exhibited in the CIG G_{AB} at multiple scales. In general, the input to the GCN is a graph $G = (\mathcal{V}, E)$ with N vertices $v_i \in \mathcal{V}$, and edges $e_{ij} = (v_i, v_j) \in E$ with weights w_{ij}. The input also contains a vertex feature matrix denoted by $X = \{\mathbf{x}_i\}_{i=1}^N$, where $\mathbf{x}_i$ is the *feature vector* of vertex v_i. For a pair of documents $\mathcal{D}_A$ and $\mathcal{D}_B$, we input their CIG G_{AB} (with N vertices) with a (concatenated) matching vector on each vertex into the GCN, such that the feature vector of vertex v_i in GCN is given by

$$\mathbf{x}_i = (\mathbf{m}_{AB}(v_i), \mathbf{m}'_{AB}(v_i)).$$

Now let us briefly describe the GCN layers [Kipf and Welling, 2016] used in Fig. 4.2 (c). Denote the weighted adjacency matrix of the graph as $A \in \mathbb{R}^{N \times N}$ where $A_{ij} = w_{ij}$ (in CIG, it is the TF-IDF similarity between vertex i and j). Let D be a diagonal matrix such that $D_{ii} = \sum_j A_{ij}$. The input layer to the GCN is $H^{(0)} = X$, which contains the original vertex features. Let $H^{(l)} \in \mathbb{R}^{N \times M_l}$ denote the matrix of *hidden representations* of the vertices in the l^{th} layer. Then each GCN layer applies the following graph convolutional filter onto the previous hidden representations:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}), \tag{4.2}$$

where $\tilde{A} = A + I_N$, I_N is the identity matrix, and $\tilde{D}$ is a diagonal matrix such that $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. They are the adjacency matrix and the degree matrix of graph G, respectively.

$W^{(l)}$ is the trainable weight matrix in the l^{th} layer. $\sigma(\cdot)$ denotes an activation function such as sigmoid or ReLU function. Such a graph convolutional rule is motivated by the first-order approximation of localized spectral filters on graphs [Kipf and Welling, 2016] and when applied recursively, can extract interaction patterns among vertices.

Finally, the hidden representations in the final GCN layer is merged into a single vector (called a graphically merged matching vector) of a fixed length, denoted by $\mathbf{m}_{AB}$, by taking the mean of the hidden vectors of all vertices in the last layer. The final matching score will be computed based on $\mathbf{m}_{AB}$, through a classification network, e.g., a multi-layered perceptron (MLP).

In addition to the graphically merged matching vector $\mathbf{m}_{AB}$ described above, we may also append other global matching features to $\mathbf{m}_{AB}$ to expand the feature set. These additional global features can be calculated, e.g., by encoding two documents directly with state-of-the-art language models like BERT [Devlin *et al.*, 2018] or by directly computing their term-based similarities. However, we show in Sec. 4.4 that such global features can hardly bring any more benefit to our scheme, as the graphically merged matching vectors are already sufficiently expressive in our problem.

4.4 Evaluation

Tasks. We evaluate the proposed approach on the task of ***identifying whether a pair of news articles report the same breaking news (or event) and whether they belong to the same series of news story***, which is motivated by a real-world news app. In fact, the proposed article pair matching schemes have been deployed in the anonymous news app for news clustering, with more than 110 millions of daily active users.

Note that traditional methods to document clustering include unsupervised text clustering and text classification into predefined topics. However, a number of breaking news articles emerge on the Internet everyday with their topics/themes unknown, so it is not possible to predefine their topics. Thus, supervised text classification cannot be used here. It is even impossible to determine how many news clusters there exist. Therefore, the task of classifying whether two news articles are reporting the same breaking news event or belong to the same story is critical to news apps and search engines for clustering, redundancy removal and topic summarization.

In our task, an "event" refers to a piece of breaking news on which multiple media sources may publish articles with different narratives and wording. Furthermore, a "story" consists of a series of logically related breaking news events. It is worth noting

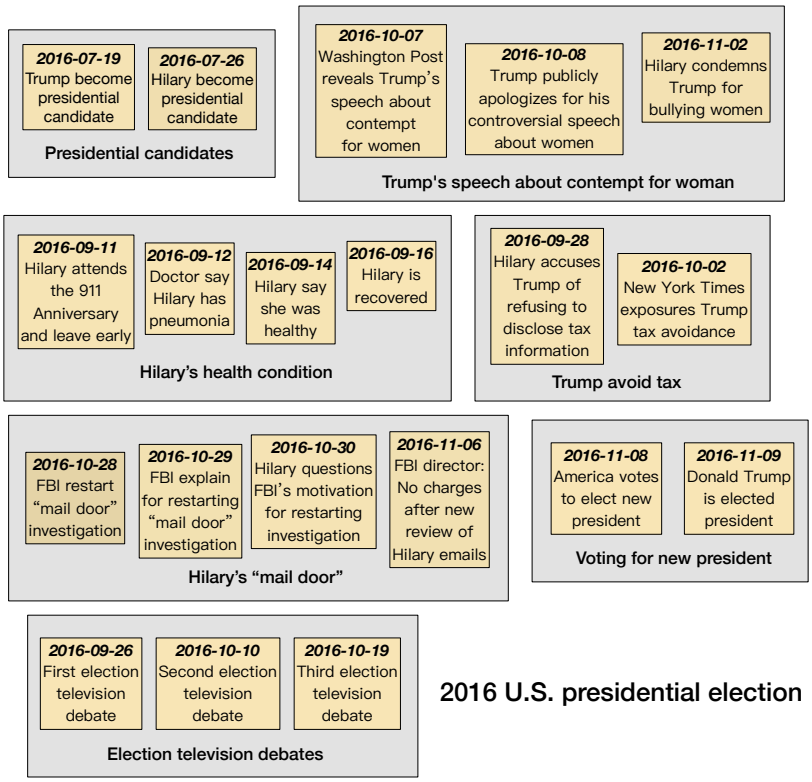


Figure 4.3: The events contained in the story "2016 U.S. presidential election".

that our objective is fundamentally different from the traditional event coreference literature, e.g., [Bejan and Harabagiu, 2010; Lee *et al.*, 2013; Lee *et al.*, 2012] or SemEval-2018 Task 5 (Counting Events) [Postma *et al.*, 2018], where the task is to detect all the events (or in fact, "actions" like shooting, car crashes) a document mentions.

In contrast, although a news article may mention multiple entities and even previous physical events, the "event" in our dataset always refers to the breaking news that the article intends to report or the incident that triggers the media's coverage. And our task is to identify whether two articles intend to report the same breaking news. For example, two articles "University of California system libraries break off negotiations with Elsevier, will no longer order their journals" and "University of California Boycotts Publishing Giant Elsevier" from two different sources are apparently intended to report the same breaking news event of UC dropping subscription to Elsevier, although other actions may be peripherally mentioned in these articles, e.g., "eight months of unsuccessful negotiations." In addition, we do not attempt to perform reading comprehension question answering tasks either, e.g., finding out how many killing incidents or car crashes there are in a year (SemEval-2018 Task 5 [Postma *et al.*, 2018]).

Dataset	Pos Samples	Neg Samples	Train	Dev	Test
CNSE	12865	16198	17438	5813	5812
CNSS	16887	16616	20102	6701	6700

Table 4.1: Description of evaluation datasets.

As a typical example, Fig. 4.3 shows the events contained in the story *2016 U.S. presidential election*, where each tag shows a breaking news event possibly reported by multiple articles with different narratives (articles not shown here). We group highly coherent events together. For example, there are multiple events about Election television debates. One of our objectives is to identify whether two news articles report the same event, e.g., a yes when they are both reporting *Trump and Hilary's first television debate*, though with different wording, or a no, when one article is reporting *Trump and Hilary's second television debate* while the other is talking about *Donald Trump is elected president*.

Datasets. To the best of our knowledge, there is no publicly available dataset for long document matching tasks. We created two datasets: the Chinese News Same Event dataset (CNSE) and Chinese News Same Story dataset (CNSS), which are labeled by professional editors. They contain long Chinese news articles collected from major Internet news providers in China, covering diverse topics in the open domain. The CNSE dataset contains $29,063$ pairs of news articles with labels representing whether a pair of news articles are reporting about the same breaking news event. Similarly, the CNSS dataset contains $33,503$ pairs of articles with labels representing whether two documents fall into the same news story. The average number of words for all documents in the datasets is 734 and the maximum value is 21791.

In our datasets, we only labeled the *major* event (or story) that a news article is reporting, since in the real world, each breaking news article on the Internet must be intended to report some specific breaking news that has just happened to attract clicks and views. Our objective is to determine whether two news articles intend to report the same breaking news.

Note that the negative samples in the two datasets are not randomly generated: we select document pairs that contain similar keywords, and exclude samples with TF-IDF similarity below a certain threshold. The datasets have been made publicly available for research purpose.

Table 4.1 shows a detailed breakdown of the two datasets. For both datasets, we use 60% of all the samples as the training set, 20% as the development (validation)

Baselines	CNSE		CNSS		Our models	CNSE		CNSS	
	Acc	F1	Acc	F1		Acc	F1	Acc	F1
I. ARC-I	53.84	48.68	50.10	66.58	**XI**. CIG-Siam	74.47	73.03	75.32	78.58
II. ARC-II	54.37	36.77	52.00	53.83	**XII**. CIG-Siam-GCN	**74.58**	**73.69**	**78.91**	**80.72**
III. DUET	55.63	51.94	52.33	60.67	**XIII**. CIG_{cd}-Siam-GCN	73.25	73.10	76.23	76.94
IV. DSSM	58.08	64.68	61.09	70.58	**XIV**. CIG-Sim	72.58	71.91	75.16	77.27
V. C-DSSM	60.17	48.57	52.96	56.75	**XV**. CIG-Sim-GCN	**83.35**	**80.96**	**87.12**	**87.57**
VI. MatchPyramid	66.36	54.01	62.52	64.56	**XVI**. CIG_{cd}-Sim-GCN	81.33	78.88	86.67	87.00
VII. BM25	69.63	66.60	67.77	70.40	**XVII**. CIG-Sim&Siam-GCN	84.64	**82.75**	89.77	90.07
VIII. LDA	63.81	62.44	62.98	69.11	**XVIII**. CIG-Sim&Siam-GCN-Sim^g	84.21	82.46	**90.03**	**90.29**
IX. SimNet	71.05	69.26	70.78	74.50	**XIX**. CIG-Sim&Siam-GCN-BERT^g	**84.68**	82.60	89.56	89.97
X. BERT fine-tuning	81.30	79.20	86.64	87.08	**XX**. CIG-Sim&Siam-GCN-Sim^g&BERT^g	84.61	82.59	89.47	89.71

Table 4.2: Accuracy and F1-score results of different algorithms on CNSE and CNSS datasets.

set, and the remaining 20% as the test set. We carefully ensure that different splits do not contain any overlaps to avoid data leakage. The metrics used for performance evaluation are the accuracy and F1 scores of binary classification results. For each evaluated method, we perform training for 10 epochs and then choose the epoch with the best validation performance to be evaluated on the test set.

Baselines. We test the following baselines:

- *Matching by representation-focused or interaction-focused deep neural network models*: DSSM [Huang *et al.*, 2013], C-DSSM [Shen *et al.*, 2014], DUET [Mitra *et al.*, 2017], MatchPyramid [Pang *et al.*, 2016], ARC-I [Hu *et al.*, 2014], ARC-II [Hu *et al.*, 2014]. We use the implementations from MatchZoo [Fan *et al.*, 2017] for the evaluation of these models.

- *Matching by term-based similarities*: BM25 [Robertson *et al.*, 2009], LDA [Blei *et al.*, 2003] and SimNet (which is extracting the five text-pair similarities mentioned in Sec. 4.3 and classifying by a multi-layer feedforward neural network).

- *Matching by a large-scale pre-training language model*: BERT [Devlin *et al.*, 2018].

Note that we focus on the capability of long text matching. Therefore, we do not use any short text information, such as titles, in our approach or in any baselines. In fact, the “relationship" between two documents is not limited to "whether the same event or not". Our algorithm is able to identify a general relationship between documents, e.g., whether two episodes are from the same season of a TV series. The definition of the relationship (e.g., same event/story, same chapter of a book) is solely

defined and supervised by the labeled training data. For these tasks, the availability of other information such as titles can not be assumed.

As shown in Table 4.2, we evaluate different variants of our own model to show the effect of different sub-modules. In model names, "CIG" means that in CIG, we directly use keywords as concepts without community detection, whereas "CIG$_{cd}$" means that each concept vertex in the CIG contains a set of keywords grouped via community detection. To generate the matching vector on each vertex, "Siam" indicates the use of Siamese encoder, while "Sim" indicates the use of term-based similarity encoder, as shown in Fig. 4.2. "GCN" means that we convolve the local matching vectors on vertices through GCN layers. Finally, "BERT^g" or "Sim^g" indicates the use of additional global features given by BERT or the five term-based similarity metrics mentioned in Sec. 4.3, appended to the graphically merged matching vector $\mathbf{m}_{AB}$, for final classification.

Implementation Details. We use Stanford CoreNLP [Manning *et al.*, 2014] for word segmentation (on Chinese text) and named entity recognition. For Concept Interaction Graph construction with community detection, we set the minimum community size (number of keywords contained in a concept vertex) to be 2, and the maximum size to be 6.

Our neural network model consists of word embedding layer, Siamese encoding layer, Graph transformation layers, and classification layer. For embedding, we load the pre-trained word vectors and fix it during training. The embeddings of out of vocabulary words are set to be zero vectors. For the Siamese encoding network, we use 1-D convolution with number of filters 32, followed by an ReLU layer and Max Pooling layer. For graph transformation, we utilize 2 layers of GCN [Kipf and Welling, 2016] for experiments on the CNSS dataset, and 3 layers of GCN for experiments on the CNSE dataset. When the vertex encoder is the five-dimensional features, we set the output size of GCN layers to be 16. When the vertex encoder is the Siamese network encoder, we set the output size of GCN layers to be 128 except the last layer. For the last GCN layer, the output size is always set to be 16. For the classification module, it consists of a linear layer with output size 16, an ReLU layer, a second linear layer, and finally a Sigmoid layer. Note that this classification module is also used for the baseline method SimNet.

As we mentioned in Sec. 4.1, our code and datasets have been open sourced. We implement our model using PyTorch 1.0 [Paszke *et al.*, 2017]. The experiments without BERT are carried out on an MacBook Pro with a 2 GHz Intel Core i7 processor and 8 GB memory. We use L2 weight decay on all the trainable variables, with parameter $\lambda = 3 \times 10^{-7}$. The dropout rate between every two layers is 0.1.

We apply gradient clipping with maximum gradient norm 5.0. We use the ADAM optimizer [Kingma and Ba, 2014] with $\beta_1 = 0.8,\ \beta_2 = 0.999,\ \epsilon = 10^{-8}$. We use a learning rate warm-up scheme with an inverse exponential increase from 0.0 to 0.001 in the first 1000 steps, and then maintain a constant learning rate for the remainder of training. For all the experiments, we set the maximum number of training epochs to be 10.

4.4.1 Results and Analysis

Table 4.2 summarizes the performance of all the compared methods on both datasets. Our model achieves the best performance on both two datasets and significantly outperforms all other methods. This can be attributed to two reasons. First, as the input of article pairs are re-organized into Concept Interaction Graphs, the two documents are aligned along the corresponding semantic units for easier concept-wise comparison. Second, our model encodes local comparisons around different semantic units into local matching vectors, and aggregate them via graph convolutions, taking semantic topologies into consideration. Therefore, it solves the problem of matching documents via divide-and-conquer, which is suitable for handling long text.

Impact of Graphical Decomposition. Comparing method XI with methods I-VI in Table 4.2, they all use the same word vectors and use neural networks for text encoding. The key difference is that our method XI compares a pair of articles over a CIG in per-vertex decomposed fashion. We can see that the performance of method XI is significantly better than methods I-VI. Similarly, comparing our method XIV with methods VII-IX, they all use the same term-based similarities. However, our method achieves significantly better performance by using graphical decomposition. Therefore, we conclude that graphical decomposition can greatly improve long text matching performance.

Note that the deep text matching models I-VI lead to bad performance, because they were invented mainly for sequence matching and can hardly capture meaningful semantic interactions in article pairs. When the text is long, it is hard to get an appropriate context vector representation for matching. For interaction-focused neural network models, most of the interactions between words in two long articles will be meaningless.

Impact of Graph Convolutions. Compare methods XII and XI, and compare methods XV and XIV. We can see that incorporating GCN layers has significantly improved the performance on both datasets. Each GCN layer updates the hidden vector of each vertex by integrating the vectors from its neighboring vertices. Thus,

the GCN layers learn to graphically aggregate local matching features into a final result.

Impact of Community Detection. By comparing methods XIII and XII, and comparing methods XVI and XV, we observe that using community detection, such that each concept is a set of correlated keywords instead of a single keyword, leads to slightly worse performance. This is reasonable, as using each keyword directly as a concept vertex provides more anchor points for article comparison . However, community detection can group highly coherent keywords together and reduces the average size of CIGs from 30 to 13 vertices. This helps to reduce the total training and testing time of our models by as much as 55%. Therefore, one may choose whether to apply community detection to trade accuracy off for speedups.

Impact of Multi-viewed Matching. Comparing methods XVII and XV, we can see that the concatenation of different graphical matching vectors (both term-based and Siamese encoded features) can further improve performance. This demonstrates the advantage of combining multi-viewed matching vectors.

Impact of Added Global Features. Comparing methods XVIII, XIX, XX with method XVII, we can see that adding more global features, such as global similarities (Sim^g) and/or global BERT encodings (BERT^g) of the article pair, can hardly improve performance any further. This shows that graphical decomposition and convolutions are the main factors that contribute to the performance improvement. Since they already learn to aggregate local comparisons into a global semantic relationship, additionally engineered global features cannot help.

Model Size and Parameter Sensitivity: Our biggest model without BERT is XVIII, which contains only $\sim$34K parameters. In comparison, BERT contains 110M-340M parameters. However, our model significantly outperforms BERT.

We tested the sensitivity of different parameters in our model. We found that 2 to 3 layers of GCN layers gives the best performance. Further introducing more GCN layers does not improve the performance, while the performance is much worse with zero or only one GCN layer. Furthermore, in GCN hidden representations of a size between 16 and 128 yield good performance. Further increasing this size does not show obvious improvement.

For the optional community detection step in CIG construction, we need to choose the minimum size and the maximum size of communities. We found that the final performance remains similar if we vary the minimum size from 2$\sim$3 and the maximum size from 6$\sim$10. This indicates that our model is robust and insensitive to these parameters.

Time complexity. For keywords of news articles, in real-world industry applica-

tions, they are usually extracted in advance by highly efficient off-the-shelf tools and pre-defined vocabulary. For CIG construction, let N_s be the number of sentences in two documents, N_w be the number of unique words in documents, and N_k represents the number of unique keywords in a document. Building keyword graph requires $\mathcal{O}(N_s N_k + N_w^2)$ complexity [Sayyadi and Raschid, 2013], and betweenness-based community detection requires $\mathcal{O}(N_k^3)$. The complexity of sentence assignment and weight calculation is $\mathcal{O}(N_s N_k + N_k^2)$. For graph classification, our model size is not big and can process document pairs efficiently.

4.5 Conclusion

We propose the *Concept Interaction Graph* to organize documents into a graph of concepts, and introduce a divide-and-conquer approach to matching a pair of articles based on graphical decomposition and convolutional aggregation. We created two new datasets for long document matching with the help of professional editors, consisting of about 60K pairs of news articles, on which we have performed extensive evaluations. In the experiments, our proposed approaches significantly outperformed an extensive range of state-of-the-art schemes, including both term-based and deep-model-based text matching algorithms. Results suggest that the proposed graphical decomposition and the structural transformation by GCN layers are critical to the performance improvement in matching article pairs.

Chapter 5

Matching Natural Language Sentences with Hierarchical Sentence Factorization

In the previous chapter, our concept interaction graph turns the problem of long document matching into short sentence matching over different vertices. Such a decomposition helps us to overcome the difficulty brought by the length of documents. However, how to estimate the semantic relatedness between short sentence pairs remains a challenging problem. Semantic matching of natural language sentences or identifying the relationship between two sentences is also a core research problem underlying many natural language tasks. Depending on whether training data is available, prior research has proposed both unsupervised distance-based schemes and supervised deep learning schemes for sentence matching. However, previous approaches either omit or fail to fully utilize the ordered, hierarchical, and flexible structures of language objects, as well as the interactions between them.

In this chapter, we propose *Hierarchical Sentence Factorization*—a technique to factorize a sentence into a hierarchical representation, with the components at each different scale reordered into a "predicate-argument" form. The proposed sentence factorization technique leads to the invention of: 1) a new unsupervised distance metric which calculates the semantic distance between a pair of text snippets by solving a penalized optimal transport problem while preserving the logical relationship of words in the reordered sentences, and 2) new multi-scale deep learning models for supervised semantic training, based on factorized sentence hierarchies. We apply our techniques to text-pair similarity estimation and text-pair relationship classification tasks, based on multiple datasets such as STSbenchmark, the Microsoft Research paraphrase identification (MSRP) dataset, the SICK dataset, etc. Extensive experiments show that the proposed hierarchical sentence factorization can be used to significantly improve

the performance of existing unsupervised distance-based metrics as well as multiple supervised deep learning models based on the convolutional neural network (CNN) and long short-term memory (LSTM).

5.1 Introduction

Semantic matching, which aims to model the underlying semantic similarity or dissimilarity among different textual elements such as sentences and documents, has been playing a central role in many Natural Language Processing (NLP) applications, including information extraction [Grishman, 1997], top-k re-ranking in machine translation [Brown *et al.*, 1993], question-answering [Yu *et al.*, 2014], automatic text summarization [Ponzanelli *et al.*, 2015]. However, semantic matching based on either supervised or unsupervised learning remains a hard problem. Natural language demonstrates complicated hierarchical structures, where different words can be organized in different orders to express the same idea. As a result, appropriate semantic representation of text plays a critical role in matching natural language sentences.

Traditional approaches represent text objects as bag-of-words (BoW), term frequency inverse document frequency (TF-IDF) [Wu *et al.*, 2008] vectors, or their enhanced variants [Paltoglou and Thelwall, 2010; Robertson and Walker, 1994]. However, such representations can not accurately capture the similarity between individual words, and do not take the semantic structure of language into consideration. Alternatively, word embedding models, such as *word2vec* [Mikolov *et al.*, 2013] and *Glove* [Pennington *et al.*, 2014], learn a distributional semantic representation of each word and have been widely used.

Based on the word-vector representation, a number of unsupervised and supervised matching schemes have been recently proposed. As an unsupervised learning approach, the Word Mover's Distance (WMD) metric [Kusner *et al.*, 2015] measures the dissimilarity between two sentences (or documents) as the minimum distance to transport the embedded words of one sentence to those of another sentence. However, the sequential and structural nature of sentences is omitted in WMD. For example, two sentences containing exactly the same words in different orders can express totally different meanings. On the other hand, many supervised learning schemes based on deep neural networks have also been proposed for sentence matching [Mueller and Thyagarajan, 2016; Severyn and Moschitti, 2015; Wang *et al.*, 2017*b*; Pang *et al.*, 2016]. A common characteristic of many of these neural network models is that they adopt a Siamese architecture, taking the word embedding sequences of a pair of sentences (or documents) as the input, transforming them into intermediate contex-

tual representations via either convolutional or recurrent neural networks, and performing scoring over the contextual representations to yield final matching results. However, these methods rely purely on neural networks to learn the complicated relationships among sentences, and many obvious compositional and hierarchical features are often overlooked or not explicitly utilized.

In this chapter, however, we argue that a successful semantic matching algorithm needs to best characterize the sequential, hierarchical and flexible structure of natural language sentences, as well as the rich interaction patterns among semantic units. We present a technique named *Hierarchical Sentence Factorization* (or *Sentence Factorization* in short), which is able to represent a sentence in a hierarchical semantic tree, with each node (semantic unit) at different depths of the tree reorganized into a normalized "predicate-argument" form. Such normalized sentence representation enables us to propose new methods to both improve unsupervised semantic matching by taking the structural and sequential differences between two text entities into account, and enhance a range of supervised semantic matching schemes, by overcoming the limitation of the representation capability of convolutional or recurrent neural networks, especially when labelled training data is limited. Specifically, we make the following contributions:

First, the proposed *Sentence Factorization* scheme factorizes a sentence recursively into a hierarchical tree of semantic units, where each unit is a subset of words from the original sentence. Words are then reordered into a "predicate-argument" structure. Such form of sentence representation offers two benefits: i) the flexible syntax structures of the same sentence, for example, active and passive sentences, can be normalized into a unified representation; ii) the semantic units in a pair of sentences can be aligned according to their depth and order in the factorization tree.

Second, for unsupervised text matching, we combine the factorized and reordered representation of sentences and the Order-preserving Wasserstein Distance [Su and Hua, 2017] (which was originally proposed to match hand-written characters in computer vision) to propose a new semantic distance metric between text objects, which we call *Ordered Word Mover's Distance*. Compared with the recently proposed Word Mover's Distance [Kusner *et al.*, 2015], our new metric achieves significant improvement by taking the sequential structures of sentences into account. For example, without considering the order of words, the Word Mover's Distance between the sentences "Tom is chasing Jerry" and "Jerry is chasing Tom" is zero. In contrast, our new metric is able to penalize such order mismatch between words, and identify the difference between the two sentences.

Third, for supervised semantic matching, we extend the existing Siamese network

architectures (both for CNN and LSTM) to multi-scaled models, where each scale adopts an individual Siamese network, taking as input the vector representations of the two sentences at the corresponding depth in the factorization trees, ranging from the coarse-grained scale to fine-grained scales. When increasing the number of layers in the corresponding neural network can hardly improve performance, hierarchical sentence factorization provides a novel means to extend the original deep networks to a "richer" model that matches a pair of sentences through a multi-scaled semantic unit matching process. Our proposed multi-scaled deep neural networks can effectively improve existing deep models by measuring the similarity between a pair of sentences at different semantic granularities. For instance, Siamese networks based on CNN and BiLSTM [Mueller and Thyagarajan, 2016; Shao, 2017] that originally only take the word sequences as the inputs.

We extensively evaluate the performance of our proposed approaches on the task of semantic textual similarity estimation and paraphrase identification, based on multiple datasets, including the STSbenchmark dataset, the Microsoft Research Paraphrase identification (MSRP) dataset, the SICK dataset and the MSRvid dataset. Experimental results have shown that our proposed algorithms and models can achieve significant improvement compared with multiple existing unsupervised text distance metrics, such as the Word Mover's Distance [Kusner *et al.*, 2015], as well as supervised deep neural network models, including Siamese Neural Network models based on CNN and BiLSTM [Mueller and Thyagarajan, 2016; Shao, 2017].

The remainder of this chapter is organized as follows. Sec. 5.2 presents our hierarchical sentence factorization algorithm. Sec. 5.3 presents our Ordered Word Mover's Distance metric based on sentence structural reordering. In Sec. 5.4, we propose our multi-scaled deep neural network architectures based on hierarchical sentence representation. In Sec. 5.5, we conduct extensive evaluations of the proposed methods based on multiple datasets on multiple tasks. The chapter is concluded in Sec. 5.6.

5.2 Hierarchical Sentence Factorization and Reordering

In this section, we present our *Hierarchical Sentence Factorization* techniques to transform a sentence into a hierarchical tree structure, which also naturally produces a reordering of the sentence at the root node. This multi-scaled representation form proves to be effective at improving both unsupervised and supervised semantic matching, which will be discussed in Sec. 5.3 and Sec. 5.4, respectively.

We first describe our desired factorization tree structure before presenting the

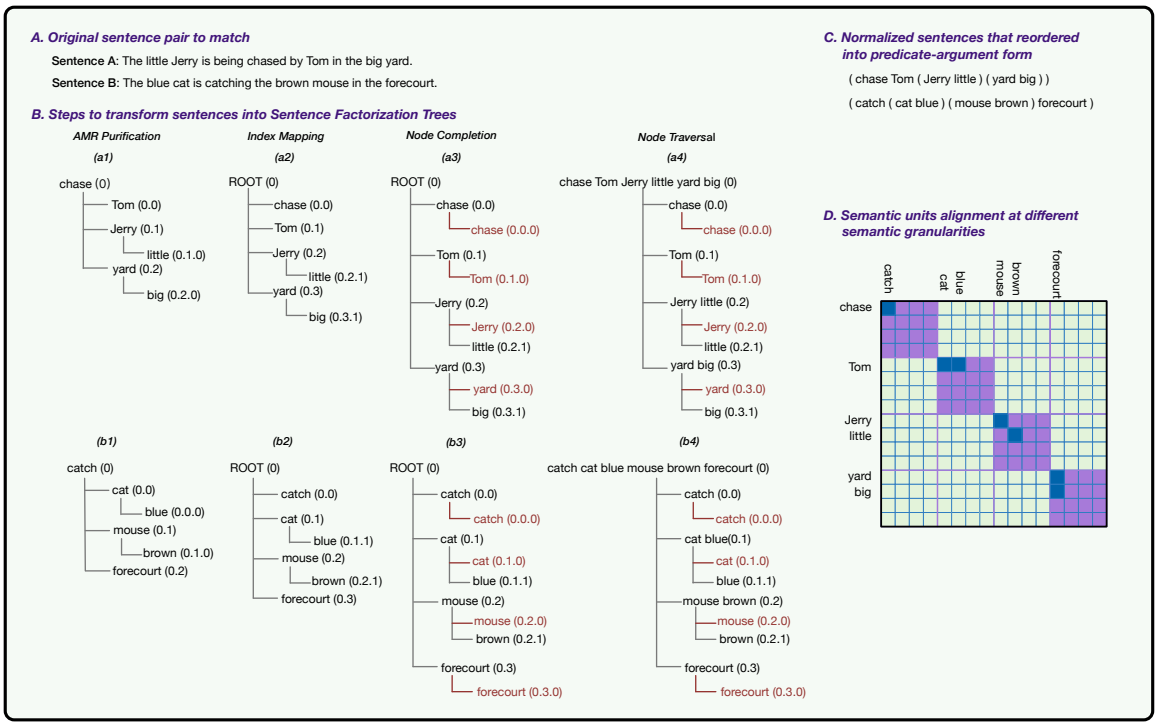


Figure 5.1: An example of the sentence factorization process. Here we show: A. The original sentence pair; B. The procedures of creating sentence factorization trees; C. The predicate-argument form of original sentence pair; D. The alignment of semantic units with the reordered form.

steps to obtain it. Given a natural language sentence S, our objective is to transform it into a semantic factorization tree denoted by T_S^f. Each node in T_S^f is called a *semantic unit*, which contains one or a few tokens (tokenized words) from the original sentence S, as illustrated in Fig. 5.1 $(a4)$, $(b4)$. The tokens in every semantic unit in T_S^f is re-organized into a "predicate-argument" form. For example, a semantic unit for "Tom catches Jerry" in the "predicate-argument" form will be "catch Tom Jerry".

Our proposed factorization tree recursively factorizes a sentence into a hierarchy of semantic units at different granularities to represent the semantic structure of that sentence. The root node in a factorization tree contains the entire sentence reordered in the predicate-argument form, thus providing a "normalized" representation for sentences expressed in different ways (e.g., passive vs. active tenses). Moreover, each semantic unit at depth d will be further split into several child nodes at depth $d+1$, which are smaller semantic sub-units. Each sub-unit also follows the predicate-argument form.

For example, in Fig. 5.1, we convert sentence A into a hierarchical factorization tree $(a4)$ using a series of operations. The root node of the tree contains the semantic

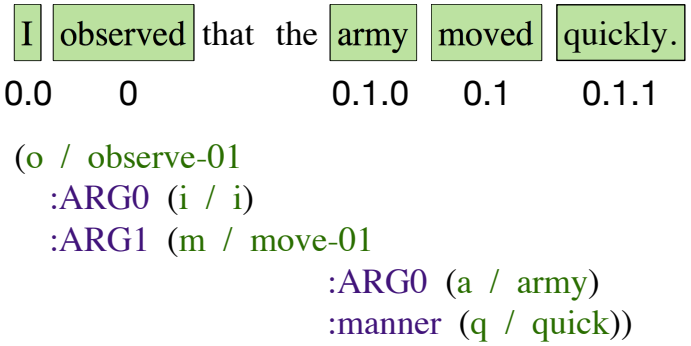


Figure 5.2: An example of a sentence and its Abstract Meaning Representation (AMR), as well as the alignment between the words in the sentence and the nodes in AMR.

unit "chase Tom Jerry little yard big", which is the reordered representation of the original sentence "The little Jerry is being chased by Tom in the big yard" in a semantically normalized form. Moreover, the semantic unit at depth 0 is factorized into four sub-units at depth 1: "chase", "Tom", "Jerry little" and "yard big", each in the "predicate-argument" form. And at depth 2, the semantic sub-unit "Jerry little" is further factorized into two sub-units "Jerry" and "little". Finally, a semantic unit that contains only one token (e.g., "chase" and "Tom" at depth 1) can not be further decomposed. Therefore, it only has one child node at the next depth through self-duplication.

We can observe that each depth of the tree contains all the tokens (except meaningless ones) in the original sentence, but re-organizes these tokens into semantic units of different granularities.

5.2.1 Hierarchical Sentence Factorization

We now describe our detailed procedure to transform a natural language sentence to the desired factorization tree mentioned above. Our Hierarchical Sentence Factorization algorithm mainly consists of five steps: 1) AMR parsing and alignment, 2) AMR purification, 3) index mapping, 4) node completion, and 5) node traversal. The latter four steps are illustrated in the example in Fig. 5.1 from left to right.

AMR parsing and alignment. Given an input sentence, the first step of our hierarchical sentence factorization algorithm is to acquire its Abstract Meaning Representation (AMR), as well as perform AMR-Sentence alignment to align the concepts in AMR with the tokens in the original sentence.

Semantic parsing [Baker *et al.*, 1998; Kingsbury and Palmer, 2002; Berant and Liang, 2014; Banarescu *et al.*, 2013; Damonte *et al.*, 2016] can be performed to gener-

ate the formal semantic representation of a sentence. Abstract Meaning Representation (AMR) [Banarescu *et al.*, 2013] is a semantic parsing language that represents a sentence by a directed acyclic graph (DAG). Each AMR graph can be converted into an AMR tree by duplicating the nodes that have more than one parent.

Fig. 5.2 shows the AMR of the sentence "I observed that the army moved quickly." In an AMR graph, leaves are labeled with concepts, which represent either English words (e.g., "army"), PropBank framesets (e.g., "observe-01") [Kingsbury and Palmer, 2002], or special keywords (e.g., dates, quantities, world regions, etc.). For example, "(a / army)" refers to an instance of the concept army, where "a" is the variable name of army (each entity in AMR has a variable name). "ARG0", "ARG1", ":manner" are different kinds of relations defined in AMR. Relations are used to link entities. For example, ":manner" links "m / move-01" and "q / quick", which means "move in a quick manner". Similarly, ":ARG0" links "m / move-01" and "a / army", which means that "army" is the first argument of "move".

Each leaf in AMR is a concept rather than the original token in a sentence. The alignment between a sentence and its AMR graph is not given in the AMR annotation. Therefore, AMR alignment [Pourdamghani *et al.*, 2014] needs to be performed to link the leaf nodes in the AMR to tokens in the original sentence. Fig. 5.2 shows the alignment between sentence tokens and AMR concepts by the alignment indexes. The alignment index 0 is for the root node, 0.0 for the first child of the root node, 0.1 for the second child of the root node, and so forth. For example, in Fig. 5.2, the word "army" in sentence is linked with index "0.1.0", which represents the concept node "a / army" in its AMR. We refer interested readers to [Banarescu *et al.*, 2013; Banarescu *et al.*, 2012] for more detailed description about AMR.

Various parsers have been proposed for AMR parsing and alignment [Flanigan *et al.*, 2014; Wang *et al.*, 2015*a*]. We choose the JAMR parser [Flanigan *et al.*, 2014] in our algorithm implementation.

AMR purification. Unfortunately, AMR itself cannot be used to form the desired factorization tree. First, it is likely that multiple concepts in AMR may link to the same token in the sentence. For example, Fig. 5.3 shows AMR and its alignment for the sentence "Three Asian kids are dancing.". The token "Asian" is linked to four concepts in the AMR graph: " continent (0.0.0)", "name (0.0.0.0)", "Asia (0.0.0.0.0)" and "wiki Asia (0.0.0.1)". This is because AMR will match a named entity with predefined concepts which it belongs to, such as "c / continent" for "Asia", and form a compound representation of the entity. For example, in Fig.5.3, the token "Asian" is represented as a continent whose name is Asia, and its Wikipedia entity name is also Asia.

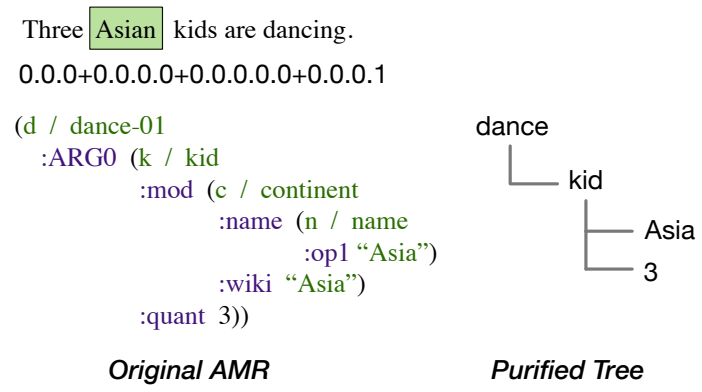


Figure 5.3: An example to show the operation of AMR purification.

In this case, we select the link index with the smallest tree depth as the token's position in the tree. Suppose $\mathcal{P}_w = \{p_1, p_2, \cdots, p_{|\mathcal{P}|}\}$ denotes the set of alignment indexes of token w. We can get the desired alignment index of w by calculating the longest common prefix of all the index strings in $\mathcal{P}_w$. After getting the alignment index for each token, we then replace the concepts in AMR with the tokens in sentence by the alignment indexes, and remove relation names (such as ":ARG0") in AMR, resulting into a compact tree representation of the original sentence, as shown in the right part of Fig. 5.3.

Index mapping. A purified AMR tree for a sentence obtained in the previous step is still not in our desired form. To transform it into a hierarchical sentence factorization tree, we perform index mapping and calculate a new position (or index) for each token in the desired factorization tree given its position (or index) in the purified AMR tree. Fig. 5.1 illustrates the process of index mapping. After this step, for example, the purified AMR trees in Fig. 5.1 $(a1)$ and $(b1)$ will be transformed into $(a2)$ and $(b2)$.

Specifically, let T_S^p denote a purified AMR tree of sentence S, and T_S^f our desired sentence factorization tree of S. Let $I_N^p = \overline{i_0.i_1.i_2.\cdots.i_d}$ denote the index of node N in T_S^p, where d is the depth of N in T_S^p (where depth 0 represents the root of a tree). Then, the index I_N^f of node N in our desired factorization tree T_S^f will be calculated as follows:

$$I_N^f := \begin{cases} \overline{0.0} & \text{if } d = 0, \\ \overline{i_0.(i_1+1).(i_2+1).\cdots.(i_d+1)} & \text{otherwise.} \end{cases} \tag{5.1}$$

After index mapping, we add an empty root node with index 0 in the new factorization tree, and link all nodes at depth 1 to it as its child nodes. Note that the i_0 in every node index will always be 0.

Node completion. We then perform node completion to make sure each branch

of the factorization tree have the same maximum depth and to fill in the missing nodes caused by index mapping, illustrated by Fig. 5.1 ($a3$) and ($b3$).

First, given a pre-defined maximum depth D, for each leaf node N^l with depth $d < D$ in the current T_S^f after index mapping, we duplicate it for $D - d$ times and append all of them sequentially to N^l, as shown in Fig. 5.1 ($a3$), ($b3$), such that the depths of the ending nodes will always be D. For example, in Fig. 5.1 with $D = 2$, the node "chase (0.0)" and "Tom (0.1)" will be extended to reach depth 2 via self-duplication.

Second, after index mapping, the children of all the non-leaf nodes, except the root node, will be indexed starting from 1 rather than 0. For example, in Fig. 5.1 ($a2$), the first child node of "Jerry (0.2)" is "little (0.2.1)". In this case, we duplicate "Jerry (0.2)" itself to "Jerry (0.2.0)" to fill in the missing first child of "Jerry (0.2)". Similar filling operations are done for other non-leaf nodes after index mapping as well.

Node traversal to complete semantic units. Finally, we complete each semantic unit in the formed factorization tree via node traversal, as shown in Fig. 5.1 ($a4$), ($b4$). For each non-leaf node N, we traverse its sub-tree by Depth First Search (DFS). The original semantic unit in N will then be replaced by the concatenation of the semantic units of all the nodes in the sub-tree rooted at N, following the order of traversal.

For example, for sentence A in Fig. 5.1, after node traversal, the root node of the factorization tree becomes "chase Tom Jerry little yard big" with index "0". We can see that the original sentence has been reordered into a predicate-argument structure. A similar structure is generated for the other nodes at different depths. Until now, each depth of the factorization tree T_S^f can express the full sentence S in terms of semantic units at different granularity.

5.3 Ordered Word Mover's Distance

The proposed hierarchical sentence factorization technique naturally reorders an input sentence into a unified format at the root node. In this section, we introduce the *Ordered Word Mover's Distance* metric which measures the semantic distance between two input sentences based on the unified representation of reordered sentences.

Assume $X \in \mathbb{R}^{d \times n}$ is a *word2vec* embedding matrix for a vocabulary of n words, and the i-th column $\mathbf{x}_i \in \mathbb{R}^d$ represents the d-dimensional embedding vector of i-th word in vocabulary. Denote a sentence $S = \overline{a_1 a_2 \cdots a_K}$ where a_i represents the i-th word (or the word embedding vector). The Word Mover's Distance considers a

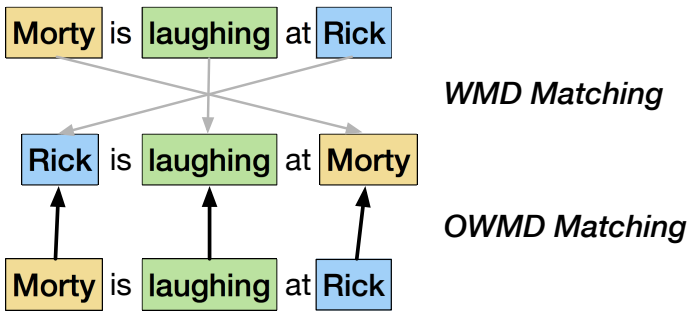


Figure 5.4: Compare the sentence matching results given by Word Mover's Distance and Ordered Word Mover's Distance.

sentence S as its normalized bag-of-words (nBOW) vectors where the weights of the words in S is $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \cdots, \alpha_K\}$. Specifically, if word a_i appears c_i times in S, then $\alpha_i = \frac{c_i}{\sum_{j=1}^{K} c_j}$.

The Word Mover's Distance metric combines the normalized bag-of-words representation of sentences with Wasserstein distance (also known as Earth Mover's Distance [Rubner *et al.*, 2000]) to measure the semantic distance between two sentences. Given a pair of sentences $S_1 = \overline{a_1 a_2 \cdots a_M}$ and $S_2 = \overline{b_1 b_2 \cdots b_N}$, where $b_j \in \mathbb{R}^d$ is the embedding vector of the j-th word in S_2. Let $\boldsymbol{\alpha} = \{\alpha_1, \cdots, \alpha_M\}$ and $\boldsymbol{\beta} = \{\beta_1, \cdots, \beta_N\}$ represents the normalized bag-of-words vectors of S_1 and S_2. We can calculate a distance matrix $D \in \mathbb{R}^{M \times N}$ where each element $D_{ij} = \|a_i - b_j\|_2$ measures the distance between word a_i and b_j (we use the same notation to denote the word itself or its word vector representation). Let $T \in \mathbb{R}^{M \times N}$ be a non-negative sparse transport matrix where T_{ij} denotes the portion of word $a_i \in S_1$ that transports to word $b_j \in S_2$. The Word Mover's Distance between sentences S_1 and S_2 is given by $\sum_{i,j} T_{ij} D_{ij}$. The transport matrix T is computed solving the following constrained optimization problem:

$$\begin{aligned} \underset{T \in \mathbb{R}_+^{M \times N}}{\text{minimize}} \quad & \sum_{i,j} T_{ij} D_{ij} \\ \text{subject to} \quad & \sum_{i=1}^{M} T_{ij} = \beta_j \quad 1 \le j \le N, \\ & \sum_{j=1}^{N} T_{ij} = \alpha_i \quad 1 \le i \le M. \end{aligned} \tag{5.2}$$

Where the minimum "word travel cost" between two bags of words for a pair of sentences is calculated to measure the their semantic distance.

However, the Word Mover's Distance fails to consider a few aspects of natural language. First, it omits the sequential structure. For example, in Fig. 5.4, the pair

of sentences "Morty is laughing at Rick" and "Rick is laughing at Morty" only differ in the order of words. The Word Mover's Distance metric will then find an exact match between the two sentences and estimate the semantic distance as zero, which is obviously false. Second, the normalized bag-of-words representation of a sentence can not distinguish duplicated words shown in multiple positions of a sentence.

To overcome the above challenges, we propose a new kind of semantic distance metric named Ordered Word Mover's Distance (OWMD). The Ordered Word Mover's Distance combines our sentence factorization technique with Order-preserving Wasserstein Distance proposed in [Su and Hua, 2017]. It casts the calculation of semantic distance between texts as an optimal transport problem while preserving the sequential structure of words in sentences. The Ordered Word Mover's Distance differs from the Word Mover's Distance in multiple aspects.

First, rather than using normalized bag-of-words vector to represent a sentence, we decompose and re-organize a sentence using the sentence factorization algorithm described in Sec. 5.2. Given a sentence S, we represent it by the reordered word sequence S' in the root node of its sentence factorization tree. Such representation normalizes a sentence into "predicate-argument" structure to better handle syntactic variations. For example, after performing sentence factorization, sentences "Tom is chasing Jerry" and "Jerry is being chased by Tom" will both be normalized as "chase Tom Jerry".

Second, we calculate a new transport matrix T by solving the following optimization problem

$$\begin{aligned} \underset{T\in\mathbb{R}_+^{M\times N}}{\text{minimize}} \quad & \sum_{i,j} T_{ij}D_{ij} - \lambda_1 I(T) + \lambda_2 KL(T||P) \\ \text{subject to} \quad & \sum_{i=1}^{M} T_{ij} = \beta'_j \quad 1 \le j \le N', \\ & \sum_{j=1}^{N} T_{ij} = \alpha'_i \quad 1 \le i \le M', \end{aligned} \tag{5.3}$$

where $\lambda_1 > 0$ and $\lambda_2 > 0$ are two hyper parameters. M' and N' denotes the number of words in S'_1 and S'_2. α'_i denotes the weight of the i-th word in normalized sentence S'_1 and β'_j denotes the weight of the j-th word in normalized sentence S'_2. Usually we can set $\boldsymbol{\alpha'} = (\frac{1}{M'}, \cdots, \frac{1}{M'})$ and $\boldsymbol{\beta'} = (\frac{1}{N'}, \cdots, \frac{1}{N'})$ without any prior knowledge of word differences.

The first penalty term $I(T)$ is the inverse difference moment [Albregtsen *et al.*, 2008] of the transport matrix T that measures local homogeneity of T. It is defined

as:

$$I(T) = \sum_{i=1}^{M'} \sum_{j=1}^{N'} \frac{T_{ij}}{(\frac{i}{M'} - \frac{j}{N'})^2 + 1}. \tag{5.4}$$

$I(T)$ will have a relatively large value if the large values of T mainly appear near its diagonal.

Another penalty term $KL(T||P)$ denotes the KL-divergence between T and P. P is a two-dimensional distribution used as the prior distribution for values in T. It is defined as

$$P_{ij} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{l^2(i,j)}{2\sigma^2}} \tag{5.5}$$

where $l(i, j)$ is the distance from position (i, j) to the diagonal line, which is calculated as

$$l(i, j) = \frac{|i/M' - j/N'|}{\sqrt{1/M'^2 + 1/N'^2}}. \tag{5.6}$$

As we can see, the farther a word in one sentence is from the other word in another sentence in terms of word orders, the less likely it will be transported to that word. Therefore, by introducing the two penalty terms $I(T)$ and $KL(T||P)$ into problem (5.3), we encourage words at similar positions in two sentences to be matched. Words at distant positions are less likely to be matched by T.

The problem (5.3) has a unique optimal solution T^{λ_1,λ_2} since both the objective and the feasible set are convex. It has been proved in [Su and Hua, 2017] that the optimal T^{λ_1,λ_2} has the same form with $diag(\mathbf{k}_1) \cdot \mathbf{K} \cdot diag(\mathbf{k}_2)$, where $diag(\mathbf{k}_1) \in \mathbb{R}^{M'}$ and $diag(\mathbf{k}_2) \in \mathbb{R}^{N'}$ are two diagonal matrices with strictly positive diagonal elements. $\mathbf{K} \in \mathbb{R}^{M' \times N'}$ is a matrix defined as

$$K_{ij} = P_{ij} e^{\frac{1}{\lambda_2}(S_{ij}^{\lambda_1} - D_{ij})}, \tag{5.7}$$

where

$$S_{ij} = \frac{\lambda_1}{(\frac{i}{M'} - \frac{j}{N'})^2 + 1}. \tag{5.8}$$

The two matrices $\mathbf{k}_1$ and $\mathbf{k}_2$ can be efficiently obtained by the Sinkhorn-Knopp iterative matrix scaling algorithm [Knight, 2008]:

$$\begin{aligned} \mathbf{k}_1 &\leftarrow \boldsymbol{\alpha}'./K\mathbf{k}_2, \\ \mathbf{k}_2 &\leftarrow \boldsymbol{\beta}'./K^T\mathbf{k}_1. \end{aligned} \tag{5.9}$$

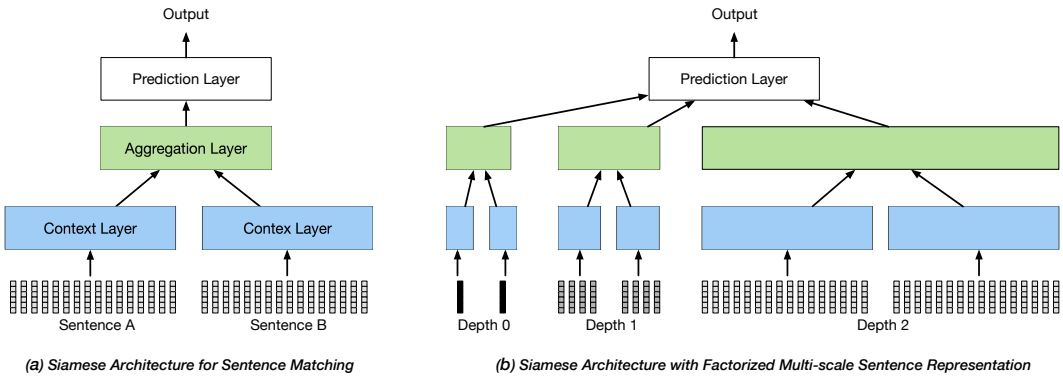


Figure 5.5: Extend the Siamese network architecture for sentence matching by feeding into the multi-scale representations of sentence pairs.

where ./ is the element-wise division operation. Compared with Word Mover's Distance, the Ordered Word Mover's Distance considers the positions of words in a sentence, and is able to distinguish duplicated words at different locations. For example, in Fig. 5.4, while the WMD finds an exact match and get a semantic distance of zero for the sentence pair "Morty is laughing at Rick" and "Rick is laughing at Morty", the OWMD metric is able to find a better match relying on the penalty terms, and gives a semantic distance greater than zero.

The computational complexity of OWMD is also effectively reduced compared to WMD. With the additional constraints, the time complexity is $O(dM'N')$ where d is the dimension of word vectors [Su and Hua, 2017], while it is $O(dp^3 \log p)$ for WMD, where p denotes the number of uniques words in sentences or documents [Kusner *et al.*, 2015].

5.4 Multi-scale Sentence Matching

Our sentence factorization algorithm parses a sentence S into a hierarchical factorization tree T_S^f, where each depth of T_S^f contains the semantic units of the sentence at a different granularity. In this section, we exploit this multi-scaled representation of S present in T_S^f to propose a multi-scaled Siamese network architecture that can extend any existing CNN or RNN-based Siamese architectures to leverage the hierarchical representation of sentence semantics.

Fig. 5.5 (a) shows the network architecture of the popular Siamese "matching-aggregation" framework [Wang and Jiang, 2016; Mueller and Thyagarajan, 2016; Severyn and Moschitti, 2015; Neculoiu *et al.*, 2016; Baudiš *et al.*, 2016] for sentence matching tasks. The matching process is usually performed as follows: First, the

sequence of word embeddings in two sentences will be encoded by a context representation layer, which usually contains one or multiple layers of LSTM, bi-directional LSTM (BiLSTM), or CNN with max pooling layers. The goal is to capture the contextual information of each sentence into a context vector. In a Siamese network, every sentence is encoded by the same context representation layer. Second, the context vectors of two sentences will be concatenated in the aggregation layer. They may be further transformed by more layers of neural network to get a fixed length matching vector. Finally, a prediction layer will take in the matching vector and outputs a similarity score for the two sentences or the probability distribution over different sentence-pair relationships.

Compared with the typical Siamese network shown in Fig. 5.5 (a), our proposed architecture shown in Fig. 5.5 (b) differs in two aspects. First, our network contains three Siamese sub-modules that are similar to (a). They correspond to the factorized representations from depth 0 (the root layer) to depth 2. We only select the semantic units from the top 3 depths of the factorization tree as our input, because usually most semantic units at depth 2 are already single words and can not be further factorized. Second, for each Siamese sub-module in our network architecture, the input is not the embedding vectors of words from the original sentences. Instead, we use semantic units at different depths of sentence factorization tree for matching. We sum up the embedding vectors of the words contained in a semantic unit to represent that unit. Assuming each semantic unit at depth d can be factorized into k semantic sub-units at depth $d+1$. If a semantic unit has less than k sub-units, we add empty units as its child node to make each non-leaf node in a factorization tree has exactly k child nodes. The empty units are embedded with a vector of zeros. After this procedure, the number of semantic units at depth d of a sentence factorization tree is k^d.

Taking Fig. 5.1 as an example. We set $k = 4$ in Fig. 5.1. For sentence A "The little Jerry is being chased by Tom in the big yard", the input at depth 0 is the sum of word embedding {chase, Tom, Jerry, little, yard, big}. The input at depth 1 are the embedding vectors of four semantic units: {chase, Tome, Jerry little, yard big}. Finally, at depth 2, the semantic units are {chase, -, -, -, Tom, -, -, -, Jerry, little, -, -, yard, big, -, -}, where "−" denotes an empty unit.

As we can see, based on this factorized sentence representation, our network architecture explicitly matches a pair of sentences at several semantic granularities. In addition, we align the semantic units in two sentences by mapping their positions in the tree to the corresponding indices in the input layer of the neural network. For example, as shown in Fig. 5.1, the semantic units at depth 2 are aligned according to their unit indices: "chase" matches with "catch", "Tom" matches with "cat blue",

Dataset	Task	Train	Dev	Test
STSbenchmark	Similarity scoring	5748	1500	1378
SICK	Similarity scoring	4500	500	4927
MSRP	Paraphrase identification	4076	-	1725
MSRvid	Similarity scoring	750	-	750

Table 5.1: Description of evaluation datasets.

"Jerry little" matches with "mouse brown", and "yard big" matches with "forecourt".

5.5 Evaluation

In this section, we evaluate the performance of our unsupervised Ordered Word Mover's Distance metric and supervised Multi-scale Sentence Matching model with factorized sentences as input. We apply our algorithms to semantic textual similarity estimation tasks and sentence pair paraphrase identification tasks, based on four datasets: STSbenchmark, SICK, MSRP and MSRvid.

5.5.1 Experimental Setup

We will start with a brief description for each dataset:

- **STSbenchmark**[Cer *et al.*, 2017]: it is a dataset for semantic textual similarity (STS) estimation. The task is to assign a similarity score to each sentence pair on a scale of 0.0 to 5.0, with 5.0 being the most similar.
- **SICK**[Marelli *et al.*, 2014]: it is another STS dataset from the SemEval 2014 task 1. It has the same scoring mechanism as STSbenchmark, where 0.0 represents the least amount of relatedness and 5.0 represents the most.
- **MSRvid**: the Microsoft Research Video Description Corpus contains 1500 sentences that are concise summaries on the content of a short video. Each pair of sentences is also assigned a semantic similarity score between 0.0 and 5.0.
- **MSRP**[Quirk *et al.*, 2004]: the Microsoft Research Paraphrase Corpus is a set of 5800 sentence pairs collected from news articles on the Internet. Each sentence pair is labeled 0 or 1, with 1 indicating that the two sentences are paraphrases of each other.

Table 5.1 shows a detailed breakdown of the datasets used in evaluation. For STSbenchmark dataset we use the provided train/dev/test split. The SICK dataset does not provide development set out of the box, so we extracted 500 instances from the training set as the development set. For MSRP and MSRvid, since their sizes are relatively small to begin with, we did not create any development set for them.

One metric we used to evaluate the performance of our proposed models on the task of semantic textual similarity estimation is the Pearson Correlation coefficient, commonly denoted by r. Pearson Correlation is defined as:

$$r = cov(X, Y)/(\sigma_X \sigma_Y), \tag{5.10}$$

where $cov(X, Y)$ is the co-variance between distributions X and Y, and σ_X, σ_Y are the standard deviations of X and Y. The Pearson Correlation coefficient can be thought as a measure of how well two distributions fit on a straight line. Its value has range [-1, 1], where a value of 1 indicates that data points from two distribution lie on the same line with a positive slope.

Another metric we utilized is the Spearman's Rank Correlation coefficient. Commonly denoted by r_s, the Spearman's Rank Correlation coefficient shares a similar mathematical expression with the Pearson Correlation coefficient, but it is applied to ranked variables. Formally it is defined as [Wikipedia, 2017]:

$$\rho = cov(rg_X, rg_Y)/(\sigma_{rg_X} \sigma_{rg_Y}), \tag{5.11}$$

where rg_X, rg_Y denotes the ranked variables derived from X and Y. $cov(rg_X, rg_Y)$, σ_{rg_X}, σ_{rg_Y} corresponds to the co-variance and standard deviations of the rank variables. The term ranked simply means that each instance in X is ranked higher or lower against every other instances in X and the same for Y. We then compare the rank values of X and Y with 5.11. Like the Pearson Correlation coefficient, the Spearman's Rank Correlation coefficient has an output range of [-1, 1], and it measures the monotonic relationship between X and Y. A Spearman's Rank Correlation value of 1 implies that as X increases, Y is guaranteed to increase as well. The Spearman's Rank Correlation is also less sensitive to noise created by outliers compared to the Pearson Correlation.

For the task of paraphrase identification, the classification accuracy of label 1 and the F1 score are used as metrics.

In the supervised learning portion, we conduct the experiments on the aforementioned four datasets. We use training sets to train the models, development set to tune the hyper-parameters and each test set is only used once in the final evaluation. For datasets without any development set, we will use cross-validation in the training

process to prevent overfitting, that is, use 10% of the training data for validation and the rest is used in training. For each model, we carry out training for 10 epochs. We then choose the model with the best validation performance to be evaluated on the test set.

5.5.2 Unsupervised Matching with OWMD

To evaluate the effectiveness of our Ordered Word Mover's Distance metric, we first take an unsupervised approach towards the similarity estimation task on the STS-benchmark, SICK and MSRvid datasets. Using the distance metrics listed in Table 5.2 and 5.3, we first computed the distance between two sentences, then calculated the Pearson Correlation coefficients and the Spearman's Rank Correlation coefficients between all pair's distances and their labeled scores. We did not use the MSRP dataset since it is a binary classification problem.

In our proposed Ordered Word Mover's Distance metric, distance between two sentences is calculated using the order preserving Word Mover's Distance algorithm. For all three datasets, we performed hyper-parameter tuning using the training set and calculated the Pearson Correlation coefficients on the test and development set. We found that for the STSbenchmark dataset, setting $\lambda_1 = 10$, $\lambda_2 = 0.03$ produces the most optimal result. For the SICK dataset, a combination of $\lambda_1 = 3.5$, $\lambda_2 = 0.015$ works best. And for the MSRvid dataset, the highest Pearson Correlation is attained when $\lambda_1 = 0.01$, $\lambda_2 = 0.02$. We maintain a max iteration of 20 since in our experiments we found that it is sufficient for the correlation result to converge. During hyper-parameter tuning we discovered that using the Euclidean metric along with $\sigma = 10$ produces better results, so all OWMD results summarized in Table 5.2 and 5.3 are acquired under these parameter settings. Finally, it is worth mentioning that our OWMD metric calculates the distances using factorized versions of sentences, while all other metrics use the original sentences. Sentence factorization is a necessary preprocessing step for the OWMD metric.

We compared the performance of Ordered Word Mover's Distance metric with the following methods:

- **Bag-of-Words (BoW)**: in the Bag-of-Words metric, distance between two sentences is computed as the cosine similarity between the word counts of the sentences.

- **LexVec** [Salle *et al.*, 2016]: calculate the cosine similarity between the averaged 300-dimensional LexVec word embedding of the two sentences.

Algorithm	STSbenchmark		SICK		MSRvid
	Test	Dev	Test	Dev	Test
BoW	0.5705	0.6561	0.6114	0.6087	0.5044
LexVec	0.5759	0.6852	0.6948	**0.6811**	0.7318
GloVe	0.4064	0.5207	0.6297	0.5892	0.5481
Fastext	0.5079	0.6247	0.6517	0.6421	0.5517
Word2vec	0.5550	0.6911	**0.7021**	0.6730	0.7209
WMD	0.4241	0.5679	0.5962	0.5953	0.3430
OWMD	**0.6144**	**0.7240**	0.6797	0.6772	**0.7519**

Table 5.2: Pearson Correlation results on different distance metrics.

Algorithm	STSbenchmark		SICK		MSRvid
	Test	Dev	Test	Dev	Test
BoW	0.5592	0.6572	0.5727	0.5894	0.5233
LexVec	0.5472	0.7032	0.5872	0.5879	0.7311
GloVe	0.4268	0.5862	0.5505	0.5490	0.5828
Fastext	0.4874	0.6424	0.5739	0.5941	0.5634
Word2vec	0.5184	0.7021	0.6082	0.6056	0.7175
WMD	0.4270	0.5781	0.5488	0.5612	0.3699
OWMD	**0.5855**	**0.7253**	**0.6133**	**0.6188**	**0.7543**

Table 5.3: Spearman's Rank Correlation results on different distance metrics.

- **GloVe** [Pennington *et al.*, 2014]: calculate the cosine similarity between the averaged 300-dimensional GloVe 6B word embedding of the two sentences.
- **Fastext** [Joulin *et al.*, 2016]: calculate the cosine similarity between the averaged 300-dimensional Fastext word embedding of the two sentences.
- **Word2vec** [Mikolov *et al.*, 2013]: calculate the cosine similarity between the averaged 300-dimensional Word2vec word embedding of the two sentences.
- **Word Mover's Distance (WMD)** [Kusner *et al.*, 2015]: estimating the semantic distance between two sentences by WMD introduced in Sec. 5.3.

Table 5.2 and Table 5.3 compare the performance of different metrics in terms of the Pearson Correlation coefficients and the Spearman's Rank Correlation coefficients. We can see that the result of our OWMD metric achieves the best performance on

Model	MSRP		SICK		MSRvid		STSbenchmark	
	Acc.(%)	F1(%)	r	ρ	r	ρ	r	ρ
MaLSTM	66.95	73.95	0.7824	0.71843	0.7325	0.7193	0.5739	0.5558
Multi-scale MaLSTM	**74.09**	**82.18**	**0.8168**	**0.74226**	**0.8236**	**0.8188**	**0.6839**	**0.6575**
HCTI	73.80	80.85	0.8408	0.7698	**0.8848**	**0.8763**	**0.7697**	**0.7549**
Multi-scale HCTI	**74.03**	**81.76**	**0.8437**	**0.7729**	0.8763	0.8686	0.7269	0.7033

Table 5.4: A comparison among different supervised learning models in terms of accuracy, F1 score, Pearson's r and Spearman's ρ on various test sets.

all the datasets in terms of the Spearman's Rank Correlation coefficients. It also produced the best Pearson Correlation results on the STSbenchmark and the MSRvid dataset, while the performance on SICK datasets are close to the best. This can be attributed to the two characteristics of OWMD. First, the input sentence is re-organized into a predicate-argument structure using the sentence factorization tree. Therefore, corresponding semantic units in the two sentences will be aligned roughly in order. Second, our OWMD metric takes word positions into consideration and penalizes disordered matches. Therefore, it will produce less mismatches compared with the WMD metric.

5.5.3 Supervised Multi-scale Semantic Matching

The use of sentence factorization can improve both existing unsupervised metrics and existing supervised models. To evaluate how the performance of existing Siamese neural networks can be improved by our sentence factorization technique and the multi-scale Siamese architecture, we implemented two types of Siamese sentence matching models, HCTI [Mueller and Thyagarajan, 2016] and MaLSTM [Shao, 2017]. HCTI is a Convolutional Neural Network (CNN) based Siamese model, which achieves the best Pearson Correlation coefficient on STSbenchmark dataset in SemEval2017 competition (compared with all the other neural network models). MaLSTM is a Siamese adaptation of the Long Short-Term Memory (LSTM) network for learning sentence similarity. As the source code of HCTI is not released in public, we implemented it according to [Shao, 2017] by Keras [Chollet *et al.*, 2015]. With the same parameter settings listed in paper [Shao, 2017] and tried our best to optimize the model, we got a Pearson correlation of 0.7697 (0.7833 in paper [Shao, 2017]) in STSbencmark test dataset.

We extended HCTI and MaLSTM to our proposed Siamese architecture in Fig. 5.5, namely the Multi-scale MaLSTM and the Multi-scale HCTI. To evaluate the

performance of our models, the experiment is conducted on two tasks: 1) semantic textual similarity estimation based on the STSbenchmark, MSRvid, and SICK2014 datasets; 2) paraphrase identification based on the MSRP dataset.

Table 5.4 shows the results of HCTI, MaLSTM and our multi-scale models on different datasets. Compared with the original models, our models with multi-scale semantic units of the input sentences as network inputs significantly improved the performance on most datasets. Furthermore, the improvements on different tasks and datasets also proved the general applicability of our proposed architecture.

Compared with MaLSTM, our multi-scaled Siamese models with factorized sentences as input perform much better on each dataset. For MSRvid and STSbenmark dataset, both Pearson's r and Spearman's ρ increase about 10% with Multi-scale MaLSTM. Moreover, the Multi-scale MaLSTM achieves the highest accuracy and F1 score on the MSRP dataset compared with other models listed in Table 5.4.

There are two reasons why our Multi-scale MaLSTM significantly outperforms MaLSTM model. First, for an input sentence pair, we explicitly model their semantic units with the factorization algorithm. Second, our multi-scaled network architecture is specifically designed for multi-scaled sentences representations. Therefore, it is able to explicitly match a pair of sentences at different granularities.

We also report the results of HCTI and Multi-scale HCTI in Table 5.4. For the paraphrase identification task, our model shows better accuracy and F1 score on MSRP dataset. For the semantic textual similarity estimation task, the performance varies across datasets. On the SICK dataset, the performance of Multi-scale HCTI is close to HCTI with slightly better Pearson' r and Spearman's ρ. However, the Multi-scale HCTI is not able to outperform HCTI on MSRvid and STSbenchmark. HCTI is still the best neural network model on the STSbenchmark dataset, and the MSRvid dataset is a subset of STSbenchmark. Although HCTI has strong performance on these two datasets, it performs worse than our model on other datasets. Overall, the experimental results demonstrated the general applicability of our proposed model architecture, which performs well on various semantic matching tasks.

5.6 Conclusion

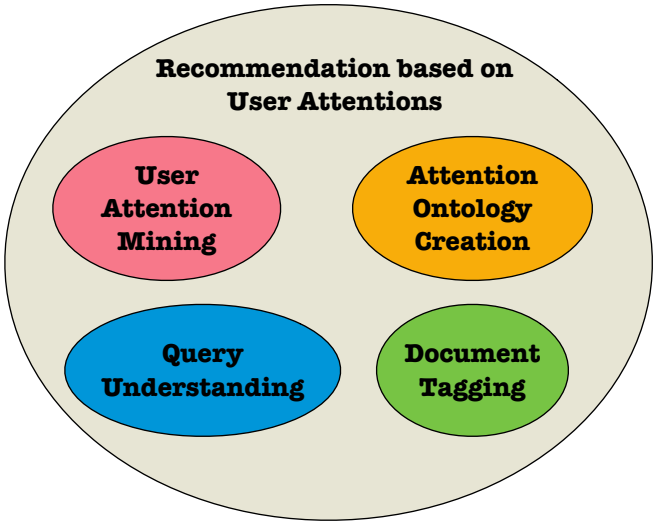
In this chapter, we propose a technique named *Hierarchical Sentence Factorization* that is able to transform a sentence into a hierarchical factorization tree. Each node in the tree is a semantic unit consists of one or several words in the sentence and reorganized into the form of "predicate-argument" structure. Each depth in the tree factorizes the sentence into semantic units of different scales. Based on the hierarchical

tree-structured representation of sentences, we propose both an unsupervised metric and two supervised deep models for sentence matching tasks. On one hand, we design a new unsupervised distance metric, named *Ordered Word Mover's Distance* (OWMD), to measure the semantic difference between a pair of text snippets. OWMD takes the sequential structure of sentences into account, and is able to handle the flexible syntactical structure of natural language sentences. On the other hand, we propose the multi-scale Siamese neural network architecture which takes the multi-scale representation of a pair of sentences as network input and matches the two sentences at different granularities.

We apply our techniques to the task of text-pair similarity estimation and the task of text-pair paraphrase identification, based on multiple datasets. Our extensive experiments show that both the unsupervised distance metric and the supervised multi-scale Siamese network architecture can achieve significant improvement on multiple datasets using the technique of sentence factorization.

Part II

Text Mining: Recognizing User Attentions for Searching and Recommendation



How to infer users' interests or attentions based on his/her historical behaviors, and how to identify the relationships between different user interests are critical research problems for recommender systems.

In chapter 6, we describe *ConcepT* in Tencent QQ Browser. It discovers user-centered concepts at the right granularity conforming to user interests, by mining a large amount of user queries and interactive search click logs. The extracted concepts have the proper granularity, are consistent with user language styles and are dynamically updated.

In chapter 7, we present *GIANT*, a mechanism to construct a user-centered, web-scale, structured ontology, containing a large number of natural language phrases conforming to user attentions at various granularities, mined from the vast volume of web documents and search click logs. Various types of edges are also constructed to maintain a hierarchy in the ontology. Compare with *ConcepT*, *GIANT* contains more types of user attentions, such as topics and events. Besides, it identifies various types of relationships between user attentions instead of only *isA* relationship. Furthermore, *GIANT* contains a unified algorithm for heterogeneous phrase mining based on a novel Query Title Interaction Graph representation.

Both *ConcepT* and *GIANT* were deployed into real-world applications, such as Tencent QQ Browser. They can be applied in different tasks, including query understanding, document tagging, story organization and so on.

Chapter 6

A User-Centered Concept Mining System for Query and Document Understanding at Tencent

Concepts embody the knowledge of the world and facilitate the cognitive processes of human beings. Mining concepts from web documents and constructing the corresponding taxonomy are core research problems in text understanding and support many downstream tasks such as query analysis, knowledge base construction, recommendation, and search. However, we argue that most prior studies extract formal and overly general concepts from Wikipedia or static web pages, which are not representing the user perspective. In this chapter, we describe our experience of implementing and deploying *ConcepT* in Tencent QQ Browser. It discovers user-centered concepts at the right granularity conforming to user interests, by mining a large amount of user queries and interactive search click logs. The extracted concepts have the proper granularity, are consistent with user language styles and are dynamically updated. We further present our techniques to tag documents with user-centered concepts and to construct a *topic-concept-instance* taxonomy, which has helped to improve search as well as news feeds recommendation in Tencent QQ Browser. We performed extensive offline evaluation to demonstrate that our approach could extract concepts of higher quality compared to several other existing methods. Our system has been deployed in Tencent QQ Browser. Results from online A/B testing involving a large number of real users suggest that the Impression Efficiency of feeds users increased by 6.01% after incorporating the user-centered concepts into the recommendation framework of Tencent QQ Browser.

6.1 Introduction

The capability of *conceptualization* is a critical ability in natural language understanding and is an important distinguishing factor that separates a human being from the current dominating machine intelligence based on vectorization. For example, by observing the words "Honda Civic" and "Hyundai Elantra", a human can immediately link them with "fuel-efficient cars" or "economy cars", and quickly come up with similar items like "Nissan Versa" and probably "Ford Focus". When one observes the seemingly uncorrelated words "beer", "diaper" and "Walmart", one can extrapolate that the article is most likely discussing topics like marketing, business intelligence or even data science, instead of talking about the actual department store "Walmart". The importance of concepts is best emphasized by the statement in Gregory Murphy's famous book *The Big Book of Concepts* that "Concepts embody our knowledge of the kinds of things there are in the world. Tying our past experiences to our present interactions with the environment, they enable us to recognize and understand new objects and events."

In order to enable machines to extract concepts from text, a large amount of effort has been devoted to knowledge base or taxonomy construction, typically represented by DBPedia [Lehmann *et al.*, 2015] and YAGO [Suchanek *et al.*, 2007] which construct taxonomies from Wikipedia categories, and Probase [Wu *et al.*, 2012] which extracts concepts from free text in web documents. However, we argue that these methods for concept extraction and taxonomy construction are still limited as compared to how a human interacts with the world and learns to conceptualize, and may not possess the proper granularity that represents human interests. For example, "Toyota 4Runner" is a "Toyota SUV" and "F150" is a "truck". However, it would be more helpful if we can infer that a user searching for these items may be more interested in "cars with high chassis" or "off-road ability" rather than another Toyota SUV like "RAV4"—these concepts are rare in existing knowledge bases or taxonomies. Similarly, if an article talks about the movies "the Great Gatsby", "Wuthering Heights" and "Jane Eyre", it is also hard to infer that the article is actually about "book-to-film adaptations". The fundamental reason is that taxonomies such as DBPedia [Lehmann *et al.*, 2015] and Probase [Wu *et al.*, 2012], although maintaining structured knowledge about the world, are not designed to conceptualize from the *user's perspective* or to infer the user intention. Neither can they exhaust all the complex connections between different instances, concepts and topics that are discussed in different documents. Undoubtedly, the ability for machines to conceptualize just as a user would do—to extract trending and user-centered concept terms that are constantly evolving and are

expressed in user language—is critical to boosting the intelligence of recommender systems and search engines.

In this chapter, we propose *ConcepT*, a concept mining system at Tencent that aims to discover concepts at the right granularity conforming to user interests. Different from prior work, *ConcepT* is not based on mining web pages only, but mining from huge amounts of query logs and search click graphs, thus being able to understand user intention by capturing their interaction with the content. We present our design of *ConcepT* and our experience of deploying it in Tencent QQ Browser, which has the largest market share in Chinese mobile browser market with more than 110 millions daily active users. *ConcepT* serves as the core taxonomy system in Tencent QQ Browser to discover both time-invariant and trending concepts.

ConcepT can significantly boost the performance of both searching and content recommendation, through the taxonomy constructed from the discovered user-centered concepts as well as a concept tagging mechanism for both short queries and long documents that accurately depict user intention and document coverage. Up to today, *ConcepT* has extracted more than $200,000$ high-quality user-centered concepts from daily query logs and user click graphs in QQ Browser, while still growing at a rate of $11,000$ new concepts found per day. Although our system is implemented and deployed for processing Chinese query and documents, the proposed techniques in *ConcepT* can easily be adapted to other languages.

Mining user-centered concepts from query logs and search click graphs has brought about a number of new challenges. First, most existing taxonomy construction approaches such as Probase [Wu *et al.*, 2012] extract concepts based on Hearst patterns, like "such as", "especially", etc. However, Hearst patterns have limited extraction power, since high-quality patterns are often missing in short text like queries and informal user language. Moreover, existing methods extract concepts from web pages and documents that are usually written by experts in the *writer perspective*. However, search queries are often informal and may not observe the syntax of a written language [Hua *et al.*, 2015]. Hence, it is hard if not impossible to mine "user perspective" concepts based on predefined syntax patterns.

There are also many studies on keyphrase extraction [Shang *et al.*, 2018; Liu *et al.*, 2015; Mihalcea and Tarau, 2004]. They measure the importance or quality of all the N-grams in a document or text corpus, and choose keyphrases from them according to the calculated scores. As a result, such methods can only extract continuous text chunks, whereas a concept may be discontinuous or may not even be explicitly mentioned in a query or a document. Another concern is that most of such N-gram keyphrase extraction algorithms yield poor performance on short text snippets such

as queries. In addition, deep learning models, such as sequence-to-sequence, can also be used to generate or extract concepts. However, deep learning models usually rely on large amounts of high-quality training data. For user-centered concept mining, manually labeling such a dataset from scratch is extremely costly and time consuming.

Furthermore, many concepts in user queries are related to recent trending events whereas the concepts in existing taxonomies are mostly stable and time-invariant. A user may search for "Films for New Year (贺岁大片)" or "New Japanese Animation in April (四月新番)" in Tencent QQ Browser. The semantics of such concepts are evolving over time, since apparently we have different new animations or films in different years. Therefore, in contrast to existing taxonomies which mostly maintain long-term stable knowledge, it will be challenging yet beneficial if we can also extract time-varying concepts and dynamically update the taxonomies constructed.

We make the following novel contributions in the design of *ConcepT*:

First, we extract candidate user-centered concepts from vast query logs by two unsupervised strategies: 1) bootstrapping based on pattern-concept duality: a small number of predefined string patterns can be used to find new concepts while the found concepts can in turn be used to expand the pool of such patterns; 2) query-title alignment: an important concept in a query would repeat itself in the document title clicked by the user that has input the query.

Second, we further train a supervised sequence labeling Conditional Random Field (CRF) and a discriminator based on the initial seed concept set obtained, to generalize concept extraction and control the concept quality. These methods are complementary to each other and are best suited for different cases. Evaluation based on a labeled test dataset has shown that our proposed concept discovery procedure significantly outperforms a number of existing schemes.

Third, we propose effective strategies to tag documents with potentially complex concepts to depict document coverage, mainly by combining two methods: 1) matching key instances in a document with their concepts if their *isA* relationships exist in the corresponding constructed taxonomy; 2) using a probabilistic inference framework to estimate the probability of a concept provided that an instance is observed in its context. Note that the second method can handle the case when the concept words do not even appear in the document. For example, we may associate an article containing "celery", "whole wheat bread" and "tomato" with the concept "diet for weight loss" that a lot of users are interested in, even if the document does not have exact wording for "weight loss" but has context words such as "fibre", "healthy", and "hydrated".

Last but not least, we have constructed and maintained a three-layered *topic-*

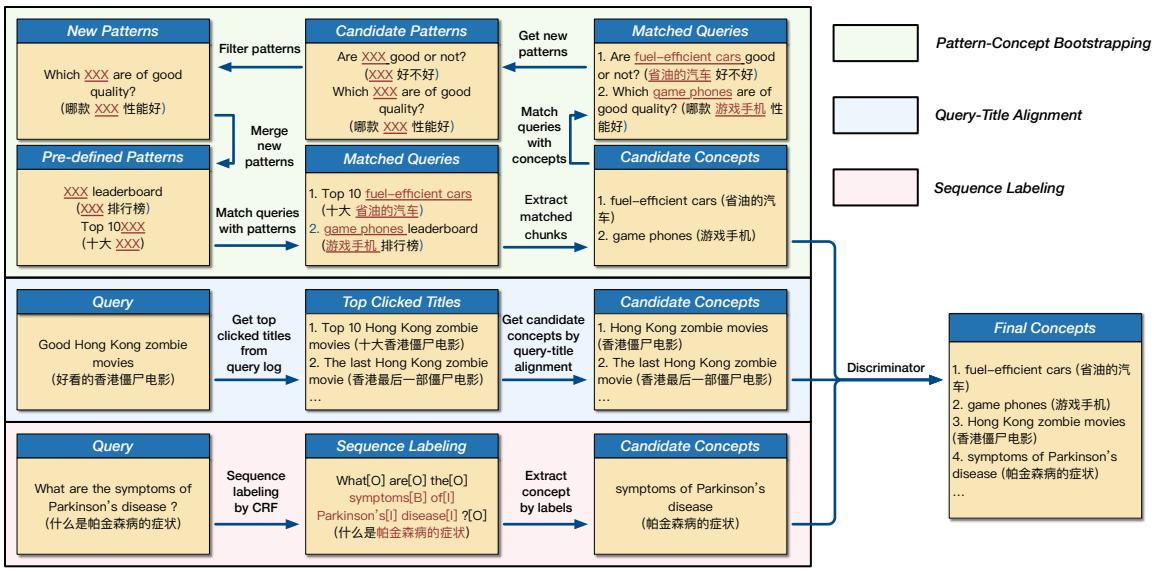


Figure 6.1: The overall process of concept mining from user queries and query logs.

concept-instance taxonomy, by identifying the *isA* relationships among instances, concepts and topics based on machine learning methods, including deep neural networks and probabilistic models. Such a user-centered taxonomy significantly helps with query and document understanding at varying granularities.

We have evaluated the performance of *ConcepT*, and observed that it can improve both searching and recommendation results, through both offline experiments and a large-scale online A/B test on more than 800,000 real users conducted in the QQ Browser mobile app. The experimental results reveal that our proposed methods can extract concepts more accurately from Internet user queries in contrast to a variety of existing approaches. Moreover, by performing query conceptualization based on the extracted concepts and the correspondingly constructed taxonomy, we can improve the results of search engine according to a pilot user experience study in our experiments. Finally, *ConcepT* also leads to a higher Impression Efficiency as well as user duration in the real world according to the large-scale online A/B test on the recommender system in feeds stream (text digest content recommended to users in a stream as they scroll down in the mobile app). The results suggest that the Impression Efficiency of the users increases by 6.01% when *ConcepT* system is incorporated for feeds stream recommendation.

6.2 User-Centered Concept Mining

Our objective of user-centered concept mining is to derive a word/phrase from a given user query which can best characterize this query and its related click logs at the proper granularity.

Denote a user query by $q = w_1^q w_2^q \cdots w_{|q|}^q$, which is a sequence of words. Let Q be the set of all queries. Denote a document title by $t = w_1^t w_2^t \cdots w_{|t|}^t$, another sequence of words. Given a user query q and its corresponding top-ranked clicked titles $T^q = \{t_1^q, t_2^q, \cdots, t_{|T^q|}^q\}$ from query logs, we aim to extract a concept phrase $\mathbf{c} = w_1^c w_2^c \cdots w_{|\mathbf{c}|}^c$ that represents the main semantics or the intention of the query. Each word $w_i^c \in \mathbf{c}$ belongs to either the query q or one of the corresponding clicked titles $t_j^q \in T^q$.

An overview of the detailed steps of user-centered concept mining from queries and query logs in *ConcepT* is shown in Fig. 6.1, which mainly consists of three approaches: pattern-concept bootstrapping, query-title alignment, as well as supervised sequence labeling. All the extracted concepts are further filtered by a discriminator. We utilize bootstrapping and query-title alignment to automatically accumulate an initial seed set of *query-concept* pairs, which can help to train sequence labeling and the discriminator, to extract a larger amount of concepts more accurately.

Bootstrapping by Pattern-Concept Duality. We first extract an initial set of seed concepts by applying the bootstrapping idea [Brin, 1998] only to the set of user queries Q (without the clicked titles). Bootstrapping exploits *Pattern-Concept Duality*, which is:

- Given a set of patterns, we can extract a set of concepts from queries following these patterns.
- Given a set of queries with extracted concepts, we can learn a set of patterns.

Fig. 6.1 (a) illustrates how bootstrapping is performed on queries Q. First, we manually define a small set of patterns which can be used to accurately extract concept phrases from queries with high confidence. For example, "Top 10 XXX (十大XXX)" is a pattern (with original Chinese expression in parenthesis) that can be used to extract seed concepts. Based on this pattern, we can extract concepts: "fuel-efficient cars (省油的汽车)" and "gaming phones (游戏手机)" from the queries "Top 10 fuel-efficient cars (十大省油的汽车)" and "Top 10 gaming phones (十大游戏手机)", respectively.

We can in turn retrieve more queries that contain these extracted concepts and derive new patterns from these queries. For example, a query "Which gaming phones

have the best performance? (哪款游戏手机性能好?)" also contains the concept "gaming phones (游戏手机)". Based on this query, a new pattern "Which XXX have the best performance? (哪款XXX性能好?)" is found.

We also need to shortlist and control the quality of the patterns found in each round. Intuitively speaking, a pattern is valuable if it can be used to accurately extract a portion of existing concepts as well as to discover new concepts from queries. However, if the pattern is too general and appears in a lot of queries, it may introduce noise. For example, a pattern "Is XXX good? (XXX好不好?)" underlies a lot of queries including "Is the fuel-efficient car good? (省油的车好不好?)" and "Is running everyday good (每天跑步好不好?)", whereas "running everyday (每天跑步)" does not serve as a sufficiently important concept in our system. Therefore, given a new pattern $\mathbf{p}$ found in a certain round, let n_s be the number of concepts in the existing seed concept set that can be extracted from query set Q by $\mathbf{p}$. Let n_e be the number of new concepts that can be extracted by $\mathbf{p}$ from Q. We will keep the pattern $\mathbf{p}$ if it satisfies: 1) $\alpha < \frac{n_s}{n_e} < \beta$, and 2) $n_s > \delta$, where $\alpha, \beta,$ and δ are predefined thresholds. (We set $\alpha = 0.6,\ \beta = 0.8,$ and $\delta = 2$ in our system.)

Concept mining by query-title alignment. Although bootstrapping helps to discover new patterns and concepts from the query set Q in an iterative manner, such a pattern-based method has limited extraction power. Since there are a limited number of high-quality syntax patterns in queries, the recall rate of concept extraction has been sacrificed for precision. Therefore, we further propose to extract concepts from both a query and its top clicked link titles in the query log.

The intuition is that a concept in a query will also be mentioned in the clicked titles associated with the query, yet possibly in a more detailed manner. For example, "The last Hong Kong zombie movie (香港|最后|一|部|僵尸|电影)" or "Hong Kong zombie comedy movie (香港|搞笑|僵尸|电影)" convey more specific concepts of the query "Hong Kong zombie movie (香港|僵尸|电影)" that leads to the click of these titles. Therefore, we propose to find such concepts based on the alignment of queries with their corresponding clicked titles. The steps are listed in the following:

1. Given a query q, we retrieve the top clicked titles $T^q = \{t_1^q, t_2^q, \cdots, t_{|T^q|}^q\}$ from the query logs of q, i.e., T^q consists of document titles that are clicked by users for more than N times during the past D days ($N = 5$ and $D = 30$ in our system).

2. For query q and each title $t \in T^q$, we enumerate all the N-grams in them.

3. Let N-gram $g_{in}^q = w_i^q w_{i+1}^q \cdots w_{i+n-1}^q$ denote a text chunk of length n starting from position i of query q, and

$g^t_{jm} = w^t_j w^t_{j+1} \cdots w^t_{j+m-1}$ denote a text chunk of length m starting from position j of title t. For each pair of such N-grams, $< g^q_{in}, g^t_{jm} >$, we identify g^t_{jm} as a candidate concept if: i) g^t_{jm} contains all the words of g^q_{in} in the same order; ii) $w^q_i = w^t_j$, and $w^q_{i+n-1} = w^t_{j+m-1}$.

Query-title alignment extends concept extraction from query set alone to concept discovery based on the query logs, thus incorporating some information of the user's interaction into the system.

Supervised sequence labeling. The above unsupervised methods are still limited in their generalizability. We further perform supervised learning and train a Conditional Random Field (CRF) to label the sequence of concept words in a query or a title, where the training dataset stems from the results of the bootstrapping and query-title alignment process mentioned above, combined with human reviews as detailed in the Appendix. Specifically, each word is represented by its tag features, e.g., Part-of-Speech or Named Entity Recognition tags, and the contextual features, e.g., the tag features of its previous word and succeeding word, the combination pattern of tags of contextual words and the word itself. These features are fed into a CRF to yield a sequence of labels, identifying the concept chunk, as shown in Fig. 6.1.

The above approaches for concept mining are complementary to each other. Our experience shows that CRF can better extract concepts from short text when they have clear boundary with surrounding non-concept words, e.g., "What cars are fuel-efficient (省油的汽车有哪些)". However, when the concept is split into multiple pieces, e.g., "What gifts should we prepare for birthdays of parents? (父母过生日准备什么礼物?)", the query-title alignment approach can better capture the concept that is scattered in a query.

A Discriminator for quality control. Given the concepts extracted by above various strategies, we need to evaluate their value. For example, in Fig. 6.1, the concept "The last Hong Kong zombie movie (香港|最后|一|部|僵尸|电影)" is too fine-grained and maybe only a small amount of users are interested in searching it. Therefore, we further train a classifier to determine whether each discovered concept is worth keeping.

We represent each candidate concept by a variety of its features such as whether this concept has ever appeared as a query, how many times it has been searched and so on (more details in Appendix). With these features serving as the input, we train a classifier, combining Gradient Boosting Decision Tree (GBDT) and Logistic Regression, to decide whether to accept the candidate concept in the final list or not. The training dataset for the discriminator is manually created. We manually check a found concept to see whether it is good (positive) or not sufficiently good (negative).

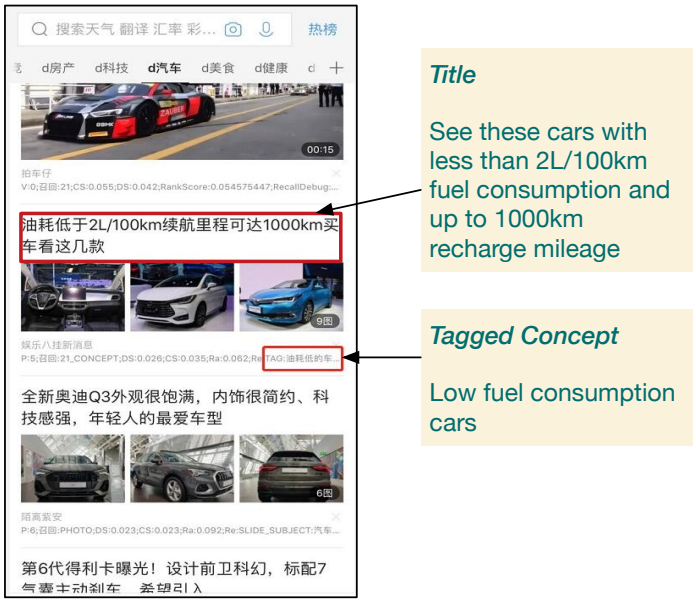


Figure 6.2: Example of concept tagging for documents in the feeds stream of Tencent QQ Browser.

Figure 6.3: The overall procedures of concept tagging for documents. We combine both a matching-based approach with a scoring-based approach to handle different situations.

Our experience reveals that we only need 300 samples to train such a discriminator. Therefore, the creation of the dataset incurs minimum overhead.

6.3 Document Tagging and Taxonomy Construction

In this section, we describe our strategies for tagging each document with pertinent user-centered concepts to depict its coverage. Based on document tagging, we further construct a 3-layered *topic-concept-instance* taxonomy which helps with feeds recommendation in Tencent QQ Browser.

6.3.1 Concept Tagging for Documents

While the extracted concepts can characterize the implicit intention of user queries, they can also be used to describe the main topics of a document. Fig. 6.2 shows an example of concept tagging in Tencent QQ Browser based on the *ConcepT* system. Suppose that a document titled "See these cars with less than 2L/100km fuel consumption and up to 1000km recharge mileage" can be tagged with the concept "low

fuel-consumption cars", even though the title never explicitly mentions these concept words. Such concept tags for documents, if available, can help improve search and recommendation performance. Therefore, we propose to perform concept tagging for documents.

Given a document d and a set of concepts $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_{|\mathbf{C}|}\}$, our problem is selecting a subset of concepts $\mathbf{C}^d = \{\mathbf{c}_1^d, \mathbf{c}_2^d, \cdots, \mathbf{c}_{|\mathbf{C}^d|}^d\}$ from $\mathbf{C}$ that are most related to the content of d. Fig. 6.3 presents the procedures of concept tagging for documents. To link appropriate concepts with a document, we propose a probabilistic inference-based approach, together with a matching-based method to handle different situations.

Specifically, our approach estimates the correlation between a concept and a document through the key instances contained in the document. When no direct *isA* relationship can be found between the key instances in the document and the concepts, we use probabilistic inference as a general approach to identify relevant concepts by utilizing the context information of the instances in the document. Otherwise, the matching-based approach retrieves candidate concepts which have *isA* relationships with the key instances in a taxonomy we have constructed (which be explained at the end of this section). After that, it scores the coherence between a concept and a document based on the title-enriched representation of the concept.

Key instance extraction. Fig. 6.3 shows our approach for key instance extraction. Firstly, we rank document words using GBRank [Zheng *et al.*, 2007] algorithm, based on word frequency, POS tag, NER tag, etc. Secondly, we represent each word by word vectors proposed in [Song *et al.*, 2018*b*], and construct a weighted undirected graph for top K words (we set $K = 10$). The edge weight is calculated by the cosine similarity of two word vectors. We then re-rank the keywords by TextRank [Mihalcea and Tarau, 2004] algorithm. Finally, we only keep keywords with ranking scores larger than δ_w (we use 0.5). From our experience, combining GBRank and word-vector-based TextRank helps to extract keywords that are more coherent to the topic of document.

Concept tagging by probabilistic inference. Denote the probability that concept $\mathbf{c}$ is related to document d as $p(\mathbf{c}|d)$. We propose to estimate it by:

$$p(\mathbf{c}|d) = \sum_{i=1}^{|E^d|} p(\mathbf{c}|\mathbf{e}_i^d) p(\mathbf{e}_i^d|d), \tag{6.1}$$

where E^d is the key instance set of d, and $p(\mathbf{e}_i^d|d)$ is the document frequency of instance $\mathbf{e}_i^d \in E^d$. $p(\mathbf{c}|\mathbf{e}_i^d)$ estimates the probability of concept $\mathbf{c}$ given $\mathbf{e}_i^d$. However, as the *isA* relationship between $\mathbf{e}_i^d$ and $\mathbf{c}$ may be missing, we further infer the conditional

probability by taking the contextual words of $\mathbf{e}_i^d$ into account:

$$p(\mathbf{c}|\mathbf{e}_i^d) = \sum_{j=1}^{|X_{E^d}|} p(\mathbf{c}|\mathbf{x}_j)p(\mathbf{x}_j|\mathbf{e}_i^d) \tag{6.2}$$

$p(\mathbf{x}_j|\mathbf{e}_i^d)$ is the co-occurrence probability of context word $\mathbf{x}_j$ with e_i^d. We consider two words as co-occurred if they are contained in the same sentence. X_{E^d} are the set of contextual words of $\mathbf{e}_i^d$ in d. Denote $\mathbf{C}^{\mathbf{x}_j}$ as the set of concepts containing $\mathbf{x}_j$ as a substring. $p(\mathbf{c}|\mathbf{x}_j)$ is defined as:

$$p(\mathbf{c}|\mathbf{x}_j) = \begin{cases} \frac{1}{|\mathbf{C}^{\mathbf{x}_j}|}. & \text{if } \mathbf{x}_j\text{is a substring of } \mathbf{c}, \\ 0 & \text{otherwise.} \end{cases} \tag{6.3}$$

For example, in Fig. 6.3, suppose $\mathbf{e}_i^d$ extracted from d is "Toyota RAV4 (丰田RAV4)". We may haven't establish any relationship between this instance and any concept. However, we can extract contextual words "fuel-efficient (省油)" and "durable (耐用)" from d. Based on these contextual words, we can retrieve candidate concepts that containing these words, such as "fuel-efficient cars (省油的汽车)" and "durable cellphones (耐用的手机)". We then estimate the probability of each candidate concept by above equations.

Concept tagging by matching. The probabilistic inference-based approach decomposes the correlation between a concept and a document through the key instances and their contextual words in the document. However, whenever the *isA* relationship between the key instances of d and $\mathbf{C}$ is available, we can utilize it to get candidate concepts directly, and calculate the matching score between each candidate concept and d to decide which concepts are coherent to the document.

First, we introduce how the *isA* relationship between *concept-instance* pairs can be identified. On one hand, given a concept, we retrieve queries/titles containing the same modifier in the context and extract the instances contained in the queries/titles. For example, given concept "fuel-efficient cars (省油的汽车)", we may retrieve a query/title "fuel-efficient Toyota RAV4 (省油的丰田RAV4)", and extract instance "Toyota RAV4 (丰田RAV4)" from the query/title, as it shares the same modifier "fuel-efficient (省油的)" with the given concept. After we getting a candidate instance $\mathbf{e}$, we estimate $p(\mathbf{c}|\mathbf{e})$ by Eqn. (6.2). On another hand, we can also extract *concept-instance* pairs from various semi-structured websites where a lot of *concept-instance* pairs are stored in web tables.

Second, we describe our matching-based approach for concept tagging. Let $E^d = \{\mathbf{e}_1^d, \mathbf{e}_2^d, \cdots, \mathbf{e}_{|E^d|}^d\}$ donate the set of key instances extracted from d, and $C^d = \{\mathbf{c}_1^d, \mathbf{c}_2^d, \cdots, \mathbf{c}_{|C^d|}^d\}$ donate the retrieved candidate concepts by the *isA* relationship of instances in E^d.

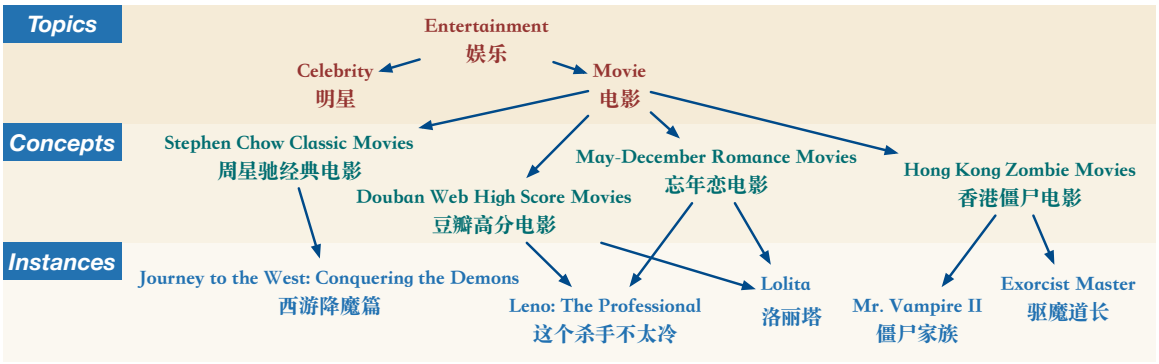


Figure 6.4: An example to show the extracted topic-concept-instance hierarchy.

For each candidate concept $\mathbf{c}_i^d$, we enrich its representation by concatenating the concept itself with the top N (we use 5) titles of user clicked links. We then represent enriched concept and the document title by TF-IDF vectors, and calculate the cosine similarity between them. If $sim(\mathbf{c}, d) > \delta_u$ (we set it as 0.58), we tag $\mathbf{c}$ to d; otherwise we reject it. Note that the *isA* relationship between *concept-instance* pairs and the enriched representation of concepts are all created in advance and stored in a database.

Fig. 6.3 shows an example of matching-based concept tagging. Suppose we extract key instance "Snow White (白雪公主)" from a document, we can retrieve related concepts "bed time stories (睡前故事)" and "fairy tales (童话故事)" based on *isA* relationship. The two concepts are further enriched by the concatenation of top clicked titles. Finally, we match candidate concepts with the original document, and keep highly related concepts.

6.3.2 Taxonomy Construction

We have also constructed a *topic-concept-instance* taxonomy based on the concepts extracted from queries and query logs. It can reveal the hierarchical relationship between different topics, concepts and instances. Currently our constructed taxonomy consists of 31 pre-defined topics, more than 200,000 user-centered concepts, and more than 600,000 instances. Among them, 40,000 concepts contain at least one instance, and 200,000 instances have been identified with a *isA* relationship with at least one concept. Based on the taxonomy, we can improve the user experience in search engines by understanding user implicit intention via query conceptualization, as well as enhance the recommendation performance by matching users and documents at different semantic granularities. We will demonstrate such improvements in detail in Sec. 6.4.

Fig. 6.4 shows a three-layered taxonomy that consists of topics, concepts and instances. The taxonomy is a Directed Acyclic Graph (DAG). Each node is either a topic, a concept or an instance. We predefined a list that contains $N_t = 31$ different topics, including entertainment, technology, society and so on. The directed edges indicate *isA* relationships between nodes.

We have already introduced our approach for *isA* relationship between *concept-instance* pairs. We need to further identify the relationship between *topic-concept* pairs. First, we represent each document as a vector through word embedding and pooling, and perform topic classification for documents through a carefully designed deep neural network (see Appendix for details). After that, given a concept $\mathbf{c}$ and a topic $\mathbf{p}$, suppose there are $n^{\mathbf{c}}$ documents that are correlated to concept $\mathbf{c}$, and among them there are $n^{\mathbf{c}}_{\mathbf{p}}$ documents that belong to topic $\mathbf{p}$. We then estimate $p(\mathbf{p}|\mathbf{c})$ by $p(\mathbf{p}|\mathbf{c}) = n^{\mathbf{c}}_{\mathbf{p}}/n^{\mathbf{c}}$. We identify the *isA* relationship between $\mathbf{c}$ and $\mathbf{p}$ if $p(\mathbf{p}|\mathbf{c}) > \delta_t$ (we set $\delta_t = 0.3$). Our experience shows that most of the concepts belong to one or two topics.

6.4 Evaluation

In this section, we first introduce a new dataset for the problem of concept mining from user queries and query logs, and compare our proposed approach with variety of baseline methods. We then evaluate the accuracy of the taxonomy constructed from extracted user-centered concepts, and show that it can improve search engine results by query rewriting. Finally, we run large-scale online A/B testing to show that the concept tagging on documents significantly improves the performance of recommendation in real world.

We deploy the *ConcepT* system which includes the capability of concept mining, tagging, and taxonomy construction in Tencent QQ Browser. For offline concept mining, our current system is able to extract around 27,000 concepts on a daily basis, where about 11,000 new concepts are new ones. For online concept tagging, our system processes 40 documents per second. More details about implementation and deployment can be found in appendix.

6.4.1 Evaluation of Concept Mining

The User-Centered Concept Mining Dataset (UCCM). As user-centered concept mining from queries is a relative new research problem and there is no public dataset available for evaluation, we created a large-scale dataset containing $10,000$

samples. Our *UCCM* dataset is sampled from the queries and query logs of Tencent QQ Broswer, from November 11, 2017 to July 1, 2018. For each query, we keep the document titles clicked by more than 2 users in previous day. Each sample consists of a query, the top clicked titles from real world query log, and a concept phrase labeled by 3 professional product managers in Tencent and 1 PhD student. We have published the *UCCM* dataset for research purposes [1].

Methodology and Compared Models. We evaluate our comprehensive concept mining approach with the following baseline methods and variants of our method:

- **TextRank [Mihalcea and Tarau, 2004]**. The classical graph-based ranking model for keyword extraction.[2]
- **THUCKE [Liu *et al.*, 2011]**. It regards keyphrase extraction as a problem of translation, and learns translation probabilities between the words in input text and the words in keyphrases.[3]
- **AutoPhrase [Shang *et al.*, 2018]**. A state-of-the-art quality phrase mining algorithm that extracts quality phrases based on knowledge base and POS-guided segmentation.[4]
- **Pattern-based matching with query (Q-Pattern)**. Extract concepts from queries based on patterns from bootstrapping.
- **Pattern-based matching with title (T-Pattern)**. Extract concepts from titles based on patterns from bootstrapping.
- **CRF-based sequence labeling with query (Q-CRF)**. Extract concepts from queries by CRF.
- **CRF-based sequence labeling with titles (T-CRF)**. Extract concepts from click titles by CRF.
- **Query-Title alignment (QT-Align)**. Extract concepts by query-title alignment strategy.

For the T-Pattern and T-CRF approach, as each click title will give a result, we select the most common one as the final result given a specific query. For the TextRank, THUCKE, and AutoPhrase algorithm, we take the concatenation of user query and

[1] https://github.com/BangLiu/ConcepT
[2] https://github.com/letiantian/TextRank4ZH
[3] https://github.com/thunlp/THUCKE
[4] https://github.com/shangjingbo1226/AutoPhrase

Method	Exact Match	F1 Score
TextRank	0.1941	0.7356
THUCKE	0.1909	0.7107
AutoPhrase	0.0725	0.4839
Q-Pattern	0.1537	0.3133
T-Pattern	0.2583	0.5046
Q-CRF	0.2631	0.7322
T-CRF	0.3937	0.7892
QT-Align	0.1684	0.3162
Our approach	**0.8121**	**0.9541**

Table 6.1: Compare different algorithms for concept mining.

click titles as input, and extract the top 5 keywords or phrases. We then keep the keywords/phrases contained in the query and concatenate them in the same order as in the query, then use it as the final result.

We use Exact Match (EM) and F1 to evaluate the performance. The exact match score is 1 if the prediction is exactly the same as groundtruth or 0 otherwise. F1 measures the portion of overlap tokens between the predicted phrase and the groundtruth concept.

Evaluation results and analysis. Table 6.1 compares our model with different baselines on the UCCM dataset in terms of Exact Match and F1 score. Results demonstrate that our method achieves the best EM and F1 score. This is because: first, the pattern-based concept mining with bootstrapping helps us to construct a collection of high-quality patterns which can accurately extract concepts from queries in an unsupervised manner. Second, the combination of sequence labeling by CRF and query-title alignment can recognize concepts from both queries and click titles under different situations, i.e., either the concept boundary in query is clear or not.

We can see the methods based on TextRank [Mihalcea and Tarau, 2004], THUCKE [Liu *et al.*, 2011] and AutoPhrase [Shang *et al.*, 2018] do not give satisfactory performance. That is because existing keyphrases extraction approaches are better suited for extracting keywords or phrases from a long document or a corpus. In contrast, our approach is specially designed for the problem of concept mining from user queries and click titles. Comparing our approach with its variants, including Q-Pattern, Q-CRF, T-CRF and QT-Align, we can see that each component cannot achieve comparable performance as ours independently. This demonstrates the effectiveness of combining different strategies in our system.

Metrics / Statistics	Value
Mean #Instances per Concept	3.44
Max #Instances per Concept	59
isA Relationship Accuracy	96.59%

Table 6.2: Evaluation results of constructed taxonomy.

6.4.2 Evaluation of Document Tagging and Taxonomy Construction

Evaluation of document tagging. For concept tagging on documents, our system currently processes around 96,700 documents per day, where about 35% of them can be tagged with at least one concept. We create a dataset containing $11,547$ documents with concept tags for parameter tuning, and we also open-source it for research purpose (see appendix for more details). We evaluate the performance of concept tagging based on this dataset. The result shows that the precision of concept tagging for documents is 96%. As the correlated concept phrases may even not show in the text, we do not evaluate the recall rate.

Evaluation of taxonomy construction. We randomly sample 1000 concepts from our taxonomy. As the relationships between *concept-instance* pairs are critical to query and document understanding, our experiment mainly focus on evaluating them. For each concept, we check whether the *isA* relationship between it and its instances is correct. We ask three human judges to evaluate them. For each concept, we record the number of correct instances and the number of incorrect ones.

Table 6.2 shows the evaluation results. The average number of instances for each concept is 3.44, and the maximum concept contains 59 instances. Note that the scale of our taxonomy is keep growing with more daily user queries and query logs. For the *isA* relationships between *concept-instance* pairs, the accuracy is 96.59%.

Table 6.3 shows a part of *topic-concept-instance* tuples from our taxonomy. We can see that the extracted concepts are expressed from "user perspective", such as "Female stars with a beautiful smile (笑容最美的女明星)" or "Mobile game for office workers (适合上班族玩的手游)". At the same time, the relationships between concepts and instances are also established based on user activities. For example, when a certain number of users click the documents related to "Sasaki Nozomi (佐佐木希)" when they are searching "Female stars with a beautiful smile (笑容最美的女明星)", our system will be able to recognize the *isA* relationship between the concept and the instance.

Topics	Concepts	Instances
Entertainment (娱乐)	Movies adapted from a novel (小说改编成的电影)	The Great Gatsby (了不起的盖茨比), Anna Karenina (安娜·卡列尼娜), Jane Eyre (简爱)
Entertainment (娱乐)	Female stars with a beautiful smile (笑容最美的女明星)	Ayase Haruka (绫濑遥), Sasaki Nozomi (佐佐木希), Dilraba (迪丽热巴)
Society (社会)	Belt and Road countries along the route (一带一路沿线国家)	Palestine (巴勒斯坦), Syria (叙利亚), Mongolia (蒙古), Oman (阿曼)
Games (游戏)	Mobile game for office workers (适合上班族玩的手游)	Pokemon (口袋妖怪), Invincible Asia (东方不败)

Table 6.3: Part of the *topic-concept-instance* samples created by *ConcepT* system.

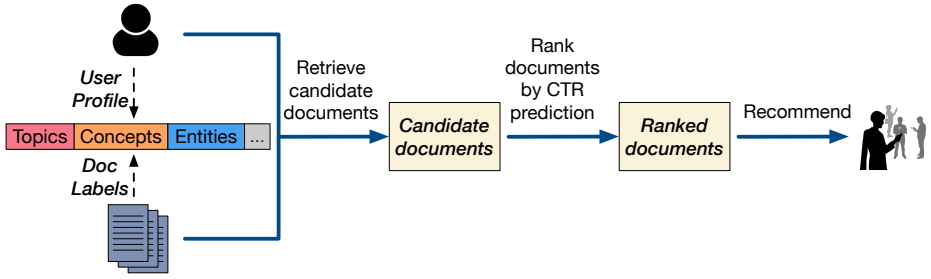


Figure 6.5: The framework of feeds recommendation in Tencent QQ Browser.

6.4.3 Online A/B Testing for Recommendation

We perform large-scale online A/B testing to show how concept tagging on documents helps with improving the performance of recommendation in real world applications. Fig. 6.5 illustrates the recommendation architecture based on our *ConcepT* system. In our system, both users and documents are tagged with interested or related topics, concepts and instances. We first retrieve candidate documents by matching users with documents, then we rank candidate documents by a Click-Through Rate (CTR) prediction model. The ranked documents are pushed to users in the feeds stream of Tencent QQ Browser.

For online A/B testing, we split users into buckets where each bucket contains $800,000$ of users. We first observe and record the activities of each bucket for 7 days based on the following metrics:

- **Impression Page View (IPV)**: number of pages that matched with users.

Metrics	Percentage Lift	Metrics	Percentage Lift
IPV	+0.69%	UCR	+0.04%
IUV	+0.06%	AUC	+0.21%
CPV	+0.38%	UD	**+0.83%**
CUV	+0.16%	IE	**+6.01%**

Table 6.4: Online A/B testing results.

- **Impression User View (IUV)**: number of users who has matched pages.
- **Click Page View (CPV)**: number of pages that users clicked.
- **Click User View (CUV)**: number of users who clicked pages.
- **User Conversion Rate (UCR)**: $\frac{CUV}{IUV}$.
- **Average User Consumption (AUC)**: $\frac{CPV}{CUV}$.
- **Users Duration (UD)**: average time users spend on a page.
- **Impression Efficiency (IE)**: $\frac{CPV}{IUV}$.

We then select two buckets with highly similar activities. For one bucket, we perform recommendation without the concept tags of documents. For another one, the concept tags of documents are utilized for recommendation. We run our A/B testing for 3 days and compare the result by above metrics. The Impression Efficiency (IE) and Users Duration (UD) are the two most critical metrics in real world application, because they show how many contents users read and how much time they spend on an application.

Table 6.4 shows the results of our online A/B testing. In the online experiment, we observe a statistically significant IE gain (6.01%) and user duration (0.83%). The page views and user views for click or impression, as well as user conversation rate and average user consumptions, are all improved. These observations prove that the concept tagging for documents greatly benefits the understanding of documents and helps to better match users with their potential interested documents. With the help of user-centered concepts, we can better capture the contained topics in a document even if it does not explicitly mention them. Given more matched documents, users spend more times and reading more articles in our feeds.

6.4.4 Offline User Study of Query Rewriting for Searching

Here we evaluate how user-centered concept mining can help with improving the results of search engines by query rewriting based on conceptualization. We create a evaluation dataset which contains 108 queries from Tencent QQ Browser. For each query $\mathbf{q}$, we analyze the concept $\mathbf{c}$ conveyed in the query, and rewrite the query by concatenating each of the instances $\{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_n\} \in \mathbf{c}$ with q. The rewritten queries are in the format of "$\mathbf{q}$ $\mathbf{e}_i$". For the original query, we collect the top 10 search results returned by Baidu search engine, the largest search engine in China. Assume we replace a query by K different instances. We collect top $\lceil \frac{10}{K} \rceil$ search results from Baidu for each of the rewritten queries, combining and keeping 10 of them as the search result after query rewriting.

We ask three human judges to evaluate the relevancy of the results. For each search reuslt, we record majority vote, i.e., "relevant" or "not relevant", of the human judges, and calculate the percentage of relevance of original queries and rewritten queries. Our evaluation results show that the percentage of relevant top 10 results increases from 73.1% to 85.1% after rewriting the queries with our strategy. The reason is that the concept mining for user queries helps to understand the intention of user queries, and concatenating the instances belonging to the concept with the original query provides the search engine more relevant and explicit keywords. Therefore, the search results will better match user's intention.

6.5 Information for Reproducibility

6.5.1 System Implementation and Deployment

We implement and deploy our *ConcepT* system in Tencent QQ Browser. The concept mining module and taxonomy construction module are implemented in Python 2.7, and they run as offline components. For document tagging module, it is implemented in C++ and runs as an online service. We utilize MySQL for data storage.

In our system, each component works as a service and is deployed on Tencent Total Application Framework (Tencent TAF)[5]. Tencent TAF is a high-performance remote procedure call (RPC) framework based on name service and Tars protocol, it also integrate administration platform, and implement hosting-service via flexible schedule. It has been used in Tencent since 2008, and supports different programming languages. For online document concept tagging, it is running on 50 dockers. Each docker is configured with six 2.5 GHz Intel Xeon Gold 6133 CPU cores and 6 GB

[5]https://github.com/TarsCloud/Tars

memory. For offline concept mining and taxonomy construction, they are running on 2 dockers with the same configuration.

ALGORITHM 2: Offline concept mining process.

Data: Queries and query logs in a day
Result: Concepts

```
Check whether successfully obtained the queries and logs;
if succeed then
    Perform concept mining by our proposed approach;
else
    Break;
```

ALGORITHM 3: Offline *isA* relationship discovery between concepts and instances.

Data: News documents, the vocabulary of instances, concepts, and the index between key terms and concepts
Result: *isA* relationship between concepts and instances

```
for each document do
    Get the instances in the document based on the vocabulary;
    for each instance do
        Get the intersection of concept key terms and the terms co-occurred in the same
          sentence with document instances;
        Get related concepts that containing at least one key term in the intersection;
        Get <instance, key terms, concepts> tuples based on the results of above steps;
Get the co-occurrence features listed in Table 6.5, and classify whether existing isA
  relationship between the instances and candidate concepts.
```

Algorithm 1-4 show the running processes of each component in ConcepT. For offline concept mining from queries and search logs, the component is running on a daily basis. It extracts around 27,000 concepts from 25 millions of query logs everyday, and about 11,000 of the extracted concepts are new. For offline relationship discovery in taxonomy construction, the component runs every two weeks. For online concept tagging for documents, the processing speed is 40 documents per second. It performs concept tagging for about 96,700 documents per day, where about 35% of them can be tagged with at least one concept.

6.5.2 Parameter Settings and Training Process

We have described the threshold parameters in our work. Here we introduce the features we use for different components in our system, and describe how we train each component. Table 6.5 lists the input features we use for different sub-modules in our *ConcepT* system.

ALGORITHM 4: Online probabilistic inference-based concept tagging for documents.

Data: News documents, *isA* relationship between instances and concepts
Result: Documents with concept tags

```
for each document do
    Perform word segmentation;
    Extract key instances by the approach described in Fig. 6.3;
    Get candidate concepts by the isA relationship between concepts and key instances;
    for each concept do
        Calculate the coherence between the candidate concept and the document by the
          probabilistic inference-based approach;
        Tag the concept to the document if the coherence is above a threshold;
```

ALGORITHM 5: Online matching-based concept tagging for documents.

Data: News documents
Result: Documents with concept tags

```
for each document do
    Perform word segmentation;
    Extract key terms by TF-IDF;
    Get candidate concepts containing above key terms;
    Get the title-enriched representation of candidate concepts;
    Represent document and each candidate concept by TF-IDF vector;
    for each concept do
        Calculate cosine similarity between the candidate concept and the document;
        Tag the concept to the document if the similarity is above a threshold;
```

Training process. For concept mining, we randomly sample 15,000 query search logs in Tencent QQ Browser within one month. We extract concepts for these query logs using approaches introduced in Sec. 6.2, and the results are manually checked by Tencent product managers. The resulting dataset is used to train the classifier in query-title alignment-based concept mining, and the Conditional Random Field in our model. We utilize CRF++ v0.58 to train our model. 80% of the dataset is used as training set, 10% as development set and the remaining 10% as test set.

For concept tagging, we randomly sample 10,000 news articles from the feeds stream of Tencent QQ browser during a three-month period, where each topic contains abut 800 to 1000 articles. We iteratively perform concept tagging for documents based on the approaches described in Sec. 6.3. After each iteration, we manually check whether the tagged concepts are correlated or not. Then we update our dataset and retrain the models of concept tagging. The iteration process is topped until no more new concepts can be tagged to documents. The resulting dataset is used to train the classifiers and set the hyper-parameters in concept tagging. We use 80% of the

Task	Features
Document key instance extraction	Whether the topic of instance is the same with the topic of document; whether it is the same with the instance in title; whether the title contains the instance topic; the frequency of the instance among all instances in the document; the percentage of sentences containing the instance.
Classify whether a short text is a concept	Whether the short text ever shown as a user query; how many times it has been searched; Bag-of-Word representation of the text; the topic distribution of user clicked documents given that short text as query.
Train CRF for concept mining from query	word, NER, POS, <previous word, word>, <previous word, next word>, <previous POS, POS>, <POS, next POS>, <previous POS, word>, <word, next POS>.

Table 6.5: The features we use for different tasks in ConcepT.

dataset as training set, 10% as development set and the remaining 10% as test set.

6.5.3 Publish Our Datasets

We have published our datasets for research purpose and they can be accessed from `https://github.com/BangLiu/ConcepT`. Specifically, we open source the following datasets:

- **The UCCM dataset**. It is used to evaluate the performance of our approach for concept mining and it contains $10,000$ samples.
- **The document tagging dataset**. It is used to evaluate the document tagging accuracy of ConcepT, and it contains 11,547 documents with concept tags.
- **Topic-concept-instance taxonomy**. It contains 1000 *topic-concept-instance* samples from our constructed taxonomy.
- **The seed concept patterns for bootstrapping-based concept mining**. It contains the seed string patterns we utilized for bootstrapping-based concept mining from queries.
- **Pre-defined topic list**. It contains our 31 pre-defined topics for taxonomy construction.

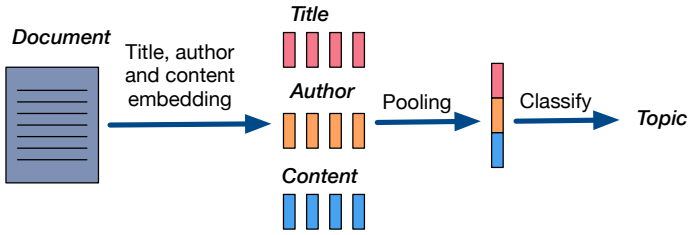


Figure 6.6: Document topic classification.

Query	Concept
What are the Qianjiang specialties (黔江的特产有哪些)	Qianjiang specialties (黔江特产)
Collection of noodle snacks cooking methods (面条小吃的做法大全)	noodle snacks cooking methods (面条小吃的做法)
Which cars are cheap and fuel-efficient? (有什么便宜省油的车)	cheap and fuel-efficient cars (便宜省油的车)
Jiyuan famous snacks (济源有名的小吃)	Jiyuan snacks (济源小吃)
What are the symptoms of depression? (抑郁症有什么症状)	symptoms of depression (抑郁症症状)
Large-scale games of military theme (军事题材的大型游戏)	Military games (军事游戏)

Table 6.6: Examples of queries and the extracted concepts given by ConcepT.

6.5.4 Details about Document Topic Classification

Topic classification aims to classify a document d into our predefined N_t (it is 31 in our system) topic categories, including entertainment, events, technology and so forth. Fig. 6.6 illustrates our model for document topic classification. We represent the title, author, and content of document d by word vectors. Then we apply max pooling to title and author embeddings, and mean pooling to content embeddings. The results of pooling operations are concatenated into a fix-length vector representation of d. We then classify it by a feed forward neural network. The accuracy of our model is 95% on a labeled dataset containing 35,000 news articles.

6.5.5 Examples of Queries and Extracted Concepts

Table 6.6 lists some examples of user queries, together with the concepts extracted by ConcepT. We can see that the concepts are appropriate to summarize the core user intention in queries.

6.6 Conclusion

In this chapter, we describe our experience of implementing *ConcepT*, a user-centered concept mining and tagging system at Tencent that designed to improve the understanding of both queries and long documents. Our system extracts user-centered concepts from a large amount of user queries and query logs, as well as performs concept tagging on documents to characterize the coverage of documents from user-perspective. In addition, *ConcepT* further identifies the *isA* relationship between concepts, instances and topics to constructs a 3-layered *topic-concept-instance* taxonomy. We conduct extensive performance evaluation through both offline experiments and online large-scale A/B testing in the QQ Browser mobile application on more than $800,000$ real users. The results show that our system can extract featured, user-centered concepts accurately from user queries and query logs, and it is quite helpful for both search engines and recommendation systems. For search engines, the pilot user study in our experiments shows that we improve the results of search engine by query conceptualization. For recommendation, according to the real-world large-scale online A/B testing, the Impression Efficiency improves by 6.01% when incorporating *ConcepT* system for feeds recommendation in Tencent QQ Browser.

Chapter 7

Scalable Creation of a Web-scale Ontology

Understanding what online users may pay attention to on the web is key to content recommendation and search services. These services will benefit from a highly structured and web-scale ontology of entities, concepts, events, topics and categories. While existing knowledge bases and taxonomies embody a large volume of entities and categories, we argue that they fail to discover properly grained concepts, events and topics in the language style of online users. Neither is a logically structured ontology maintained among these notions. In this chapter, we present *GIANT*, a mechanism to construct a user-centered, web-scale, structured ontology, containing a large number of natural language phrases conforming to user attentions at various granularities, mined from the vast volume of web documents and search click logs. Various types of edges are also constructed to maintain a hierarchy in the ontology. We present our detailed techniques used in GIANT, and evaluate the proposed models and methods as compared to a variety of baselines, as well as deploy the resulted Attention Ontology in real-world applications, involving over a billion users, to observe its effect on content recommendation. The online performance of the ontology built by GIANT proves that it can significantly improve the click-through rate in news feeds recommendation.

7.1 Introduction

In a fast-paced modern society, people have to carefully choose what they pay attention to in their overstimulated daily lives. The vast and diverse information on the Internet makes it an ever-increasing challenge for services to increase the attention span of online users. A variety of recommendation services [Konstan, 2008; Ado-

mavicius and Tuzhilin, 2005; Koutrika, 2018; Bobadilla *et al.*, 2013; Adomavicius and Tuzhilin, 2011] have been built to find or recommend relevant information to users based on their search queries or article viewing histories. These services could potentially benefit from a large, web-based, structured, and reusable ontology created from unstructured Internet documents, with a particular focus on points that users pay attention to.

Content recommendation suffers from two long-standing problems: *inaccurate recommendation* and *monotonous recommendation*. Inaccurate recommendation is largely due to a lack of an ontology containing terms that can describe user interests at a proper granularity. Most current news recommendation services rely on keyword-based content matching. For example, if a user reads articles about "Theresa May's resignation speech", recommender systems may use the entities "Theresa May", "Speech" and "Resignation Speech" to further retrieve articles related to these keywords. However, the user is probably truly concerned with the topic "Brexit Negotiation". Entities are often too fine-grained and categories are too coarse-grained to provide a suitable resolution to represent underlying user intents. On the other hand, monotonous recommendation, which means that users may receive pushes about the same entities or events, is rooted in the incapability of existing systems to extrapolate beyond the visited keywords. For example, when an article contains the words "brown rice", "treadmill" and "celery juice", it is most likely discussing "weight loss recipe" or "healthy recipe". Categorizing "brown rice" into food and "treadmill" into may not help recommendation the best. We argue that what is required is a web-scale ontology that can abstract entities (keywords) into concepts, events, and topics in the open domain, while maintaining a structured hierarchy among them to facilitate extrapolation and inference of user interests.

However, mining a large volume of user-centered concepts, events or topics from the web is a new and challenging task that has not been addressed by prior literature. For concept mining, most existing taxonomy or knowledge bases, such as Probase [Wu *et al.*, 2012], DBPedia [Lehmann *et al.*, 2015], CN-DBPedia [Xu *et al.*, 2017], YAGO [Suchanek *et al.*, 2007], extract concepts/entities from Wikipedia and web pages based on Hearst patterns, via "such as", "especially", "including" etc. However, concepts that can be extracted by Hearst patterns are clearly limited. Moreover, web pages written in the author's perspective, such as Wikipedia, are not best at tracking phrases from the user's perspective, such as "top-5 restaurants for families". For event mining, most existing approaches [Aone and Ramos-Santacruz, 2000; Miwa *et al.*, 2010; McClosky *et al.*, 2011; Yang and Mitchell, 2016] rely on the ACE definition of events [Doddington *et al.*, 2004; Grishman *et al.*, 2005] and perform closed-domain

event extraction via supervised learning, which is not scalable to diverse types of events in the open domain. There are also works attempting to extract events from social networks such as Twitter [Watanabe *et al.*, 2011; Ritter *et al.*, 2012; Atefeh and Khreich, 2015; Cordeiro and Gama, 2016]. But they represent events by clusters of keywords or phrases, without identifying a clear hierarchy or structured ontology.

In this chapter, we propose **GIANT**, a web-based structured ontology construction mechanism that aims to automatically discover high-quality natural language phrases, which we call user *attentions*, at the right granularity conforming to user interests, as well as to maintain a hierarchy of them, by mining vast amounts of unstructured web documents and search query logs. GIANT produces and maintains a web-scale ontology, named the Attention Ontology (AO), which consists of around 2 million different kinds of nodes (i.e., categories, concepts, entities, topics, and events), while still growing. In addition to categories (e.g., cars, technology) and fine-grained entities (e.g., iPhone XS, Honda Civic) that appear in existing taxonomies, Attention Ontology also contains abstractive nodes at a medium granularity, including newly discovered *concepts*, *events* and *topics*, in terms of the language of web users instead of authors. By modeling documents with abstractive terms that users would actually pay attention to, the Attention Ontology proves to be effective in greatly improving content recommendation in the open domain. For example, with the ontology, the system can infer a user's interest on the concept "economy cars" if he/she browsed articles about "Honda Civic", even if the exact wording of economy cars does not appear in the article. The system can also extrapolate that a user cares about all events related to the topic "Brexit Negotiation" if he/she read an article about "Theresa May's resignation".

We present our design of GIANT and our experience of deploying it in several anonymous real world applications, involving more than one billion active users all around the globe. GIANT currently serves as the core ontology or taxonomy construction system in these commercial applications for discovering long-term and trending user attentions. In designing GIANT, we have made the following novel contributions:

First, instead of constructing an ontology based on Wikipedia or other author-centered documents, we propose a framework to construct it from the *click graph*, a large bipartite graph of search queries and their corresponding clicked documents, which contains a plethora of phrases in the language style of web users as well as information about user behavior. In addition, as compared to Wikipedia and other existing knowledge bases, which contain general knowledge about the world, queries from users can reflect hot events or topics at the present time. GIANT relies on the linkage from queries to the clicked documents to discover key phrases that represent

user attentions as well as to construct a hierarchy out of them. As an intuitive toy example, if a query "top-5 family restaurants" frequently leads to the click-through of a certain restaurant, we can recognize "top-5 family restaurants" as a concept and the restaurant is an entity of the concept.

To extract a large amount of key phrases that represent user attentions from the click graph, we propose GCTSP-Net, a general multi-task neural network model which can extract different types of phrases (including concepts, events, and topics) at scale. We first introduce a graphical representation, named Query Title Interaction Graph (QTIG), to represent information in a cluster of queries and their clicked document titles as a graph, with nodes being keywords and edges between nodes being how they are traversed in these queries and titles. Then we employ a graph-to-sequence neural network, i.e., the Relational Graph Convolutional Network (R-GCN) [Schlichtkrull *et al.*, 2018], to encode the QTIG for each query cluster, and generate a natural language phrase as an abstractive description of the cluster, by node classification and asymmetric traveling salesman decoding (ATSP-decoding). The extracted phrases become the nodes in the Attention Ontology.

To maintain a hierarchy and structure for the ontology, we propose various approaches to identify the edges between nodes in the Attention Ontology and tag each edge with one of the three types of relationships, i.e., *isA*, *involve*, and *correlate*. Multiple ad-hoc and probabilistic methods are proposed to identify different types of edges. The GCTSP-Net is also used to identify the *involve* relationship, that is to recognize the key arguments in an events (i.e., triggers, entities and locations). The constructed edges can benefit a variety of applications. For instance, we can tag users and documents with abstractive nodes in the ontology to depict document's coverage by extrapolating beyond its verbatim language. We can also perform query conceptualization based on the ontology to increase the diversity of retrieved results. Besides, the correlation between events can leveraged to discover an evolving story, which enables the tracking of an event series and keep the interested users updated.

Last but not least, we introduce efficient strategies to quickly build the training data necessary for GIANT to perform phrase mining and relation identification, with minimum human efforts. For phrase mining, we combine a bootstrapping strategy based on pattern-phrase duality [Brin, 1998; Liu *et al.*, 2019*c*] and text alignment based on query-title conformity. For relationship identification, we utilize the co-occurrence of different phrases in queries, documents, and consecutive queries to extract samples of phrase pairs with target relationships. These labeled examples automatically mined from the click graph are then used to train the proposed machine learning models to generalize and scale up in different sub-tasks.

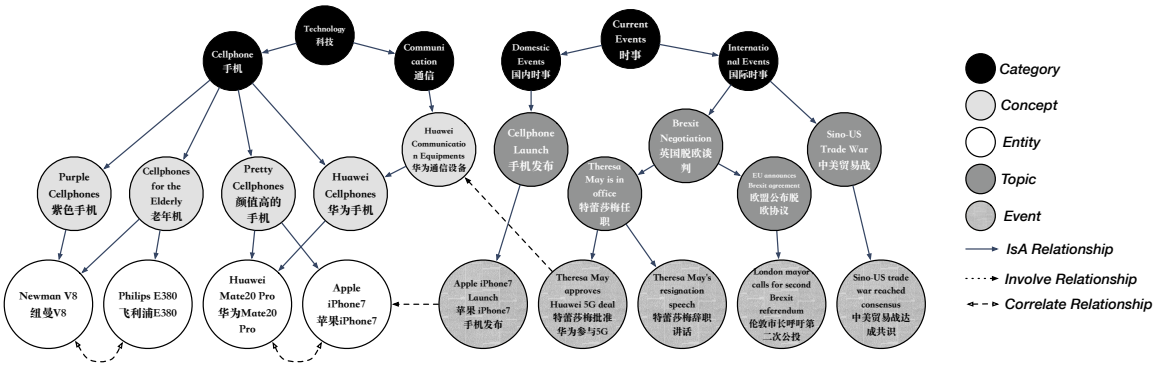


Figure 7.1: An example to illustrate our *Attention Ontology* (AO) for user-centered text understanding.

We have performed extensive evaluation of GIANT. The assessment of the constructed ontology shows that our system can extract a plethora of high-quality phrases from search logs, with a large amount of accurately identified relationships between these phrases, leading to a highly structured web-scale ontology system. We compare our proposed GCTSP-Net model with a variety of baseline approaches to evaluate its performance and superiority on multiple tasks. The experimental results show that our approach can extract heterogeneous phrases more accurately from the click graph as compared to existing methods. Finally, we have deployed the resulted Attention Ontology constructed by GIANT in multiple real-world anonymous applications that serve more than one billion users in total. We evaluate the effect of introducing Attention Ontology in an application that involves a news feed stream component that pushes news articles to mobile users as they scroll down their screen on their mobile devices. The results suggest that GIANT can significantly improve the click-through rate in news feeds recommendation.

The rest of the chapter is organized as follows: Section 7.2 provides an overview to the framework of GIANT system, as well as illustrates the Attention Graph constructed by GIANT. Section 7.3 introduce our proposed techniques for extracting the nodes and constructing the edges in Attention Graph, as well as our strategies for fast dataset creation. In Section 7.4, we show how to apply the constructed Attention Graph to user interests modeling and text understanding. Section 7.6 concludes the chapter.

7.2 The Attention Ontology

We provide an overview of the *Attention Ontology* (AO) to be constructed for web-scale text understanding, with a focus on mining the terms that grab online users'

attention and connections among them. Online users often pay attention to specific entities or events of their interests. Existing knowledge bases and taxonomies, e.g., Probase [Wu *et al.*, 2012] and DBPedia [Lehmann *et al.*, 2015], contain general knowledge about the world yet often do not incorporate terms, e.g., weight loss recipes, best family SUVs, that online users actually spend their time on.

In the proposed Attention Ontology, each node is represented by a phrase or a collection of phrases of the same meaning mined from Internet. We use the term "attention" as a general name for *entities*, *concepts*, *events*, *topics*, and *categories*, which represent five types of information that can capture an online user's attention at different semantic granularities. Such attention phrases can be utilized to conceptualize user interests and depict document coverages. For instance, if a user wants to buy a family vehicle for road trips, he/she may input such a query "vehicles choices for family road trip". From this query, we could extract the concept, "family road trip vehicles", and tag it to matching entities such as "Honda Odyssey Minivan" or "Ford Edge SUV". We could then recommend articles related to these Honda and Ford vehicles, even if they do not contain the exact wording of "family road trip". In essence, the Attention Ontology enables us to achieve a user-centered understanding of web documents and queries, which improves the performance of search engines and recommender systems.

Figure 7.1 shows an illustration of the Attention Ontology, which is in the form of a Directed Acyclic Graph (DAG). Specifically, there are five types of nodes:

- **Category**: a category node defines a broad field that encapsulates many related topics or concepts. For example, technology, current events, entertainment, sports, finance and so forth. In our system, we pre-define a 3-level categories hierarchy, which consists of 1,206 different categories.

- **Concept**: a concept is a group of things that share some common attributes [Liu *et al.*, 2019*c*; Wu *et al.*, 2012], such as superheroes, MOBA games, fuel-efficient cars and so on. In contrast with coarse-grained categories and fine-grained entities, concepts can help better characterize users' interests at a suitable semantic granularity.

- **Entity**: an entity is a specific instance belonging to one or multiple concepts. For example, Iron Man is an entity belonging to the concepts "superheroes" and "Marvel superheroes".

- **Event**: an event is a real-world incident that involves a group of specific persons, organizations, or entities. It is also tagged with a certain time/location of

occurrence [Liu *et al.*, 2017]. A hot event usually is reported by a set of news articles with different wordings, and potentially from various perspectives. In our work, we represent an event with four types of attributes: *entities* (indicating who or what are involved in the event), *triggers* (indicating what kind/type of event it is), *time* (indicating when the event takes place), and *location* (indicating where the event takes place).

- **Topic**: a topic is broader than an event. A topic represents a collection of events sharing some common attributes. For example, both "Samsung Galaxy Note 7 Explosion in China" and "iPhone 6 Explosion in California" events belong to the topic "Cellphone Explosion". Similarly, events such as "Theresa May is in Office", "Theresa May's Resignation Speech" can be covered by the topic "Brexit Negotiation".

Furthermore, we define three types of edges, i.e., relationships, between nodes:

- **isA relationship**, indicating that the destination node is an instance of the source node. For example, the entity "Huawei Mate20 Pro" *isAn* instance of "Huawei Cellphones".
- **involve relationship**, indicating that the destination node is involved in an event/topic described by the source node.
- **correlate relationship**, indicating two nodes are highly correlated with each other.

The edges in the Attention Ontology reveal the types of relationships and the degrees of relatedness between nodes. A plethora of edges enables the inference of more hidden interests of a user beyond the content he/she has browsed by moving along the edges on the Attention Ontology and recommending other related nodes at a coarser or finer granularity based on the nodes the user has visited. Furthermore, by analyzing edges between event nodes, we could also keep track of a *developing story*, which usually consists of a series of events, and keep interested users updated.

7.3 Ontology Construction

GIANT is a mechanism to discover phrases that users pay attention to from the web as well as to build a structured hierarchy out of them. In this section, we present our detailed techniques to construct the Attention Ontology based on neural networks and other ad-hoc methods. The entire process consists of two phases: i) user attention

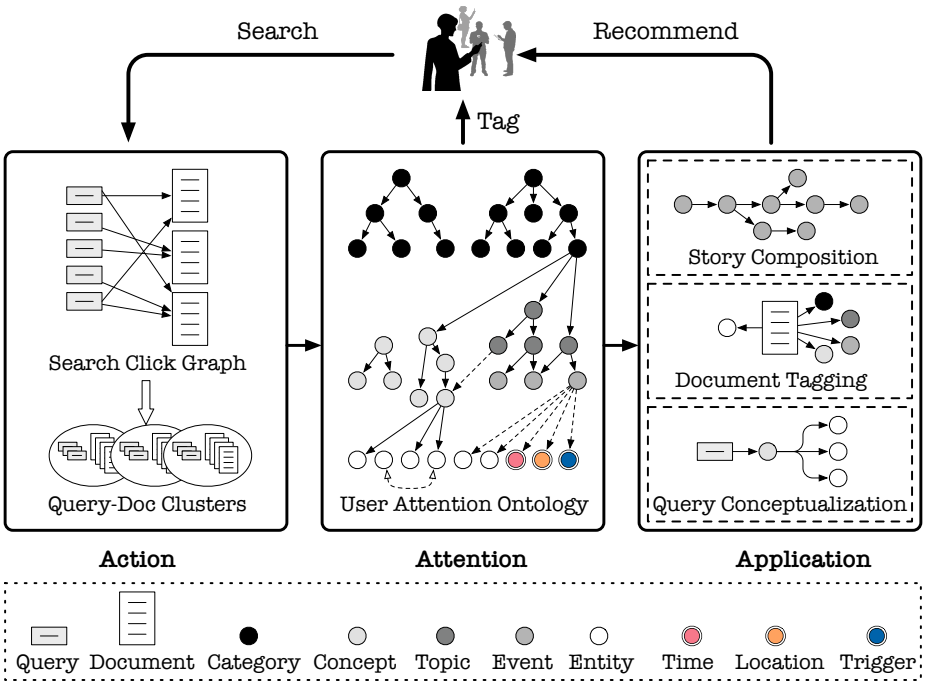


Figure 7.2: Overview of our framework for constructing the Attention Ontology and performing different tasks.

phrases mining, and ii) attention phrases linking. In the first phase, we define and extract user attention phrases in different semantic granularities from a large-scale search click graph. In the second phase, we link different extracted nodes and identify their relationships to construct a structured ontology.

Figure 7.2 shows the overall framework of GIANT, which constructs the Attention Ontology based on user search and click logs. The framework consists of three major components: action, attention, and application. In the action component, when users perform different actions (e.g., search, click, etc.), the user queries and clicked documents form a bipartite graph [Wikipedia contributors, 2019], commonly known as a search click graph. Based on it, we can collect highly correlated queries and documents by aggregating documents that correspond to a query or vice versa, into query-doc clusters. In the attention component, we can extract different types of nodes (e.g., concepts, events, topics, entities, etc.) from the query-doc clusters, as well as learn the relationships between different nodes to form the Attention Ontology. In the application component, we can apply the Attention Ontology to different applications such as query conceptualization and document tagging. On the other hand, we can also integrate different nodes to user profiles to characterize the interest of different users based on his/her historical viewing behavior. In this manner, we can characterize both users and web documents based on the constructed Attention

Ontology, which enables us to better understand and recommend documents from users' perspective.

7.3.1 Mining User Attentions

We propose a novel algorithm to extract various types of attention phrases (e.g., concepts, topics or events), which represent user attentions or interests, from a collection of queries and document titles.

Problem Definition. Suppose a bipartite search click graph $G_{sc} = (Q, D, E)$ records the click-through information over a set of queries $Q = \{q_1, q_2, \cdots, q_{|Q|}\}$ and documents $D = \{d_1, d_2, \cdots, d_{|D|}\}$. We use $|*|$ to denote the length of $*$. E is a set of edges linking queries and documents. Our objective is to extract a set of phrases $P = \{p_1, p_2, \cdots, p_{|P|}\}$ from Q and D to represent user interests. Specifically, suppose p consists of a sequence of words $\{w_{p1}, w_{p2}, \cdots, w_{p|p|}\}$. In our work, each phrase p is extracted from a subset of queries and the titles of correlated documents, and each word $w_p \in p$ is contained by at least one query or title.

ALGORITHM 6: Mining Attention Nodes

Input: a sequence of queries $Q = \{q_1, q_2, \cdots, q_{|Q|}\}$, search click graph G_{sc}.
Output: Attention phrases $P = \{p_1, p_2, \cdots, p_{|P|}\}$.

```
calculating the transport probabilities between connected query-doc pairs according to Equation (7.1) and (7.2);
for each q ∈ Q do
    run random walk to get ordered related queries Q_q and documents D_q;
end for
P = {};
for each input cluster (Q_q, D_q) do
    get document titles T_q from D_q;
    create Query-Title Interaction Graph G_qt(Q_q, T_q);
    classify the nodes in G_qt(Q_q, T_q) by R-GCN to learn which nodes belong to the output phrase;
    sort the extracted nodes by ATSP-decoding and concatenate them into an attention phrase p_q^a;
    perform attention normalization to merge p_q^a with its similar phrase in P into a sublist;
    if p_q^a is not similar to any existing phrase, append p_q^a to P;
end for
derive higher-level attention phrases by performing common suffix discovery over P, and append the new phrases into P;
create a node in the Attention Ontology for each phrase or sublist of similar phrases.
```

Algorithm 6 presents the pipeline of our system to extract attention phrases based on a bipartite search click graph. In what follows, we introduce each step in detail.

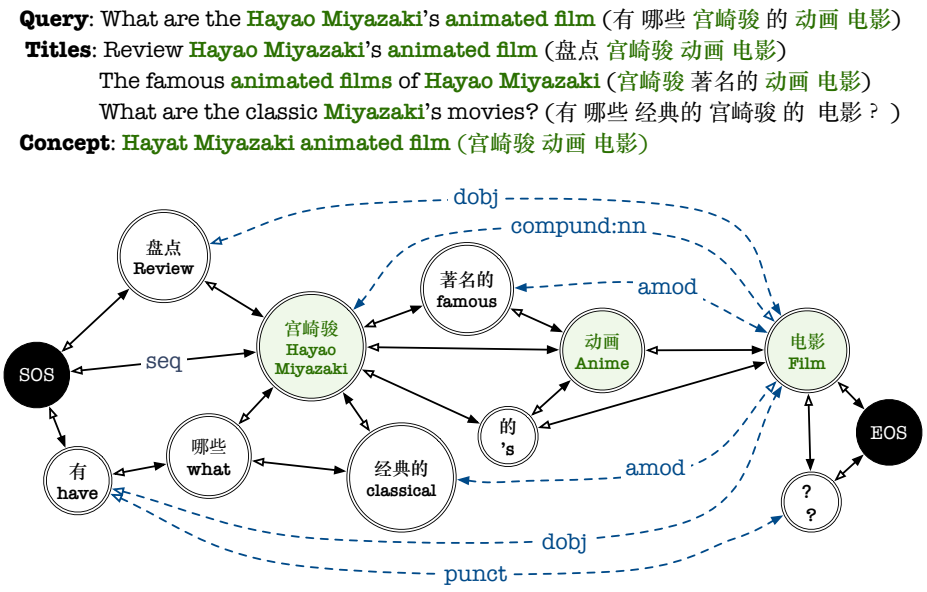


Figure 7.3: An example to show the construction of query-title interaction graph for attention mining.

Query-Doc Clustering. Suppose $c(q_i, d_j)$ represents how many times query q_i is linked to document d_j in a search click graph G_{sc} constructed from user search click logs within a period. For each query-doc pair $< q_i, d_j >$, suppose $N(q_i)$ denotes the set of documents connected with q_i, and $N(d_j)$ denotes the set of queries connected with d_j. Then we define the transport probabilities between q_i and d_j as:

$$\mathbb{P}(d_j|q_i) = \frac{c(q_i, d_j)}{\sum_{d_k \in N(q_i)} c(q_i, d_k)}, \tag{7.1}$$

$$\mathbb{P}(q_i|d_j) = \frac{c(q_i, d_j)}{\sum_{q_k \in N(d_j)} c(q_k, d_j)}. \tag{7.2}$$

From query q, we perform random walk [Spitzer, 2013] according to transport probabilities calculated above and compute the weights of visited queries and documents. For each visited query or document, we keep it if its visiting probability is above a threshold δ_v and the number of non-stop words in q is more than a half. In this way, we can derive a cluster of correlated queries Q_q and documents D_q.

Query-Title Interaction Graph Construction. Given a set of queries Q_q and D_q, the next step is to extract a representative phrase that captures the underlying user attentions or interests. Figure 7.3 shows a query-doc cluster and the concept phrase extracted from it. We can extract 'Hayao Miyazaki animated film (宫崎骏|动画|电影)" from the input query-title cluster. An attention phrase features multiple characteristics. First, the tokens in it may show up multiple times in the queries and

document titles. Second, the phrase tokens are not necessarily consecutively or fully contained by a single query or title. For example, in Figure 7.3, additional tokens such as "famous (著名的)" will be inserted into the phrase tokens, making them not a consecutive span. Third, the phrase tokens are syntactically dependent even if they are not consecutively adjacent in a query or title. For example, "Hayao Miyazaki (宫崎骏)" and "film (电影)" forms a compound noun name. Finally, the order of phrase tokens may change in different text. In Figure 7.3, the tokens in the concept phrase "Hayao Miyazaki animated film (宫崎骏|动画|电影)" are fully contained by the query and two titles, while the order of the tokens varies in different queries or titles. Other types of attention phrases such as events and topics will feature similar characteristics.

To fully exploit the characteristics of attention phrases, we propose Query-Title Interaction Graph (QTIG), a novel graph representation of queries and titles to reveal the correlations between their tokens. Based on it, we further propose GCTSP-Net, a model that takes a query-title interaction graph as input, performs node classification with graph convolution, and finally generates a phrase by Asymmetric Traveling Salesman Decoding (ATSP-decoding).

Figure 7.3 shows an example of query-title interaction graph constructed from a set of queries and titles. Denote a QTIG constructed from queries Q_q and the titles T_q of documents D_q as $G_{qt}(Q_q, T_q)$. The queries and documents are sorted by the weights calculated during the random walk. In $G_{qt}(Q_q, T_q)$, each node is a unique token belonging to Q_q or T_q. The same token present in different input text will be merged into one node. For each pair of nodes, if they are adjacent tokens in any query or title, they will be connected by a bi-directional "`seq`" edge, indicating their order in the input. In Figure 7.3, the inverse direction of a "seq" edge points to the preceding words, which is indicated by a hollow triangle pointer. If the pair of nodes are not adjacent to each other, but there exists syntactical dependency between them, they will be connected by a bi-directional dashed edge which indicates the type of syntactical dependency relationship and the direction of it, while the inverse direction is also indicated by a hollow triangle pointer.

Algorithm 7 shows the process of constructing a query-title interaction graph. We construct the nodes and edges by reading the inputs in Q_q and T_q in order. When we construct the edges between two nodes, as two nodes may have multiple adjacent relationships or syntactical relationships in different inputs, we only keep the first edge constructed. In this way, each pair of related nodes will only be connected by a bi-directional sequential edge or a syntactical edge. The idea is that we prefer the "`seq`" relationship as it shows a stronger connection than any syntactical dependency,

ALGORITHM 7: Constructing the Query-Title Interaction Graph

Input: a sequence of queries $Q = \{q_1, q_2, \cdots, q_{|Q|}\}$, document titles $T = \{t_1, t_2, \cdots, t_{|T|}\}$.
Output: a Query-Title Interaction Graph $G_{qt}(Q, T)$.

Create node set $V = \{sos, eos\}$, edge set $E = \{\}$;
for each input text passage $x \in Q$ or $x \in T$ **do**
    append "`sos`" and "`eos`" as the first and the last token of x;
    construct a new node for each token in x;
    construct a bi-directional "`seq`" edge for each pair of adjacent tokens;
    append each constructed node and edge into V or E only if the node is not contained by V, or no edge with the same source and target tokens exists in E;
end for
for each input text passage $x \in Q$ or $x \in T$ **do**
    perform syntactic parsing over x;
    construct a bi-directional edge for each dependency relationship;
    append each constructed edge into E if no edge with the same source and target tokens exists in E;
end for
construct graph $G_{qt}(Q, T)$ from node set V and edge set E.

and we prefer the syntactical relationships contained in the higher-weighted input text instead of the relationships in lower-weighted inputs. Compared with including all possible edges in a query-title interaction graph, empirical evidence suggests that our graph construction approach gives better performance for phrase mining.

After constructing a graph $G_{qt}(Q_q, T_q)$ from a query-title cluster, we extract a phrase p by our GCTSP-Net, which contains a classifier to predict whether a node belong to p, and an asymmetric traveling salesman decoder to order the predicted positive nodes.

Node Classification with R-GCN. In the GCTSP-Net, we apply Relational Graph Convolutional Networks (R-GCN) [Kipf and Welling, 2016; Gilmer *et al.*, 2017; Schlichtkrull *et al.*, 2017] to our constructed QTIG to perform node classification.

Denote a directed and labeled multi-graph as $G = (V, E, R)$ with nodes $v_i, w_i \in V$, labeled edges $e_{vw} = (v, r, w) \in E$, where $r \in R$ is a relation type. A class of graph convolutional networks can be understood as a message-passing framework [Gilmer *et al.*, 2017], where the hidden states h_v^l of each node $v \in G$ at layer l are updated based on messages m_v^{l+1} according to:

$$m_v^{l+1} = \sum_{w \in N(v)} M_l(h_v^l, h_w^l, e_{vw}), \tag{7.3}$$

$$h_v^{l+1} = U_l(h_v^l, m_v^{l+1}), \tag{7.4}$$

where $N(v)$ denotes the neighbors of v in graph G, M_l is the message function at layer l, and U_l is the vertex update function at layer l.

Specifically, the message passing function of Relational Graph Convolutional Networks is defined as:

$$h_v^{l+1} = \sigma\left(\sum_{r\in R}\sum_{w\in N^r(v)}\frac{1}{c_{vw}}W_r^l h_w^l + W_0^l h_v^l\right), \tag{7.5}$$

where $\sigma(\cdot)$ is an element-wise activation function such as $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$. $N^r(v)$ is the set of neighbors under relation $r \in R$. c_{vw} is a problem-specific normalization constant that can be learned or pre-defined (e.g., $c_{vw} = |N^r(v)|$). W_r^l and W_0^l are learned weight matrices.

We can see that an R-GCN accumulates transformed feature vectors of neighboring nodes through a normalized sum. Besides, it learns relation-specific transformation matrices to take the type and direction of each edge into account. In addition, it adds a single self-connection of a special relation type to each node to ensure that the representation of a node at layer $l+1$ can also be informed by its representation at layer l.

To avoid the rapid growth of the number of parameters when increasing the number of relations $|R|$, R-GCN exploits basis decomposition and block-diagonal decomposition to regularize the weights of each layer. For basis decomposition, each weight matrix $W_r^l \in \mathbb{R}^{d^{l+1}\times d^l}$ is decomposed as:

$$W_r^l = \sum_{b=1}^{B} a_{rb}^l V_b^l, \tag{7.6}$$

where $V_b^l \in \mathbb{R}^{d^{l+1}\times d^l}$ are base weight matrices. In this way, only the coefficients a_{rb}^l depend on r. For block-diagonal decomposition, W_r^l is defined through the direct sum over a set of low-dimensional matrices:

$$W_r^l = \bigoplus_{b=1}^{B} Q_{br}^l, \tag{7.7}$$

where $W_r^l = \mathrm{diag}(Q_{1r}^l, Q_{2r}^l, \cdots, Q_{br}^l)$ is a block-diagonal matrix with $Q_{br}^l \in \mathbb{R}^{(d^{l+1}/B)\times(d^l/B)}$. The basis function decomposition introduces weight sharing between different relation types, while the block decomposition applies sparsity constraint on the weight matrices.

In our model, we apply R-GCN with basis decomposition to query-title interaction graphs to perform node classification. For each node in the graph, we represent it by a feature vector consisting of the embeddings of the token's named entity recognition (NER) tag, part-of-speech (POS) tag, whether it is a stop word, number of characters in the token, as well as the sequential id that indicates the order each node be added

to the graph. Using the concatenation of these embeddings as the initial node vectors, we pass the graph to a multi-layer R-GCN with a softmax$(\cdot)$ activation (per node) on the output of the last layer. We label the nodes belonging to the target phrase p as 1 and other nodes as 0, and train the model by minimizing the binary cross-entropy loss on all nodes.

Node Ordering with ATSP Decoding. After classified a set of tokens V_p as target phrase nodes, the next step is to sort the nodes to get the final output. In our GCTSP-Net, we propose to model the problem as an asymmetric traveling salesman problem (ATSP), where the objective is to find the shortest route that starts from the "`sos`" node, visits each predicted positive nodes, and returns to the "`eos`" node. This approach is named as ATSP-decoding in our work.

We perform ATSP-decoding with a variant of the constructed query-title interaction graph. First, we remove all the syntactical dependency edges. Second, instead of connecting adjacent tokens by a bi-directional "`seq`" edge, we change it into unidirectional to indicate the order in input sequences. Third, we connect "`sos`" with the first predicted positive token in each input text, as well as connect the last predicted positive token in each input with the "`eos`" node. In this way, we remove the influence of prefixing and suffixing tokens in the inputs when finding the shortest path. Finally, the distance between a pair of predicted nodes is defined as the length of the shortest path in the modified query-title interaction graph. In this way, we can solve the problem with Lin-Kernighan Traveling Salesman Heuristic [Helsgaun, 2000] to get a route of the predicted nodes and output p.

We shall note that ATSP-decoding will produce a phrase that contains only unique tokens. In our work, we observe that only less than 1% attention phrases contain duplicated tokens, while most of the duplicated tokens are punctuations. Even if we need to produce duplicate tokens when applying our model to other tasks, we just need to design task-specific heuristics to recognize the potential tokens (such as count their frequency in each query and title), and construct multiple nodes for it in the query-title interaction graph.

Attention Phrase Normalization. The same user attention may be expressed by slightly different phrases. After extracting a phrase using GCTSP-Net, we merge highly similar phrases into one node in out Attention Ontology. Specifically, we examine whether a new phrase p_n is similar to an existing phrase p_e by two criteria: i) the non-stop words in p_n shall be similar (same or synonyms) with that in p_e, and ii) the TF-IDF similarity between their context-enriched representations shall be above a threshold δ_m. The context-enriched representation of a phrase is obtained by using itself as a query and concatenating the top 5 clicked titles.

Training Dataset Construction. To reduce human effort and accelerate the labeling process of training dataset creation, we design effective unsupervised strategies to extract candidate phrases from queries and titles, and provide the extracted candidates together with query-title clusters to workers as assistance. For concepts, we combine bootstrapping with query-title alignment [Liu *et al.*, 2019*c*]. The bootstrapping strategy exploits pattern-concept duality: we can extract a set of concepts from queries following a set of patterns, and we can learn a set of new patterns from a set of queries with extracted concepts. Thus, we can start from a set of seed patterns, and iteratively accumulate more and more patterns and concepts. The query-title alignment strategy is inspired by the observation that a concept in a query is usually mentioned in the clicked titles associated with the query, yet possibly in a more detailed manner. Based on this observation, we align a query with its top clicked titles to find a title chunk which fully contains the query tokens in the same order and potentially contains extra tokens within its span. Such a title chunk is selected as a candidate concept.

For events, we split the original unsegmented document titles into subtitles by punctuations and spaces. After that, we only keep the set of subtitles with lengths between L_l (we use 6) and L_h (we use 20). For each remaining subtitle, we score it by counting how many unique non-stop query tokens within it. The subtitles with the same score will be sorted by its click-through rate. Finally, we select the top ranked subtitle as a candidate event phrase.

Attention Derivation. After extracting a collection of attention nodes (or phrases), we can further derive higher level concepts or topics from them, which automatically become their parent nodes in our Attention Ontology.

On one hand, we derive higher-level concepts by applying common suffix discovery (CSD) to extracted concepts. We perform word segmentation over all concept phrases, and find out the high-frequency suffix words or phrases. If the suffixes forms a noun phrase, we add it as a new concept node. For example, the concept "animated film (动画|电影)" can be derived from "famous animated film (著名的|动画|电影)", "award-winning animated film (获奖的|动画|电影)" and "Hayao Miyazaki animated film (宫崎骏|动画|电影)", as they share the common suffix "animated film (动画|电影)"

On the other hand, we drive high-level topics by applying common pattern discovery (CPD) to extracted events. We perform word segmentation, named entity recognition and part-of-speech tagging over the event phrases. After that, we find out high-frequency event patterns and recognize the different elements in the events. If the elements (such as entities or locations of events) have *isA* relationship with one or multiple common concepts, we replace the different elements by the most fine-

grained common concept ancestor in the ontology. For example, we can derive a topic "Singer will have a concert (歌手|开|演唱会)" from "Jay Chou will have a concert (周杰伦|开|演唱会)" and "Taylor Swift will have a concert (泰勒斯威夫特|开|演唱会)", as the two phrases sharing the same pattern "XXX will have a concert (XXX|开|演唱会)", and both "Jay Chou (周杰伦)" and "Taylor Swift (泰勒斯威夫特)" belong to the concept "Singer (歌手)".

7.3.2 Linking User Attentions

The previous step produces a large set of nodes representing user attentions (or interests) in different granularities. Our goal is to construct a taxonomy based on these individual nodes to show their correlations. With edges between different user attentions, we can reason over it to infer a user's real interest.

In this section, we describe our methods to link attention nodes and construct the complete ontology. We will construct the *isA* relationships, *involve* relationships, and *correlate* relationships between categories, extracted attention nodes and entities to construct a ontology. We exploit the following action-driven strategies to link different types of nodes.

Edges between Attentions and Categories. To identify the *isA* relationship between *attention-category* pairs, we utilize the co-occurrence of them shown in user search click logs. Given an attention phrase p as the search query, suppose there are n^p clicked documents of search query p in the search click logs, and among them there are n_g^p documents belong to category $\mathbf{g}$. We then estimate $\mathbb{P}(g|p)$ by $\mathbb{P}(g|p) = n_g^p/n^p$. We identify that there is an *isA* relationship between p and g if $\mathbb{P}(g|p) > \delta_g$ (we set $\delta_g = 0.3$).

Edges between Attentions. To discover *isA* relationships, we utilize the same criteria when we perform attention derivation: we link two concepts by the *isA* relationship if one concept is the suffix of another concept, and we link two topic/event attentions by the *isA* relationship if they share the same pattern and there exists *isA* relationships between their non-overlapping tokens. Note that if a topic/event doesn't contain an element of another topic/event phrase, it also indicates that they have *isA* relationship. For example, "Jay Chou will have a concert" has *isA* relationship with both "Singer will have a concert" and "have a concert". For the *involve* relationship, we connect a concept to a topic if the concept is contained in the topic phrase.

Edges between Attentions and Entities. We extract: i) *isA* relationships between concepts and entities; ii) *involve* relationships between topics/events and entities, locations or triggers.

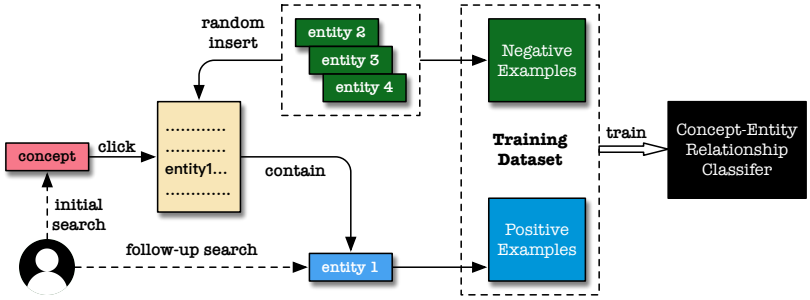


Figure 7.4: Automatic construction of the training datasets for classifying the *isA* relationship between concepts and entities.

For concepts and entities, using strategies such as co-occurrence will introduce a lot of noises, as co-occurrence doesn't always indicate an entity belongs to a concept. To solve this issue, we propose to train a concept-entity relationship classifier based on the concept and the entity's context information in the clicked documents. Labeling a training dataset for this task requires a lot of human efforts. Instead of manual labeling, we propose a method to automatically construct a training dataset from user search click graphs. Figure 7.4 shows how we construct such a training dataset. We select the concept-entity pairs from search logs as positive examples if: i) the concept and the entity are two consecutive queries from one user, and ii) the entity is mentioned by a document which a user clicked after issuing the concept as a search query. Besides, we select entities belonging to the same higher-level concept or category, and insert them into random positions of the document to create negative examples of the dataset. For the classifier, we can train a classifier such as GBDT based on manual features, or fine-tune a pre-trained language model to incorporate semantic features and infer whether the context indicates a *isA* relationship between the concept-entity pair.

For events/topics and entities, we only recognize the important entities, triggers and locations in the event/topic, and connect them by an *involve* relationship edge. We first create an initial dataset by extracting all the entities, locations, and trigger words in the events/topic based on a set of predefined trigger words and entities. Then the dataset is manually revised by workers to remove the unimportant elements. Based on this dataset, we reuse our GCTSP-Net and train it without ATSP-decoding to perform 4-class (entity, location, trigger, other) node classification over the query-title interaction graphs of the events/topics. In this way, we can recognize the different elements of an event or topic, and construct *involve* edges between them.

Edges between Entities. Finally, we construct the *correlate* relationship between entity pairs by the co-occurrence information in user queries and documents.

We utilize high frequency co-occurring entity pairs in queries and documents as positive entity pairs, and perform negative sampling to create negative entity pairs. After automatically created a dataset from search click logs and web documents, we learn the embedding vectors of entities with Hinge loss, so that the Euclidean distance between two correlated entities will be small. After learned the embedding vectors of different entities, we classify a pair of entities as correlated if their Euclidean distance is smaller than a threshold.

Note that the same approach for *correlate* relationship discovery can be applied to other type of nodes such as concepts. Currently, we only constructed such relationships between entities.

7.4 Applications

In this section, we show how our attention ontology can be applied to a series of NLP tasks to achieve user-centered text understanding.

Story Tree Formation. The relationships between events and the involved entities, triggers and locations can be utilized to cluster correlated events and form a story tree [Liu *et al.*, 2017]. A story tree organizes a collection of related events with a tree structure to highlight the evolving nature of a real-world story. Given an event p^e, we retrieve a collection of related events P^e and use them to form a story tree, which allows us to better serve the users by recommending follow-up events from P^e when they have read news articles about p^e.

Constructing a story tree from an attention ontology involves four steps: retrieving correlated events, calculating similarity matrix, hierarchical clustering, and tree formation. First, with the help of the attention ontology, we can retrieve related events set P^e give an event p^e. Specifically, the criteria to retrieve "correlated" events can be flexible. For example, we can set a requirement that each event $p_i \in P^e$ shares at least one common child entity with p^e, or we can force the triggers of them to be the same. Second, we can estimate the similarities between each pair of events based on the text similarity of event phrases and the common entities, triggers or locations shared by them. Specifically, we calculate the similarity between two events by:

$$s(p_1^e,\ p_2^e) = f_m(p_1^e,\ p_2^e) + f_g(p_1^e,\ p_2^e) + f_e(p_1^e,\ p_2^e), \tag{7.8}$$

$$f_m(p_1^e,\ p_2^e) = \text{CosineSimilarity}(\mathbf{v}^{p_1^e},\ \mathbf{v}^{p_2^e}), \tag{7.9}$$

$$f_g(p_1^e,\ p_2^e) = \text{CosineSimilarity}(\mathbf{v}^{g_1^e},\ \mathbf{v}^{g_2^e}), \tag{7.10}$$

$$f_e(p_1^e,\ p_2^e) = \text{TF-IDFSimilarity}(E^{p_1^e},\ E^{p_2^e}), \tag{7.11}$$

where $s(p_1^e,\ p_2^e)$ is the measured similarity between the two events, It is given by the sum of three scores: i) $f_m(p_1^e,\ p_2^e)$, which represents the semantic distance between two event phrases. We use the cosine similarity of BERT-based phrase encoding vectors $\mathbf{v}^{p_1^e}$ and $\mathbf{v}^{p_2^e}$ for the two events [Devlin *et al.*, 2018] ; ii) $f_g(p_1^e,\ p_2^e)$, which represents the similarity of the triggers in two events. We calculate the similarity between trigger g_1^e in event p_1^e and g_2^e in p_2^e by the cosine similarity of the word vectors $\mathbf{v}^{g_1^e}$ and $\mathbf{v}^{g_2^e}$ from [Song *et al.*, 2018*c*]; iii) $f_e(p_1^e,\ p_2^e)$, the TF-IDF similarity between the set of entities $E^{p_1^e}$ of event p_1^e and $E^{p_2^e}$ of p_2^e. After the measurement of the similarities between events, we perform hierarchical clustering to group them into hierarchical clusters. Finally, we order the events by time, and put the events in the same cluster into the same branch. In this way, the cluster hierarchy is transformed into the branches of a tree.

Document Tagging We can also utilize the attention phrases to describe the main topics of a document by tagging the correlated attentions to the document, even if the phrase is not explicitly mentioned in the document. For example, a document talking about films "Captain America: The First Avenger", "Avengers: Endgame" and "Iron Man" can be tagged with the concept "Marvel Super Hero Movies" even though the concept may not be contained by it. Similarly, a document talking about "Theresa May's Resignation Speech" can be tagged by topics "Brexit Negotiation", while traditional keyword-based methods are not able to reveal such correlations.

To tag concepts to documents, we combine a matching-based approach and a probabilistic inference-based approach based on the key entities in a document. Suppose d contains a set of key entities E^d. For each entity $e^d \in E^d$, we obtain its parent concepts P^c in the attention ontology as candidate tags. For each candidate concept $p^c \in P^c$, we score the coherence between d and p^c by calculating the TF-IDF similarity between the title of d and the context-enriched representation of p^c (i.e., the topic clicked titles of p^c).

When no parent concept can be found by the attention ontology, we identify relevant concepts by utilizing the context information of the entities in d. Denote the

probability that concept p^c is related to document d as $\mathbb{P}(p^c|d)$. We estimate it by:

$$\mathbb{P}(p^c|d) = \sum_{i=1}^{|E^d|} \mathbb{P}(p^c|e_i^d)\mathbb{P}(e_i^d|d), \tag{7.12}$$

$$\mathbb{P}(p^c|e_i^d) = \sum_{j=1}^{|X_{e_i^d}|} \mathbb{P}(p^c|x_j)\mathbb{P}(x_j|e_i^d), \tag{7.13}$$

$$\mathbb{P}(p^c|x_j) = \begin{cases} \frac{1}{|P_{x_j}^c|} & \text{if } x_j \text{ is a substring of } p^c, \\ 0 & \text{otherwise.} \end{cases} \tag{7.14}$$

where E^d is the key entities of d, $\mathbb{P}(e_i^d|d)$ is the document frequency of entity $e_i^d \in E^d$. $\mathbb{P}(p^c|e_i^d)$ estimates the probability of concept p^c given e_i^d, which is inferred from the context words of e_i^d. $\mathbb{P}(x_j|e_i^d)$ is the co-occurrence probability of context word x_j with e_i^d. We consider two words as co-occurred if they are contained in the same sentence. $X_{e_i^d}$ are the set of contextual words of e_i^d in d. $P_{x_j}^c$ is the set of concepts containing x_j as a substring.

To tag events or topics to a document, we combine longest common subsequence based (LCS-based) textural matching with Duet-based semantic matching [Mitra *et al.*, 2017]. For LCS-based matching, we concatenate a document title with the first sentence in content, and calculate the length of longest common subsequence between a topic/event phrase and the concatenated string. For Duet-based matching, we utilize the Duet neural network [Mitra *et al.*, 2017] to classify whether the phrase matches with the concatenated string. If the length of the longest common subsequence is above a threshold and the classification result is positive, we tag the phrase to the document.

Query Understanding. A user used to search about an entity may be interested in a broader class of similar entities. However, the user may not be even aware of the entities similar to the query. With the help of our ontology, we can better understand users' implicit intention and perform query conceptualization or recommendation to improve the user experience in search engines. Specifically, we analyze whether a query q contains a concept p^c or an entity e. If a query conveys a concept p^c, we can rewrite it by concatenating q with each of the entities e_i that have *isA* relationship with p^c. In this way, we rewrite the query to the format of "q e_i". If a query conveys an entity e, we can perform query recommendation by recommend users the entities e_i that have *correlate* relationship with e in the ontology.

	Category	Concept	Topic	Event	Entity
Quantity	1,206	460,652	12,679	86,253	1,980,841
Grow / day	-	11,000	-	120	-

Table 7.1: Nodes in the attention ontology.

	isA	*correlate*	*involve*
Quantity	490,741	1,080,344	160,485
Accuracy	95%+	95%+	99%+

Table 7.2: Edges in the attention ontology.

7.5 Evaluation

Our proposed GIANT system is the core ontology system in multiple applications in anonymous company and is serving more than a billion daily active users all around the world. It is implemented by Python 2.7 and C++. Each component of our system works as a service and is deployed with Tars[1], a high-performance remote procedure call (RPC) framework based on name service and Tars protocol. The attention mining and linking services are deployed on 10 dockers, with each configured with four processor Intel(R) Xeon(R) Gold 6133 CPU @ 2.50GHz and 6GB memory. Applications such as document tagging are running on 48 dockers with the same configuration. MySQL database is used for data storage. We construct the attention ontology from large-scale real-world daily user search click logs. While our system is deployed on Chinese datasets, the techniques proposed in our work can be easily adapted to other languages.

7.5.1 Evaluation of the Attention Ontology

Table 7.1 shows the statistics of different nodes in the attention ontology. We extract attention phrases from daily user search click logs. Therefore, the scale of our ontology keeps growing. Currently, our ontology contains 1,206 predefined categories, 460,652 concepts, 12,679 topics, 86,253 events and 1,980,841 entities. We are able to extract around 27,000 concepts and 400 events every day, and around 11,000 concepts and 120 events are new. For online concept and event tagging, our system processes 350 documents per second.

Table 7.2 shows the statistics and accuracies of different types of edges (relationships) in the attention ontology. Currently, we have constructed more than 490K *isA*

[1]https://github.com/TarsCloud/Tars

Categories	Concepts	Instances
Sports	Famous long-distance runner	Dennis Kipruto Kimetto, Kenenisa Bekele
Stars	Actors who committed suicide	Robin Williams, Zhang Guorong, David Strickland
Drama series	American crime drama series	American Crime Story, Breaking Bad, Criminal Minds
Fiction	Detective fiction	Adventure of Sherlock Holmes, The Maltese Falcon

Table 7.3: Showcases of concepts and the related categories and entities.

relationships, $1,080K$ *correlate* relationships, and $160K$ *involve* relationships between different nodes. The human evaluation performed by three managers in anonymous company shows the accuracies of the three relationship types are above 95%, 95% and 99%, respectively.

Categories	Topics	Events	Entities
Music	Taylor Swift and Katy Perry	Taylor Swift and Katy Perry's conciliation	Katy Perry, Taylor Swift
Cellphone	cellphone launch events	Apple news conferences 2018 mid-season, Samsung Galaxy S9 officially released	Apple, iPhone, Samsung, Galaxy S9
Esports	League of Legends season 8	LOL S8 finals announcement, IG wins the S8 final, IG's reward for winning the S8 final revealed	League of Legends, IG team, finals

Table 7.4: Showcases of events and the related categories, topics and involved entities.

To give intuition into what kind of attention phrases can be derived from search click graphs, Table 7.3 and Table 7.4 show a few typical examples of concepts and events (translated from Chinese), and some topics, categories, and entities that share *isA* relationship with them. By fully exploiting the information of user actions contained in search click graphs, we transform user actions to user attentions, and extract concepts such as "Actors who committed suicide (自杀的演员)". Besides, we can also extract events or higher level topics of users' interests, such as "Taylor Swift and

Katy Perry" if a user often inputs related queries. Based on the connections between entities, concepts, events and topics, we can also infer what a user really cares and improve the performance of recommendation.

7.5.2 Evaluation of the GCTSP-Net

We evaluate our GCTSP-Net on multiple tasks by comparing it to a variety of baseline approaches.

Datasets. To the best of our knowledge, there is no publicly available dataset suitable for the task of heterogeneous phrase mining from user queries and search click graphs. To construct the user attention ontology, we create two large-scale datasets for concept mining and event mining using the approaches described in Sec. 7.3: the Concept Mining Dataset (CMD) and the Event Mining Dataset (EMD). These two datasets contain $10,000$ examples and $10,668$ examples respectively. Each example is composed of a set of correlated queries and top clicked document titles from real-world query logs, together with a manually labeled gold phrase (concept or event). For the event mining dataset, it further contains triggers, key entities and locations of each event. We use the earliest article publication time as the time of each event example. The datasets are labeled by 3 professional product managers in anonymous company and 3 graduate students. For each dataset, we utilize 80% as training set, 10% as development set, and 10% as test set. The datasets will be published for research purposes [2].

Methodology and Models for Comparison. We compare our GCTSP-Net with the following baseline methods on the concept mining task:

- **TextRank**. A classical graph-based keyword extraction model [Mihalcea and Tarau, 2004].[3]
- **AutoPhrase**. A state-of-the-art phrase mining algorithm that extracts quality phrases based on POS-guided segmentation and knowledge base [Shang *et al.*, 2018].[4]
- **Match**. Extract concepts from queries and titles by matching using patterns from bootstrapping [Liu *et al.*, 2019*c*].
- **Align**. Extract concepts by the query-title alignment strategy described in Sec. 7.3.1.

[2]https://github.com/anonymous/GIANT
[3]https://github.com/letiantian/TextRank4ZH
[4]https://github.com/shangjingbo1226/AutoPhrase

- **MatchAlign**. Extract concepts by both pattern matching and query-title alignment strategy.
- **LSTM-CRF-Q**. Apply LSTM-CRF [Huang *et al.*, 2015] to input query.
- **LSTM-CRF-T**. Apply LSTM-CRF [Huang *et al.*, 2015] to titles.

For the TextRank and AutoPhrase algorithm, we extract the top 5 keywords or phrases from queries and titles, and concatenate them in the same order with the query/title to get the extracted phrase. For MatchAlign, we select the most frequent result if multiple phrases are extracted. For LSTM-CRF-Q/LSTM-CRF-T, it consists of a 200-dimensional word embedding layer initialized with the word vectors proposed in [Song *et al.*, 2018*c*], a BiLSTM layer with hidden size 25 for each direction, and a Conditional Random Field (CRF) layer which predicts whether each word belongs to the output phrase by Beginning-Inside–Outside (BIO) tags.

For event mining task, we compare with TextRank and LSTM-CRF. In addition, we also compare with:

- **TextSummary [Mihalcea and Tarau, 2004]**. An encoder-decoder model with attention mechanism for text summarization.[5]
- **CoverRank**. Rank queries and subtitles by counting the covered nonstop query words, as described in 7.3.1.

For TextRank, we use the top 2 queries and top 2 selected subtitles given by CoverRank, and perform re-ranking. For TextSummary, we use the 200-dimensional word embeddings in [Song *et al.*, 2018*c*], two-layer BiLSTM (256 hidden size for each direction) as encoder, and one layer LSTM with 512 hidden size and attention mechanism as decoder (beam size for decoding is 10). We feed the concatenation of queries and titles into TextSummary to generate the output. For LSTM-CRF, the LSTM layer in it is configured similarly to the encoder of TextSummary. We feed each title individually into it to get different outputs, filter the outputs by length (number of characters between 6 and 20), and finally select the phrase which belongs to the top clicked title.

For the task of event key elements recognition (key entities, trigger, location), it is a 4-class classification task over each word. We compare our model with LSTM and LSTM-CRF. The difference between LSTM and LSTM-CRF is that LSTM replaces the CRF layer in LSTM-CRF with a softmax layer.

For each baseline, we individually tune the hyper-parameters in order to achieve its best performance. As to our approach (GCTSP-Net), we stack 5-layer R-GCN

[5]https://github.com/dongjun-Lee/text-summarization-tensorflow

Method	EM	F1	COV
TextRank	0.1941	0.7356	1
AutoPhrase	0.0725	0.4839	0.9353
Match	0.1494	0.3054	0.3639
Align	0.7016	0.8895	0.9611
MatchAlign	0.6462	0.8814	0.9700
Q-LSTM-CRF	0.7171	0.8828	0.9731
T-LSTM-CRF	0.3106	0.6333	0.9062
GCTSP-Net	**0.783**	**0.9576**	**1**

Table 7.5: Compare concept mining approaches.

Method	EM	F1	COV
TextRank	0.3968	0.8102	1
CoverRank	0.4663	0.8169	1
TextSummary	0.0047	0.1064	1
LSTM-CRF	0.4597	0.8469	1
GCTSP-Net	**0.5164**	**0.8562**	0.9972

Table 7.6: Compare event mining approaches.

with hidden size 32 and number of bases $B = 5$ in basis decomposition for graph encoding and node classification. We will open-source our code together with the datasets for research purposes.

Metrics. We use Exact Match (EM), F1 and coverage rate (COV) to evaluate the performance of event and concept mining tasks. The exact match score is 1 if the prediction is exactly the same as ground-truth or 0 otherwise. F1 measures the portion of overlap tokens between the predicted phrase and the ground-truth phrase [Rajpurkar *et al.*, 2016]. The coverage rate measures the percentage of non-empty predictions of each approach. For the event key elements recognition task, we evaluate by the F1-macro, F1-micro, and F1-weighted metrics.

Evaluation results and analysis. Table 7.5, Table 7.6, and Table 7.7 compare our model with different baselines on the CMD and EMD datasets for concept mining,

Method	F1-macro	F1-micro	F1-weighted
LSTM	0.2108	0.5532	0.6563
LSTM-CRF	0.2610	0.6468	0.7238
GCTSP-Net	**0.6291**	**0.9438**	**0.9331**

Table 7.7: Compare event key elements recognition approaches.

event mining, and event key elements recognition. We can see that our unified model for different tasks significantly outperforms all baseline approaches. The outstanding performance can be attribute to: first, our query-title interaction graph efficiently encodes the information of word adjacency, word features, dependencies, query-title overlapping and so on in a structured manner, which are critical for both attention phrase mining tasks and event key elements recognition tasks. Second, the multi-layer R-GCN encoder in our model can learn from both the node features and the multi-resolution structural patterns from query-title interaction graph. Therefore, combining query-title interaction graph with R-GCN encoder, we can achieve great performance in different node classification tasks. Furthermore, the unsupervised ATSP-decoding sorts the extracted tokens to form an ordered phrase efficiently. In contrast, heuristic-based approaches and LSTM-based approaches are not robust to the noises in dataset, and cannot capture the structural information in queries and titles. In addition, existing keyphrase extraction methods such as TextRank [Mihalcea and Tarau, 2004] and AutoPhrase [Shang *et al.*, 2018] are better suited for extracting key words or phrases from long documents, lacking the ability to give satisfactory performance in our tasks.

7.5.3 Applications: Document Tagging and Story Tree Formation

Document Tagging. For document tagging, our system currently processes around $1,525,682$ documents per day, where about 35% of them can be tagged with at least one concept, and 4% can be tagged with an event. We perform human evaluation by sampling 500 documents for each major category ("game", "technology", "car", and "entertainment"). The result shows that the precision of concept tagging for documents is 88% for "game" category, 91% for "technology", 90% for "car", and 87% for "entertainment". The overall precision for documents of all categories is 88%. As to event tagging on documents, the overall precision is 96%.

Story Tree Formation. We apply the story tree formation algorithm described in Sec. 7.4 to real-world events to test its performance. Figure 7.5 gives an example to illustrate what we can obtain through our approach. In the example, each node is an event, together with the documents that can be tagged by this event. We can see that our method can successfully cluster events related to "China-US Trade", ordering the events by the published time of the articles, and show the evolving structure of coherent events. For example, the branch consists of events $3 \sim 6$ are mainly about "Sino-US trade war is emerging", the branch $8 \sim 10$ are resolving around

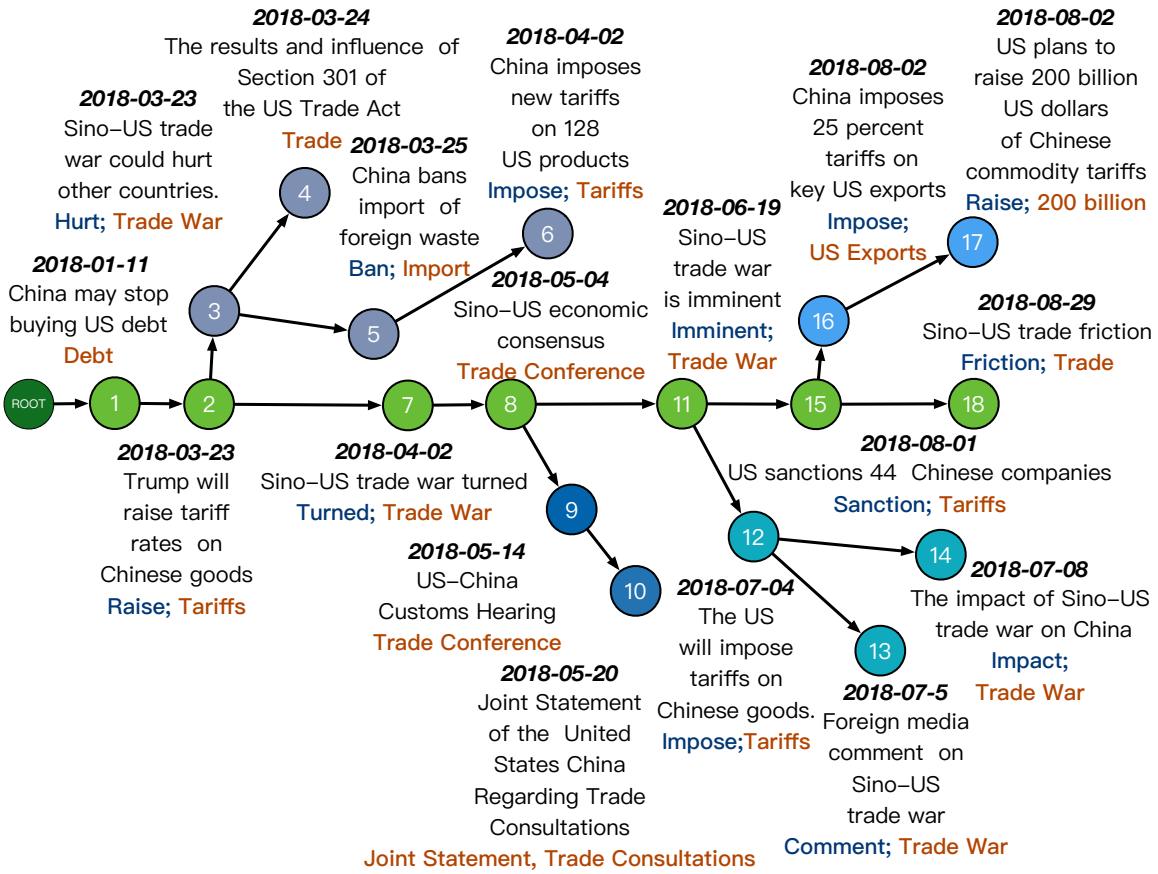


Figure 7.5: An example to show the constructed story tree given by our approach.

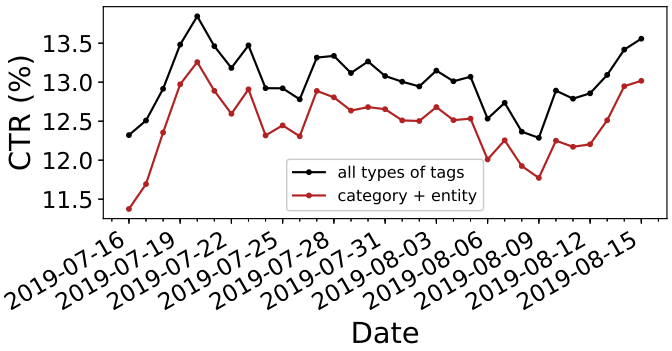


Figure 7.6: The click-through rates with/without extracted tags.

"US agrees limited trade deal with China", and 11 $\sim$ 14 are about "Impact of Sino-US trade war felt in both countries". By clustering and organizing events and related documents in such a tree structure, we can track the development of different stories (clusters of coherent events), reduce information redundancy, and improve document recommendation by recommending users the follow-up events they are interested in.

7.5.4 Online Recommendation Performance

We evaluate the effect of our attention ontology on recommendations by analyzing its performance in the news feeds stream of an anonymous application which has more than 110 million daily active users. In the application, both users and articles are tagged with categories, topics, concepts, events or entities from the attention ontology, as shown in Figure 7.2. The application recommends news articles to users based on a variety of strategies, such as content-based recommendation, collaborative filtering and so on. For the content-based recommendation, it matches users with articles through the common tags they share.

We analyze the Click-Through Rate (CTR) given by different tags from July 16, 2019 to August 15, 2019 to evaluate their effects. Click-through rate is the ratio of the number of users who click on a recommended link to the number of total users who received the recommendation. Figure 7.6 compares the CTR when we recommend documents to users with different strategies. Traditional recommender systems only utilize the category tags and entity tags. We can see that including more types of attention tags (topics, events, or concepts) in recommendation can constantly help improving the CTR on different days: the average CTR improved from 12.47% to 13.02%. The reason is that the extracted concepts, events or topics can depict user interests with suitable granularity. They are quite helpful for solving the inaccurate recommendation and monotonous recommendation problem.

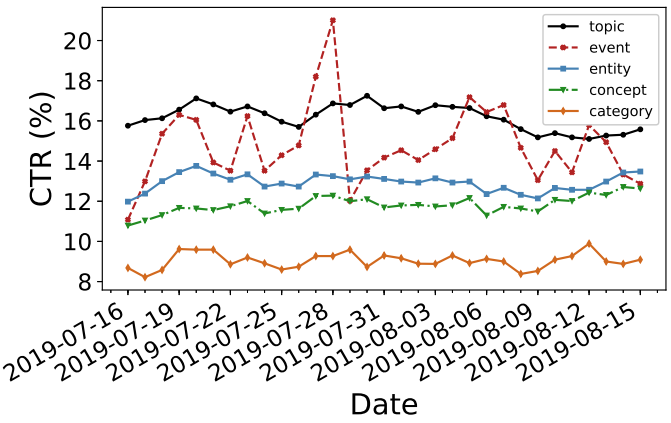


Figure 7.7: The click-through rates of different tags.

We further analyze the effects of each type of tags in recommendation. Figure 7.7 shows the CTR of the recommendations given by different types of tags. The average CTR for topic, event, entity, concept and category are 16.18%, 14.78%, 12.93%, 11.82%, and 9.04%, respectively. The CTR of both topic tags and event tags are much higher than that of category and entities, which shows the effectiveness of our constructed ontology. For events, the events happening on each days dramatically different with each other, and they are not always attractive to users. Therefore, the CTR of event tags is less stable than the topic tags. For concept tags, they are generalization of fine-grained entities and has *isA* relationship with them. As there may have noises when we perform user interest inference using the relationships between entities and concepts, the CTR of concepts are slightly lower than entities. However, compared to entities, our experience show that concepts can significantly increase the diversity of recommendation and are helpful in solving the problem of monotonous recommendation.

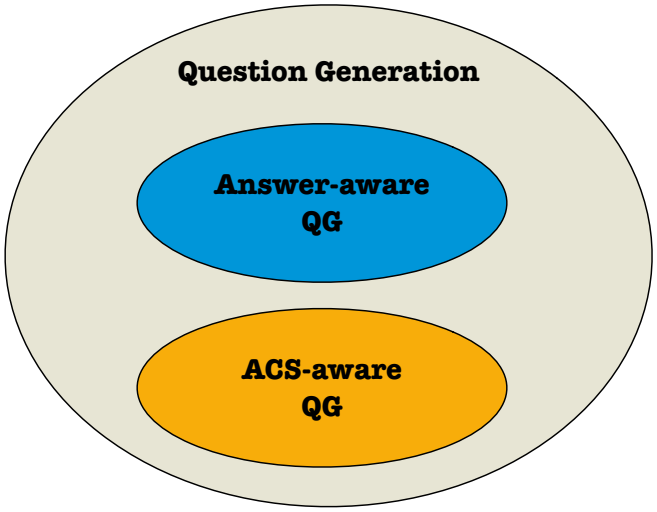
7.6 Conclusion

In this chapter, we describe our design and implementation of GIANT, a system that proposed to construct a web-scale user attention ontology from large amount of query logs and search click graphs for various applications. It consists of around two millions of heterogeneous nodes with three types of relationships between them, and keeps growing with newly retrieved nodes and identified relationships every day. To construct the ontology, we propose the query-title interaction graph to represent the correlations (such as adjacency or syntactical dependency) between the tokens in correlated queries and document titles. Furthermore, we propose the GCTSP-Net to

extract multi-type phrases from the query-title interaction graph, as well as recognize the key entities, triggers or locations in events. After constructing the attention ontology by our models, we apply it to different applications, including document tagging, story tree formation, as well as recommendation. We run extensive experiments and analysis to evaluate the quality of the constructed ontology and the performance of our new algorithms. Results show that our approach outperforms a variety of baseline methods on three tasks. In addition, our attention ontology significantly improves the CTR of news feeds recommendation in a real-world application.

Part III

Text Generation: Asking Questions for Machine Reading Comprehension



Automatic question generation is an important technique that can improve the training of question answering, help chatbots to start or continue a conversation with humans, and provide assessment materials for educational purposes. It is critical to both human and machine intelligence.

In chapter 8, we propose our Clue Guided Copy Network for Question Generation (CGC-QG), which is a sequence-to-sequence generative model with copying mechanism, yet employing a variety of novel components and techniques to boost the performance of question generation. Our model explicitly predicts the potential clue words in an input sentence through a clue prediction module, and learns to copy potential clue words and generate questions through a sequence-to-sequence model with copy mechanism.

The Clue Guided Copy Network for Question Generation is a type of answer-aware question generation model. For answer-aware question generation models, they are ineffective at generating a large number of high-quality question-answer pairs from unstructured text, since given an answer and an input passage, question generation is inherently a one-to-many mapping problem. Therefore, in chapter 9, we further propose Answer-Clue-Style-aware Question Generation (ACS-QG), a novel system aimed at automatically generating diverse and high-quality question-answer pairs from unlabeled text corpus at scale by mimicking the way a human asks questions. Instead of predicting the clue words for question generation like what we did in chapter 8, we select appropriate answers and correlated clues from unlabeled text, and utilize them as inputs to generate questions

Chapter 8

Learning to Generate Questions by Learning What not to Generate

Automatic question generation is an important technique that can improve the training of question answering, help chatbots to start or continue a conversation with humans, and provide assessment materials for educational purposes. Existing neural question generation models are not sufficient mainly due to their inability to properly model the process of how each word in the question is selected, i.e., whether repeating the given passage or being generated from a vocabulary. In this chapter, we propose our Clue Guided Copy Network for Question Generation (CGC-QG), which is a sequence-to-sequence generative model with copying mechanism, yet employing a variety of novel components and techniques to boost the performance of question generation. In CGC-QG, we design a multi-task labeling strategy to identify whether a question word should be copied from the input passage or be generated instead, guiding the model to learn the accurate boundaries between copying and generation. Furthermore, our input passage encoder takes as input, among a diverse range of other features, the prediction made by a clue word predictor, which helps identify whether each word in the input passage is a potential clue to be copied into the target question. The clue word predictor is designed based on a novel application of Graph Convolutional Networks onto a syntactic dependency tree representation of each passage, thus being able to predict clue words only based on their context in the passage and their relative positions to the answer in the tree. We jointly train the clue prediction as well as question generation with multi-task learning and a number of practical strategies to reduce the complexity. Extensive evaluations show that our model significantly improves the performance of question generation and out-performs all previous state-of-the-art neural question generation models by a substantial margin.

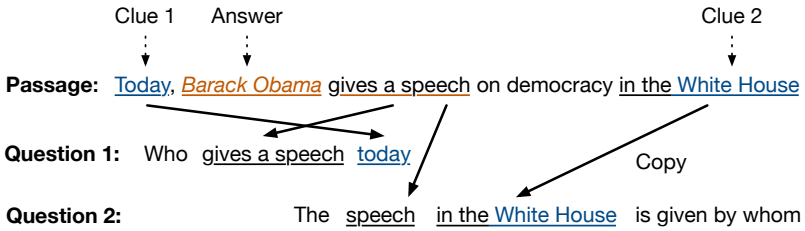


Figure 8.1: Questions are often asked by repeating some text chunks in the input passage, while there is great flexibility as to which chunks are repeated.

8.1 Introduction

Asking questions plays a vital role for both the growth of human beings and the improvement of artificial intelligent systems. As a dual task of question answering, question generation based on a text passage and a given answer has attracted much attention in recent years. One of the key applications of question generation is to automatically produce question-answer pairs to enhance machine reading comprehension systems [Du *et al.*, 2017; Yuan *et al.*, 2017; Tang *et al.*, 2017; Tang *et al.*, 2018]. Another application is generating practice exercises and assessments for educational purposes [Heilman and Smith, 2010; Heilman, 2011; Danon and Last, 2017]. Besides, question generation is also important in conversational systems and chatbots such as Siri, Cortana, Alexa and Google Assistant, helping them to kick-start and continue a conversation with human users [Mostafazadeh *et al.*, 2016].

Conventional methods for question generation rely on heuristic rules to perform syntactic transformations of a sentence to factual questions [Heilman, 2011; Chali and Hasan, 2015], following grammatical and lexical analysis. However, such methods require specifically crafted transformation and generation rules, with low generalizability. Recently, various neural network models have been proposed for question generation [Du *et al.*, 2017; Zhou *et al.*, 2017; Hu *et al.*, 2018; Kim *et al.*, 2018; Gao *et al.*, 2018]. These models formulate the question generation task as a sequence-to-sequence (Seq2Seq) neural learning problem, where different types of encoders and decoders have been designed. Like many other text generation tasks, the copying or pointer mechanism [Gulcehre *et al.*, 2016] is also widely adopted in question generation to handle the copy phenomenon between input passages and output questions, e.g., [Zhou *et al.*, 2017; Yuan *et al.*, 2017; Subramanian *et al.*, 2017; Kim *et al.*, 2018; Song *et al.*, 2018*a*]. However, we point out that a common limitation that hurdles question generation models mainly use copying mechanisms to handle out-of-vocabulary (OOV) issues, and fail to mark a clear boundary between the set of question words that should be directly copied from the input text and those that should be generated

instead.

In this chapter, we generate questions by learning to identify where each word in the question should come from, i.e., whether generated from a vocabulary or copied from the input text, and given the answer and its context, what words can potentially be copied from the input passage. People usually repeat text chunks in an input passage when asking questions about it, and generate remaining words from their own language to form a complete question. For example, in Fig. 8.1, given an input passage "Today, Barack Obama gives a speech on democracy in the White House" and an answer "Barack Obama", we can ask a question "The speech in the White House is given by whom?" Here the text chunks "speech" and "in the White House" are copied from the input passage. However, in the situation that a vocabulary word is an overlap word between a passage and a question, existing models do not clearly identify the underlying reason about the overlapping. In other words, whether this word is copied from the input or generated from vocabulary is not properly labeled. For example, some approaches [Zhou *et al.*, 2017] only take out-of-vocabulary (OOV) words that are shared by both the input passage and target question as copied words, and adopt a copying mechanism to learn to copy these OOV words into questions. In our work, given a passage and a question in a training dataset, we label a word as copied word if it is a nonstop word shared by both the passage and the question, and its word frequency rank in the vocabulary is lower than a threshold. We further aggressively shortlist the output vocabulary based on frequency analysis of the non-copied question words (according to our labeling criteria). Combining this labeling strategy with target vocabulary reduction, during the process of question generation, our model can make better decisions about when to copy or generate, and predict what to generate more accurately.

After applying the above strategies, a remaining problem is which word we should choose to copy. For example, in Fig. 8.1, we can either ask the question "Who gives a speech today?" or the question "The speech in the White House is given by whom?", where the two questions are related to two different copied text chunks "today" and "White House". We can see that asking a question about a passage and a given answer is actually a "one-to-many" mapping problem: we can ask questions in different ways based on which subset of words we choose to copy from input. Therefore, how to enable a neural model to learn what to copy is a critical problem. To solve this issue, we propose to predict the potential clue words in input passages. A word is considered as a "clue word" if it is helpful to reduce the uncertainty of the model about how to ask a question or what to copy, such as "White House" in question 2 of Fig. 8.1. In our model, we directly use our previously mentioned copy word labeling strategy to

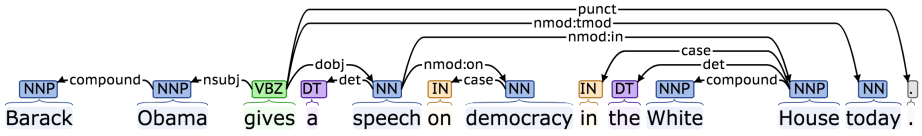


Figure 8.2: An example to show the syntactic structure of an input sentence. Clue words "White House" and "today" are close to the answer chunk "Barack Obama" with respect to the graph distance, though they are not close to each other in terms of the word order distance.

assign a binary label to each input word to indicate whether it is a clue word or not.

To predict the potential clue words in input, we have designed a novel clue prediction model that combines the syntactic dependency tree of an input with Graph Convolutional Networks. The intuition underlying our model design is that: the words copied from an input sentence to an output question are usually closely related to the answer chunk, and the patterns of the dependency paths between the copied words and answer chunks can be captured via Graph Convolutional Networks. Combining the graphical representation of input sentences with GCN enables us to incorporate both word features, such as word vector representation, Named Entity (NE) tags, Part-of-Speech (POS) tags and so on, and the syntactic structure of sentences for clue words prediction.

To generate questions given an answer chunk and the predicted clue words distribution in an input sentence, we apply the Seq2Seq framework which contains our feature-rich sentence encoder and a decoder with attention and copy mechanism. The clue word prediction results are incorporated into the encoder by a binary feature embedding. Based on our multi-task labeling strategy, i.e., labeling for both clue prediction and question generation, we jointly learn different components of our model in a supervised manner.

We performed extensive evaluation on two large question answering datasets: the SQuAD dataset v1.1 [Rajpurkar *et al.*, 2016] and the NewsQA dataset [Trischler *et al.*, 2016]. For each dataset, we use the answer chunk and the sentence that contains the answer chunk as input, and try to predict the question as output. We compared our model with a variety of existing rule-based and neural network based models. Our approach achieves significant improvement by jointly learning the potential clue words distribution for copy and the encoder-decoder framework for generation, and out-performs state-of-the-art approaches.

Passage:
The Soviet Union and *the People's Republic of China* supported post–World War II communist movements in foreign nations and colonies to advance their own interests, but were not always successful.

Question:
Who along with Russia supported post WW-II communist movements?

Answer:
the People's Republic of China

Figure 8.3: An example from the SQuAD dataset. Our task is to generate questions given an input passage and an answer. In SQuAD dataset, answers are sub spans of the passages.

8.2 Problem Definition and Motivation

In this section, we formally introduce the problem of question generation, and illustrate the motivation behind our work.

8.2.1 Answer-aware Question Generation

Let us denote a passage by P, a question related to this passage by Q, and the answer of that question by A. A passage can be either an input sentence or a paragraph, based on different datasets. A passage consists of a sequence of words $P = \{p_t\}_{t=1}^{|P|}$ where $|P|$ denotes the length of P. A question $Q = \{q_t\}_{t=1}^{|Q|}$ contains words from either a predefined vocabulary V or from the input text P. The task is finding the most probable question $\hat{Q}$ given an input passage and an answer:

$$\hat{Q} = \arg\max_{Q} prob(Q|P, A). \tag{8.1}$$

Fig. 8.3 shows an example of the dataset used in our work. Note that the answer A of the question Q is limited to sub spans of the input passage. However, our work can be easily adapted to the cases where the answer is not a sub span of the input passage by adding an extra answer encoder.

8.2.2 What to Ask: Clue Word Prediction

Even given the answer of a desired question, the task of question generation is still not a one-to-one mapping, as there are potentially multiple aspects related to the

given answer. For example, in Fig. 8.2, given the answer chunk "Barack Obama", the questions "The speech in the White House is given by whom?", "Who gives a speech on democracy?", and "Today who gives a speech?" are all valid questions. As we can see, although these questions share the same answer, they can be asked in different ways based on which word or phrase we choose to copy (e.g., "White House", "democracy", or "today").

To resolve this issue, we propose to predict the potential "clue words" in the input passage. A "clue word" is defined as a word or phrase that is helpful to guide the way we ask a question, such as "White House", "democracy", and "today" in Fig. 8.2. We design a clue prediction module based on the syntactical dependency tree representation of passages and Graph Convolutional Networks. Graph Convolutional Networks generalize Convolutional Neural Networks to graph-structured data, and have been successfully applied to natural language processing tasks, including semantic role labeling [Marcheggiani and Titov, 2017], document matching [Liu *et al.*, 2018*a*; Zhang *et al.*, 2018*b*], relation extraction [Zhang *et al.*, 2018*c*], and so on. The intuition underlying our model design is that clue words will be more closely connected to the answer in dependency trees, and the syntactical connection patterns can be learned via graph convolutional networks. For example, in Fig. 8.2, the graph path distances between answer "Barack Obama" to "democracy", "White House", and "today" are 3, 3, and 2, while the corresponding word order distances between them are 5, 8, and 10, respectively.

8.2.3 How to Ask: Copy or Generate

Another important problem of question generation is when to choose a word from the target vocabulary and when to copy a word from the input passage during the generation process. People tend to repeat or paraphrase some text pieces in a given passage when they ask a question about it. For instance, in Fig. 8.3, the question "Who along with Russia supported post WW-II communist movements", the text pieces "supported", "post", and "communist movements" are repeating the input passage, while "Russia" and "WW-II" are synonymous replacements of input phrases "The Soviet Union" and "World War II". Therefore, the copied words in questions should not be restricted to out-of-vocabulary words. Although this phenomenon is well-known by existing approaches, they do not properly and explicitly distinguish whether a word is from copy or from generate.

In our model, we consider the non-stop words that are shared by both an input passage and the output question as clue words, and encourage the model to copy clue

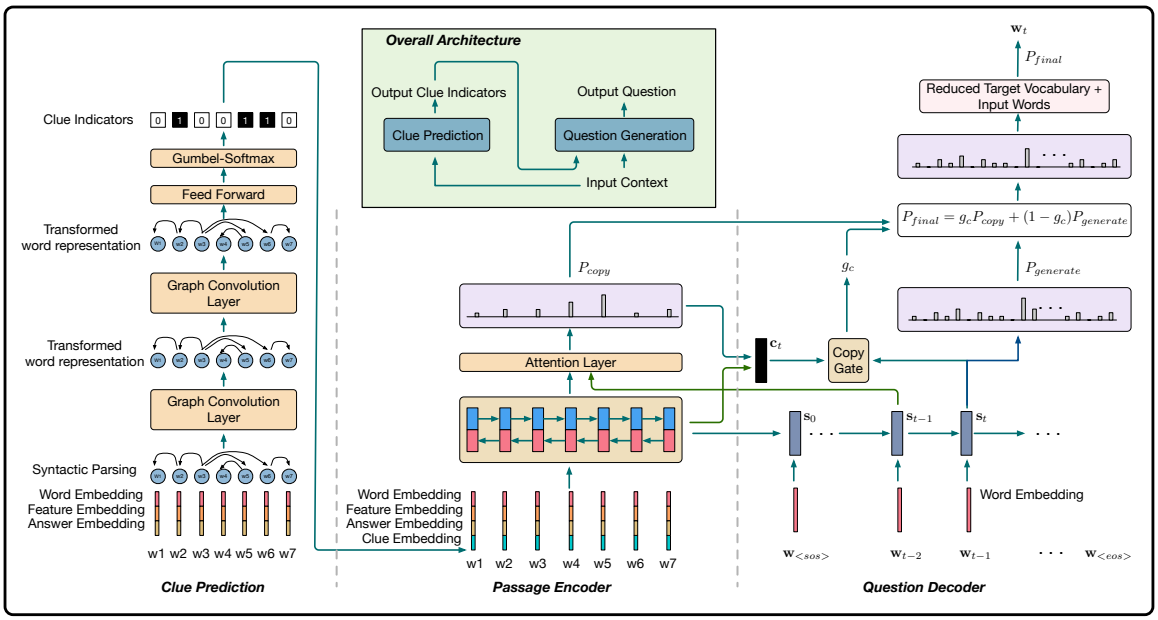


Figure 8.4: Illustration of the overall architecture of our proposed model. It contains a GCN-based clue word predictor, a masked feature-rich encoder for input passages, and an attention-and-copy-based decoder for generating questions.

words from input. After labeling clue words, we train a GCN-based clue predictor to learn whether each word in source text can be copied to the target question. The predicted clue words are further fed into a Seq2Seq model with attention and copy mechanism to help with question generation. Besides, different from existing approaches that share a vocabulary between source passages and target questions, we reduce the size of the target vocabulary of questions to be smaller than source passages and only include words with word frequency higher than a threshold. The idea is that the low-frequency words in questions are usually copied from source text rather than generated from a vocabulary. By combining the above strategies, our model is able to learn when to generate or copy a word, as well as which words to copy or generate during the progress of question generation. We will describe our operations in more detail in the next section.

8.3 Model Description

In this section, we introduce our proposed framework in detail. Similar to [Zhou *et al.*, 2017; Serban *et al.*, 2016; Du *et al.*, 2017], our question generator is based on an encoder-decoder framework, with attention and copying mechanisms incorporated. Fig. 8.4 illustrates the overall architecture and detailed components in our model.

Our model consists of three components: the clue word predictor, passage encoder and question decoder.

The clue word predictor predicts potential clue words that may appear a target question, based on the specific context of the input passage (without knowing the target question). It utilizes a syntactic dependency tree to reveal the relationship of answer tokens relative to other tokens in a sentence. Based on the tree representation of the input passage, we predict the distribution of clue words by a GCN-based encoder applied on the tree and a Straight-Through (ST) Gumbel-Softmax estimator [Jang *et al.*, 2016] for clue word sampling. The outputs of the clue word predictor are fed to the passage encoder to advise the encoder about what words may potentially be copied to the target question.

The passage encoder incorporates both the predicted clue word distribution and a variety of other feature embeddings of the input words, such as lexical features, answer position indicators, etc. Combined with a proposed low-frequency masking strategy (a shortlist strategy to reduce complexity on tuning input word embeddings), our encoder learns to better capture the useful input information with fewer trainable parameters.

Finally, the decoder jointly learns the probabilities of generating a word from vocabulary and copying a word from the input passage. At the decoder side, we introduce a multitask learning strategy to intentionally encourage the copying behavior in question generation. This is achieved by explicitly labeling the copy gate with a binary variable when a (non-stop) word in the question also appears in the input passage in the training set. Other multi-task labels are also incorporated to accurately learn when and what to copy. We further shortlist the target vocabulary based on the frequency distribution of non-overlap words. These strategies, assisted by the encoded features, help our model to clearly learn which word in the passage is indeed a clue word to be copied into the target question, while generating other non-clue words accurately.

The entire model is trained end-to-end via multitask learning, i.e., by minimizing a weighted sum of losses associated with different labels. In the following, we first introduce the encoder and decoder, followed by a description of the clue word predictor.

8.3.1 The Passage Encoder with Masks

Our encoder is based on bidirectional Gated Recurrent Unit (BiGRU) [Chung *et al.*, 2014], taking word embeddings, answer position indicators, lexical and frequency

features of words, as well as the output of the clue word predictor as the input. Specifically, for each word p_i in input passage P, we concatenate the following features to form a concatenated representation w_i to be input into the encoder:

- **Word Vector**. We initialize each word vector by Glove embedding [Pennington *et al.*, 2014]. If a word is not covered by Glove, we initialize its word vector randomly.

- **Lexical Features**. We perform Named Entity Recognition (NER), Part-of-Speech (POS) tagging and Dependency Parsing (DEP) on input passages using spaCy [Matthew Honnibal, 2015], and concatenate the lexical feature embedding vectors.

- **Binary Features**. We check whether each word is lowercase or not, whether it is a digit or like a number (such as word "three"), and embed these features by vectors.

- **Answer Position**. Similar to [Zhou *et al.*, 2017], we utilize the B/I/O tagging scheme to label the position of a given answer, where a word at the beginning of an answer is marked with **B**, **I** denotes the continuation of the answer, while words not contained in an answer are marked with **O**.

- **Word Frequency Feature**. We derive the word vocabulary from passages and questions in the training dataset. We then rank all the words in a descending order in terms of word frequencies, such that the first word is the most frequent word. The top r_h words are labeled as frequent words. Words ranked between r_h and r_l are labeled as intermediate words. The remaining with rank lower than r_l are labeled as rare words, where r_h and r_l are two predefined thresholds. In this way, each word will be assigned a frequency tag **L** (low frequency), **H** (highly frequent) or **M** (medium frequency).

- **Clue Indicator Feature**. In our model, the clue predictor (which we will introduce in more detail in Sec. 8.3.3) assigns a binary value to each word to indicate whether it is a potential clue word or not.

Denote an input passage by $P = (w_1, w_2, \cdots, w_{|P|})$. The BiGRU encoder reads the input sequence $w_1, w_2, \cdots, w_{|P|}$ and produces a sequence of hidden states $h_1, h_2, \cdots, h_{|P|}$ to represent the passage P. Each hidden state is a concatenation of a forward repre-

sentation and a backward representation:

$$h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i], \tag{8.2}$$

$$\overrightarrow{h}_i = \mathbf{BiGRU}(w_i, \overrightarrow{h}_{i-1}), \tag{8.3}$$

$$\overleftarrow{h}_i = \mathbf{BiGRU}(w_i, \overleftarrow{h}_{i+1}), \tag{8.4}$$

where $\overrightarrow{h}_i$ and $\overleftarrow{h}_i$ are the forward and backward hidden states of the i-th token in P, respectively.

Furthermore, rather than learning the full representation w_i for every word in the vocabulary, we use a masking strategy to replace the word embeddings of low-frequency words with a special <l> token, such that the information of low-frequency words is only represented by its answer/clue indicators and other augmented feature embeddings, except the word vectors. This strategy can improve performance due to two reasons. First, the augmented tagging features of a low-frequency word tend to be more influential than the word meaning in question generation. For example, given a sentence "<PERSON> likes playing football.", a question that can be generated is "What does <PERSON> like to play?"—what the token "<PERSON>" exactly is does not matter. This way, the masking strategy helps the model to omit the fine details that are not necessary for question generation. Second, the number of parameters that need be learned, especially the number of word embeddings that need be tuned, is largely reduced. Therefore, masking out unnecessary word embeddings while only keeping the corresponding augmented features and indicators does not hurt the performance of question generation. It actually improves training by reducing the model complexity.

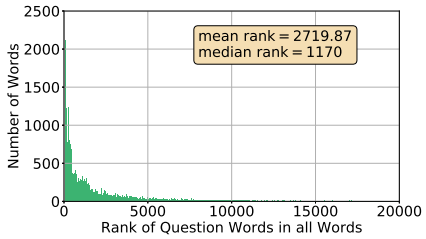
8.3.2 The Question Decoder with Aggressive Copying

In the decoding stage, we utilize another GRU with copying mechanism to generate question words sequentially based on the encoded input passage representation and previously decoded words. We first initialize the hidden state of the decoder GRU by passing the last backward encoder hidden state $\overleftarrow{h}_1$ to a linear layer:

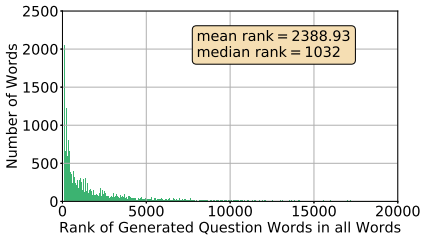
$$s_0 = tanh(\mathbf{W}_0 \overleftarrow{h}_1 + b). \tag{8.5}$$

For each decoding time step t, the GRU reads the embedding of the previous word w_{t-1}, previous attentional context vector c_{t-1}, and its previous hidden state s_{t-1} to calculate its current hidden state:

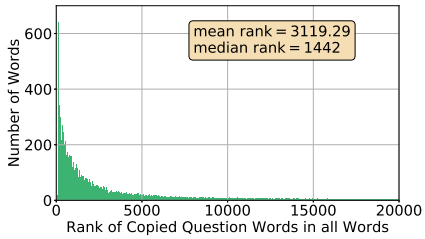
$$s_t = \mathbf{GRU}([w_{t-1}; c_{t-1}], s_{t-1}). \tag{8.6}$$



(a) Rank Distribution of All Question Words



(b) Rank Distribution of Generated Question Words



(c) Rank Distribution of Copied Question Words

Figure 8.5: Comparing the rank distributions of all question words, words from generation, and words from copy.

The context vector c_t for time step t is generated through the concatenated attention mechanism [Luong *et al.*, 2015]. Attention mechanism calculates a semantic match between encoder hidden states and the decoder hidden state. The attention weights indicate how the model spreads out the amount it cares about different encoder hidden states during decoding. At time step t, the attention weights and the context vector are calculated as:

$$e_{t,i} = v^{\intercal} tanh(\mathbf{W}_s s_t + \mathbf{W}_h h_i), \tag{8.7}$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j=1}^{|P|} \exp(e_{t,j})}, \tag{8.8}$$

$$c_t = \sum_{i=1}^{|P|} \alpha_{t,i} h_i. \tag{8.9}$$

Combining the previous word embedding w_{t-1}, the current decoder state s_t and the current context vector c_t, we can calculate a readout state r_t by an MLP maxout layer with dropouts [Goodfellow *et al.*, 2013]. Then the readout state is passed to a linear layer and a softmax layer to predict the probabilities of the next word over the

decoder vocabulary:

$$r_t = \mathbf{W}_{rw} w_{t-1} + \mathbf{W}_{rc} c_t + \mathbf{W}_{rs} s_t \tag{8.10}$$

$$m_t = [\max\{r_{t,2j-1}, r_{t,2j}\}]^{\intercal}_{j=1,\ldots,d} \tag{8.11}$$

$$p(y_t|y1, \cdots, y_{t-1}) = \text{softmax}(\mathbf{W}_o m_t), \tag{8.12}$$

where r_t is a 2-D vector.

The above module generates question words from a given vocabulary. Another important method to generate words is copying from source text. Copy or point mechanism [Gulcehre *et al.*, 2016] was introduced into sequence-to-sequence models to allow the copying of unknown words from the input text, which solves the out-of-vocabulary (OOV) problem. When decoding at time step t, the probability of copying is given by:

$$g_c = \sigma(\mathbf{W}_{cs} s_t + \mathbf{W}_{cc} c_t + b), \tag{8.13}$$

where σ is the Sigmoid function, and g_c is the probability of copying. For the copy probability of each input word, we reuse the attention weights given by Equation (8).

Copying mechanism has also been used in question generation [Zhou *et al.*, 2017; Song *et al.*, 2018*a*; Hu *et al.*, 2018], however, mainly to solve the OOV issue. Here we leverage the copying mechanism to enable the copying of potential clue words, instead of being limited to OOV words, from input. Different from existing methods, we take a more aggressive approach to train the copying mechanism via multitask learning based on different labels. That is, when preparing the training dataset, we explicitly label a word in a target question as a word copied from the source passage text if it satisfies all the following criteria:

i) it appears in both source text and target question;

ii) it is not a stop word;

iii) its frequency rank in the vocabulary is lower than a threshold r_h.

The remaining words in the question are considered as being generated from the vocabulary. Such a binary label (copy or not copy), together with which input word the question word is copied from, as well as the target question, are fed into different parts of the decoder as labels for multi-task model training. This is to intentionally encourage the copying of potential clue words into the target question rather than generating them from a vocabulary.

The intuition behind such an aggressive copying mechanism can be understood by checking the frequency distributions of both generated words and copied words

(as defined above) in the training dataset of SQuAD. Fig. 8.5 shows the frequency distributions of all question words, and then of generated question words and copied question words. The mean and median rank of generated words are 2389 and 1032, respectively. While for copied words, they are 3119 and 1442. Comparing Fig. 8.5(b) with Fig. 8.5(c), we can see that question words generated from the vocabulary tends to be clustered toward high ranked (or frequent) words. On the other hand, the fraction of low ranked (or infrequent) words in copied words are much greater than that in generated words. This means the generated words are mostly from frequent words, while the majority of low-frequency words in the long tail are copied from the input, rather than generated.

This phenomenon matches our intuition: when people ask a question about a given passage, the generated words tend to be commonly used words, while the repeated text chunks from source passage may contain relatively rare words such as names and dates. Based on this observation, we further propose to reduce the vocabulary at the decoder for target question generation to be top N frequently generated words, where N is a predefined threshold that varies according to datasets.

8.3.3 A GCN-Based Clue Word Predictor

Given a passage and some answer positions, our clue word predictor aims to identify the clue words in the passage that can help to ask a question and are also potential candidates for copying, by understanding the semantic context of the input passage. Again, in the training dataset, the non-stop words that are shared by both an input passage and an output question are aggressively labeled as *clue words*.

We note that clue words are, in fact, more closely connected to the answer chunk in the form of syntactic dependency trees than in word sequences. Fig. 8.6 shows our observation on the SQuAD dataset. For each training example, we get the nonstop words that appear in both the input passage and the output question. For each clue word in the training set, we find its shortest undirected path to the answer chunk based on the dependency parsing tree of the passage. For each jump on the shortest path, we record the dependency type. We also calculate the distance between each clue word and the answer in terms of the number of words between them. As shown in Fig. 8.6(a), *prep*, *pobj*, and *nsubj* appear frequently on these shortest paths. Comparing Fig. 8.6(b) with Fig. 8.6(c), we can see that the distances in terms of dependency trees are much smaller than those in terms of sequential word orders. The average and median distances on the dependency trees are 4.41 and 4, while those values are 10.23 and 7 for sequential word distances.

In order to predict the positions of clue words based on their dependencies on the answer chunk (without knowing the question which is yet to be generated), we use a Graph Convolutional Network (GCN) to convolve over the word features on the dependency tree of each passage, as shown in Fig. 8.4. The predictor consists of four layers:

Embedding layer. This layer shares the same features with the passage encoder, except that it does not include the clue indicators. Therefore, each word is represented by its word embedding, lexical features, binary features, the word frequency feature, and an answer position indicator.

Syntactic dependency parsing layer. We obtain the syntactic dependency parsing tree of each passage by spaCy [Matthew Honnibal, 2015], where the dependency edges between words are directed. In our model, we use the syntactic structure to represent the structure of passage words.

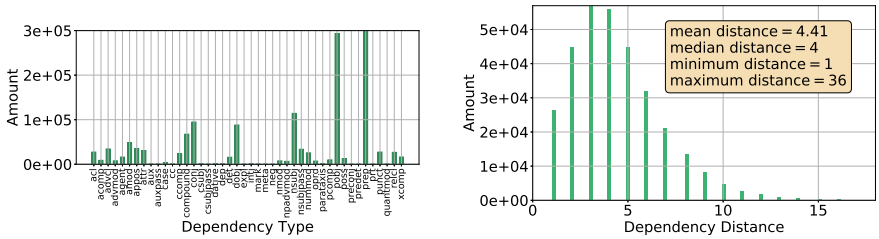
Encoding layer. The objective of the encoding layer is to encode the context information into each word based on the dependency tree. We utilize a multi-layered GCN to incorporate the information of neighboring word features into each vertex, which is a word. After L GCN layers, the hidden vector representation of each word will incorporate the information of its neighboring words that are no more than L hops away in the dependency tree.

Output layer. After obtaining the context-aware representation of each word in the passage, we calculate the probability of each word being a clue word. A linear layer is utilized to calculate the unnormalized probabilities. We subsequently sample the binary clue indicator for each word through a Gumbel-Softmax layer, given the unnormalized probabilities. A sampled value of 1 indicates that the word is predicated as a clue word.

GCN Operations on Dependency Trees

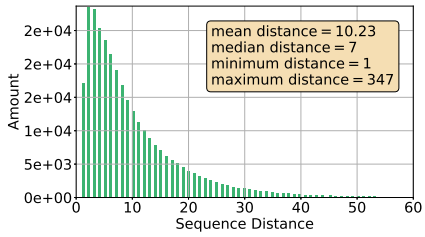
We now introduce the operations performed in each GCN layer [Kipf and Welling, 2016; Zhang *et al.*, 2018*c*]. GCNs generalize the CNN from low-dimensional regular grids to high-dimensional irregular graph domains. In general, the input to a GCN is a graph $G = (\mathcal{V}, E)$ with N vertices $v_i \in \mathcal{V}$, and edges $e_{ij} = (v_i, v_j) \in E$. The edges can be weighted with weights w_{ij}, or unweighted. The input also contains a vertex feature matrix denoted by $X = \{\mathbf{x}_i\}_{i=1}^{N}$, where $\mathbf{x}_i$ is the *feature vector* of vertex v_i.

Since a dependency tree is also a graph, we can perform the graph convolution operations on dependency trees by representing each tree into its corresponding adjacency matrix form. Now let us briefly introduce the GCN propagation layers used in our model [Zhang *et al.*, 2018*c*]. The weighted adjacency matrix of the graph is



(a) Distribution of Syntactic Dependency Types between Copied Words and the Answer

(b) Distribution of Syntactic Dependency Distances between Copied Words and theAnswer



(c) Distribution of Sequential Word Distances between Copied Words and the Answer

Figure 8.6: Comparing the distributions of syntactic dependency distances and sequential word distances between copied words and the answer in each training sample.

denoted as $A \in \mathbb{R}^{N \times N}$ where $A_{ij} = w_{ij}$. For unweighted graphs, the weights are either 1 or 0. In an L-layer GCN, let $h_i^{(l-1)}$ denotes the input vector and $h_i^{(l)}$ denotes the output vector of node i at the l-th layer. We will utilize a multi-layer GCN with the following layer-wise propagation rule [Zhang *et al.*, 2018*c*]:

$$h_i^{(l)} = \sigma(\sum_{j=1}^{N} \tilde{\mathbf{A}}_{ij} \mathbf{W}^{(l)} h_j^{(l-1)} / d_i + b^{(l)}), \tag{8.14}$$

where σ is a nonlinear function (e.g., ReLU), and $W^{(l)}$ is a linear transformation. $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ where $\mathbf{I}_N$ is an $n \times n$ identity matrix. $d_i = \sum_{j=1}^{N} \tilde{\mathbf{A}}_{ij}$ is the degree of node (or word in our case) i in the graph (or dependency tree).

In our experiments, we treat the dependency trees as undirected, i.e., $\forall i, j, \mathbf{A}_{ij} = \mathbf{A}_{ji}$. Besides, as we already included the dependency type information in the embedding vectors of each word, we do not need to incorporate the edge type information in the adjacency matrix.

Stacking this operation by L layers gives us a deep GCN network. The input to the nodes in the first layer of the GCN are the feature vectors of words in the passage. After L layers of transformations, we can obtain a context-aware representation of each word. We then feed them into a linear layer to get the unnormalized probability of each word being a clue word. After that, the unnormalized probabilities are fed

to a Straight-Through (ST) Gumbel-Softmax estimator to sample an N-dimensional binary vector indicating whether each of the N words is a clue word or not.

Gumbel-Softmax

Gumbel-Softmax [Jang *et al.*, 2016] is a method of sampling discrete random variables in neural networks. It approximates one-hot vectors sampled from a categorical distribution by making them continuous, therefore the gradients of model parameters can be calculated using the reparameterization trick and the standard backpropagation. Gumbel-Softmax distribution is motivated by Gumbel-Max trick [Maddison *et al.*, 2014], an algorithm for sampling from a categorical distribution. Let $(p_1, ..., p_k)$ denotes a k-dimensional categorical distribution where the probability p_i of class i is defined as:

$$p_i = \frac{\exp(\log(\pi_i))}{\sum_{j=1}^{k} \exp(\log(\pi_j))}, \tag{8.15}$$

where π_i is the unnormalized log probability of class i. We can easily draw a one-hot sample $\mathbf{z} = (z_1, \cdots, z_k) \in \mathbb{R}^k$ from the distribution by the following equations:

$$z_i = \begin{cases} 1, \ i = \arg\max_j (log(\pi_j) + g_j) \\ 0, \ \text{otherwise} \end{cases} \tag{8.16}$$

$$g_i = -\log(-\log(u_i)), \tag{8.17}$$

$$u_i \sim \text{Uniform}(0, 1) \tag{8.18}$$

where g_i is Gumbel noise used to perturb each $\log(\pi_i)$. In this way, taking arg max is equivalent to drawing a sample using probabilities $(p_1, \cdots, p_k)$.

Gumbel-Softmax distribution replaces the arg max function by differentiable softmax function. Therefore, a sample $\mathbf{y} = (y_1, \cdots, y_k)$ drawn from Gumbel-Softmax distribution is given by:

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau)}, \tag{8.19}$$

where τ is a temperature parameter. The Gumbel-Softmax distribution resembles the one-hot sample when τ diminishes to zero.

Straight-Through (ST) Gumbel-Softmax estimator [Jang *et al.*, 2016] is a discrete version of the continuous Gumbel-Softmax estimator. It takes different paths in the forward and backward propagation. In the forward pass, it discretizes a continuous probability vector $\mathbf{y}$ sampled from the Gumbel-Softmax distribution into a one-hot

Table 8.1: Description of evaluation datasets.

Dataset	Train	Dev	Test	P-Length*	Q-Length*	A-Length*
SQuAD	86,635	8,965	8,964	32.72	11.31	3.19
NewsQA	77,538	4,341	4,383	28.14	7.82	4.60

* P-Length: average number of tokens of passages.
* Q-Length: average number of tokens of questions.
* A-Length: average number of tokens of answers.

vector $\mathbf{y}^{ST} = (y_1^{ST}, \cdots, y_k^{ST})$ by:

$$y_i^{ST} = \begin{cases} 1, & i = \arg\max_j y_j, \\ 0, & \text{otherwise.} \end{cases} \tag{8.20}$$

In the backward pass, it uses the continuous $\mathbf{y}$, so that the error signal can still backpropagate.

Using the ST Gumbel-Softmax estimator, our model is able to sample a binary clue indicator vector for an input passage. Then the clue indicator vector is fed into the passage encoder for question generations, as shown in Fig. 8.4.

8.4 Evaluation

In this section, we evaluate the performance of our proposed models on the SQuAD dataset and the NewsQA dataset, and compare them with state-of-the-art question generation models.

8.4.1 Datasets, Metrics and Baselines

The SQuAD dataset is a reading comprehension dataset, consisting of questions posed by crowd-workers on a set of Wikipedia articles, where the answer to every question is a segment of text from the corresponding reading passage. SQuAD 1.1 is used in our experiment containing 536 Wikipedia articles and more than 100K question-answer pairs. When processing a sample from dataset, instead of using the entire document, we take the sentence that contains the answer as the input. Since the test set is not publicly available, we use the data split proposed by [Zhou *et al.*, 2017] where the original dev set is randomly split into a dev test and a test set of equal size.

In the NewsQA dataset, there are 120K questions and their corresponding answers as well as the documents that are CNN news articles. Questions are written by questioners in natural language with only the headlines and highlights of the articles available to them. With the information of the questions and the full articles, answerers select related sub-spans from the passages of the source text and mark them as answers. Multiple answers may be provided to a same question by different answerers

and they are ranked by validators based on the quality of the answers. In our experiment, we picked a subset of NewsQA where answers are top-ranked and are composed of a contiguous sequence of words within the input sentence of the document.

Table 8.1 shows the number of samples in each set and the average number of tokens of the input sentences, questions, and answers listed in columns P-Length, Q-Length, and A-Length respectively.

We report the evaluation results with following metrics.

- **BLEU** [Papineni *et al.*, 2002]. BLEU measures precision by how much the words in prediction sentences appear in reference sentences at the corpus level. BLEU-1, BLEU-2, BLEU-3, and BLEU-4, use 1-gram to 4-gram for calculation, respectively.
- **ROUGE-L** [Lin, 2004]. ROUGE-L measures recall by how much the words in reference sentences appear in prediction sentences using Longest Common Subsequence (LCS) based statistics.
- **METEOR** [Denkowski and Lavie, 2014]. METEOR is based on the harmonic mean of unigram precision and recall, with recall weighted higher than precision.

In the experiments, we have eight baseline models for comparison. Results reported on PCFG-Trans, MPQG, and NQG++ are from experiments we conducted using published code on GitHub. For other baseline models, we directly copy the reported performance given in their papers. We report all the results on the SQuAD dataset, and for the NewsQA dataset, we can only report the baselines with open source code available.

- **PCFG-Trans** [Heilman, 2011] is a rule-based system that generates a question based on a given answer word span.
- **MPQG** [Song *et al.*, 2018*a*] proposed a Seq2Seq model that matches the answer with the passage before generating the question.
- **SeqCopyNet** [Zhou *et al.*, 2018] proposed a method to improve the copying mechanism in Seq2Seq models by copying not only a single word but a sequence of words from the input sentence.
- **seq2seq+z+c+GAN** [Yao *et al.*, 2018] proposed a model employed in GAN framework using the latent variable to capture the diversity and learning disentangled representation using the observed variable.

- **NQG++** [Zhou *et al.*, 2017] proposed a Seq2Seq model with a feature-rich encoder to encode answer position, POS and NER tag information.
- **Answer-focused Position-aware model** [Sun *et al.*, 2018] incorporates the answer embedding to help generate an interrogative word matching the answer type. And it models the relative distance between the context words and the answer for the model to be aware of the position of the context words when generating a question.
- **s2sa-at-mp-gsa** [Zhao *et al.*, 2018] proposed a model which contains a gated attention encoder and a maxout pointer decoder to address the challenges of processing long text inputs. This model has a paragraph-level version and a sentence-level version. For the purpose of fair comparison, we report the results of the sentence-level model to match with our settings.
- **ASs2s** [Kim *et al.*, 2018] proposed an answer-separated Seq2Seq to identify which interrogative word should be used by replacing the target answer in the original passage with a special token.

For our models, we evaluate the following versions:

- **CGC-QG (no feature-rich embedding)**. We name our model as Clue Guided Copy for Question Generation (CGC-QG). In this variant, we only keep the embedding of words, answer position indicators, and clue indicators for each token, and remove the embedding vectors of other features.
- **CGC-QG (no target reduction)**. This model variant does not contain target vocabulary reduction operation.
- **CGC-QG (no clue prediction)**. The clue predictor and clue embedding are removed in model variant.
- **CGC-QG**. This is the complete version of our proposed model.

8.4.2 Experiment Settings

We implement our models in PyTorch 0.4.1 [Paszke *et al.*, 2017] and train the model with a single Tesla P40. We utilize spaCy [Matthew Honnibal, 2015] to perform dependency parsing and extract lexical features for tokens. As to the vocabulary, we collect it from the training dataset and keep the top 20K most frequent words for both datasets.

We set the threshold $r_h = 100$ and $r_l = 2000$. For target vocabulary reduction, we set $N = 2000$. The embedding dimension of word vector is set to be 300 and initialized by GloVe. The word vectors of words that are not contained in GloVe are initialized randomly. The word frequency features are embedded to 32-dimensional vectors, and other features and indicators are embedded to 16-dimensional vectors. All embedding vectors are trainable in the model. We use a single layer BiGRU with hidden size 512 for the encoder, and a single layer undirected GRU with hidden size 512 for the decoder. The dropout rate $p = 0.1$ is applied to the encoder, decoder, and the attention module.

During training, we optimize the Cross-Entropy loss function for clue prediction, question generation, and question copying, and perform gradient descent by the Adam [Kingma and Ba, 2014] optimizer with an initial learning rate $l_r = 0.001$, two momentum parameters are $\beta_1 = 0.8$ and $\beta_2 = 0.999$ respectively, and $\epsilon = 10^{-8}$. The mini-batch size for each update is set to 32 and model is trained for up to 10 epochs (as we found that usually the models derive the best performance after 6 or 7 epochs). We apply gradient clipping with range $[-5, 5]$ for Adam. Besides, exponential moving average is applied on all trainable variables with a decay rate 0.9999. When testing, beam search is conducted with beam width 20. The decoding process stops when a token $<EOS>$ that represents end of sentence is generated.

8.4.3 Main Results

Table. 8.2 and Table. 8.3 compare the performance of our model with existing question generation models on SQuAD and NewsQA in terms of different evaluation metrics. For the SQuAD dataset, we compare our model with all the baseline methods we have listed. As to the NewsQA dataset, since only a part of the baseline methods made their code public, we compare our model with approaches that have open source code. We can see that our model achieves the best performance on both datasets and significantly outperforms state-of-the-art algorithms. On the SQuAD dataset, given an input sentence and an answer, the BLEU-4, ROUGE-L, and METEOR of our result are 17.55, 44.53, and 21.24 respectively, while corresponding previous state-of-the-art results are 16.17, 44.24, and 19.67 from different approaches. Similarly, our method also gives a significantly better performance on NewsQA compared with the baselines in Table 8.3.

The reason is that we combine different strategies in our model to make it learn when to generate or copy a word, and what to generate or copy. First, our model learns to predict clue words through a GCN-based clue predictor. Second, our en-

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METEOR
PCFG-Trans*	28.77	17.81	12.64	9.47	31.68	18.97
SeqCopyNet	–	–	–	13.02	44.00	–
seq2seq+z+c+GAN	44.42	26.03	17.60	13.36	40.42	17.70
NQG++*	42.36	26.33	18.46	13.51	41.60	18.18
MPQG	–	–	–	13.91	–	–
Answer-focused Position-aware model	43.02	28.14	20.51	15.64	–	–
s2sa-at-mp-gsa	44.51	29.07	21.06	15.82	44.24	19.67
ASs2s	–	–	–	16.17	–	–
CGC-QG (no feature-rich embedding)	45.50	29.63	21.58	16.38	43.11	20.52
CGC-QG (no target reduction)	45.80	30.27	22.29	17.05	44.09	20.79
CGC-QG (no clue prediction)	45.58	30.07	22.08	16.80	44.51	20.80
CGC-QG	**46.58**	**30.90**	**22.82**	**17.55**	**44.53**	**21.24**

Experiments are conducted on baselines followed by a "*" using released source code. Results of other baselines are copied from their papers where unreported metrics are marked "–".

Table 8.2: Evaluation results of different models on SQuAD dataset.

coder incorporates a variety of embeddings of different features and clue indicators. Combining with the masking strategy, our model can better discover the relationship between input patterns and output patterns. Third, the reduced target vocabulary also helps our model to better capture when to copy or generate, and the generator is easier to train with a reduced vocabulary size. And most importantly, our new criteria of marking a question word as copied word (as described in Sec.8.3.3) helps the model to make better decisions on which path to go, i.e., to copy or to generate, during question generation. By incorporating part of these new strategies and modules into our model, we can achieve performance better than state-of-the-art models on SQuAD and NewsQA. With all these designs implemented, our model gives the best performance on both datasets.

There is a significant gap between the performance on SQuAD and on NewsQA due to the different characteristics of the datasets. The average answer length of NewsQA is 44.2% larger than it of SQuAD according to the statistics shown in Table 8.1. Long answers usually hold more information and are more difficult to generate questions. Furthermore, reference questions in NewQA tend to have less strict grammars and more diverse phrasings. To give a typical example, "Iran criticizes who?" is a reference question in NewsQA which does not start with an interrogative word but ends with one. These characteristics make the performance on NewsQA not as good as on SQuAD. However, our approach is still significantly better than the compared approaches on NewsQA dataset. It demonstrates that copy from the input is a general phenomenon across different datasets. Our model better captures what copied words are and what generated words are in a question based on our new criteria of labeling a question word as copied word or not.

Table 8.3: Evaluation results of different models on NewsQA dataset.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METEOR
PCFG-Trans*	16.90	7.94	4.72	3.08	23.78	13.74
MPQG*	35.70	17.16	9.64	5.65	39.85	14.13
NQG++*	40.33	22.47	14.83	9.94	42.25	16.72
CGC-QG (no feature-rich embedding)	39.35	22.10	14.36	9.99	42.00	16.60
CGC-QG (no target reduction)	40.00	22.84	15.01	10.52	42.33	16.89
CGC-QG (no clue prediction)	39.85	22.82	14.96	10.45	43.16	17.11
CGC-QG	**40.45**	**23.52**	**15.68**	**11.06**	**43.16**	**17.43**

Experiments are conducted on baselines followed by a "*" using released source code.

8.4.4 Analysis

We evaluate the impact of different modules in our model by ablation tests. Table 8.2 and Table 8.3 list the performance of our model variants with different sub-components removed.

When we remove the extra feature embeddings in our model, i.e., the embeddings of POS, NER, Dependency Types, word frequency levels (low-frequent, median-frequent, high-frequent), and binary features (whether it is lowercase, digit), the performance drops significantly. This is because the tags and feature embeddings represent each token in different aspects. The number of different tags is much smaller than the number of different words. Therefore, the patterns which can be learned from these tags and features are more obvious than what we can learn from word embeddings. Even though a well-trained word vector may contain the information of other features such as POS or NER, explicitly concatenating these feature embedding vectors helps the model to capture the patterns to ask a question more easily.

Removing the operation of target vocabulary reduction also hurts the performance of our model. As we discussed earlier, the non-overlap question words (or generated words) are mostly covered by the high frequency words. Reducing the target vocabulary size helps our model to better learn the probabilities of generating these words. On the other hand, it also encourages the model to better capture what they can copy from input text.

Finally, without the clue prediction module, the performance also drops on both datasets. This is because when given an answer span in a passage, asking a question about it is still a one-to-many mapping problem. Our clue prediction module learns how people select the related clue words to further reduce the uncertainty of how to ask a question by learning from a large training dataset. With predicted clue indicators incorporated into the encoder of generator, our model can fit the way how people ask questions in the dataset.

8.5 Conclusion

In this chapter, we demonstrate the effectiveness of teaching the model to make decisions during the question generation process on which words to generate and to copy. We label the nonstop and overlap words between input passages and questions as copy targets and use such labels to train our model. Besides, we further observe that the distribution of generated question words are mostly common words with relative high frequency. Based on this observation, we reduce the vocabulary size for generating question words. To help the model better capture how to ask a question and alleviate the issue of one-to-many mapping when asking a question, we propose a GCN-based clue prediction module to predict which part of words can be a clue word to ask a question given an answer. It utilizes the syntactic dependency tree representation of a passage to encode the information of each token in the passage, and sample a clue indicator for each token using a Straight-Through (ST) Gumbel-Softmax estimator. Our simulation results on the SQuAD dataset and NewsQA dataset show that our model outperforms a range of existing state-of-the-art approaches significantly.

Chapter 9

Asking Questions the Human Way: Scalable Question-Answer Generation from Text Corpus

In the previous chapter, we propose Clue Guided Copy Network for Question Generation. It is a type of answer-aware question generation model. For answer-aware question generation models, they are ineffective at generating a large number of high-quality question-answer pairs from unstructured text, since given an answer and an input passage, question generation is inherently a one-to-many mapping problem. A natural idea is: feeding the generative model with more input information to improve the quality and generated questions, as well as making the generation process more controllable. For example, instead of predicting the clue words for question generation like what we did in the previous chapter, why not just select appropriate answers and correlated clues from unlabeled text, and utilize them as inputs to generate questions?

In this chapter, we propose Answer-Clue-Style-aware Question Generation (ACS-QG), a novel system aimed at automatically generating diverse and high-quality question-answer pairs from unlabeled text corpus at scale by mimicking the way a human asks questions. Our system consists of: i) an information extractor, which samples multiple types of assistive information to guide question generation; ii) neural question generators, which generates diverse and controllable questions about a passage, utilizing the extracted assistive information as an input; and iii) a neural quality controller, which filters out low-quality generated data based on text entailment. We compare our question generation models with existing approaches and perform pilot user studies to evaluate the quality of the generated question-answer pairs. The evaluation results show that our system dramatically outperforms state-of-the-art neural question generation models in terms of the generation quality, while being scalable in the meantime. With models trained on a relatively smaller amount of data, we can

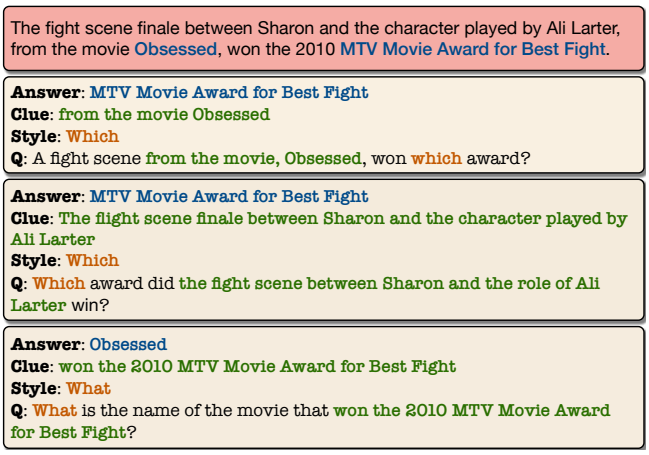


Figure 9.1: Given the same input sentence, we can ask diverse questions based on our different choices about i) what is the target answer; ii) which answer-related chunk is utilized as clue, and iii) what type of questions is asked.

generate 2.8 million quality-assured question-answer pairs from a million sentences in Wikipedia.

9.1 Introduction

Automatically generating question-answer pairs from unlabeled text passages is of great value to many applications, such as assisting the training of machine reading comprehension systems [Tang *et al.*, 2017; Tang *et al.*, 2018; Du and Cardie, 2018], generating queries/questions from documents to improve search engines [Han *et al.*, 2019], training chatbots to start or continue a conversation [Shum *et al.*, 2018], generating exercises for educational purposes [Heilman and Smith, 2010; Heilman, 2011; Danon and Last, 2017], and generating FAQs for web documents [Krishna and Iyyer, 2019]. In particular, many question-answering tasks such as machine reading comprehension and chatbots require a large amount of labeled data for supervised training, acquiring which is time-consuming and expensive. Automatic question-answer generation makes it possible to provide these systems with scalable data and to transfer the model well-trained on one domain with carefully labelled data to new domains.

Despite a large number of studies on neural question generation, it remains a significant challenge to generate high-quality QA pairs from unstructured text at large quantities. Most existing neural network models approach the question generation problem as answer-aware question generation, where the answer to the target question is provided as an input together with the context. Based on these inputs, they

formulate the task as a Sequence-to-Sequence (Seq2Seq) problem, and design various encoder, decoder, and input features to improve the quality of generated questions [Serban *et al.*, 2016; Du *et al.*, 2017; Liu *et al.*, 2019*b*; Zhou *et al.*, 2017; Song *et al.*, 2018*a*; Hu *et al.*, 2018; Du and Cardie, 2018]. However, the performance of answer-aware question generation models are far from satisfying, due to the one-to-many mapping nature of the task. Figure 9.1 shows an example of this phenomenon. Given the same input text "The fight scene finale between Sharon and the character played by Ali Larter, from the movie Obsessed, won the 2010 MTV Movie Award for Best Fight.", we can ask a variety of questions based on it. If we select the text chunk "MTV Movie Award for Best Fight" as the answer, we can still ask different questions such as "A fight scene from the movie, Obsessed, won which award?" or "Which award did the fight scene between Sharon and the role of Ali Larter win?" from different aspects.

We argue that when a human asks a question based on a passage, she will consider various factors. First, she will still select an answer as a major target that her question will point to. Second, she will decide which piece of information will be presented or rephrased in her question to set constraints or context for the question. We call this piece of information as *clue*, as the target answer may be related to different clues in the passage. Third, even the same question may be expressed in different styles (e.g., "what", "who", "why", etc.). For example, they can ask "which award" or "what is the name of the award" to express the same meaning. If we narrow down to answer, clue, and question style, the process of question generation will become much closer to a one-to-one mapping problem, essentially mimicking the human way of asking questions. Hence, introducing such information into question-answer generation will help to reduce the difficulty of the task.

In this chapter, we propose Answer-Clue-Style-aware Question Generation (ACS-QG) designed for scalable generation of high-quality question-answer pairs from unlabeled text corpus. Just like a human will ask a question with clue and style in mind, our system first automatically extracts multiple types of information from an input passage to assist question generation. Based on the extracted multi-aspect information, we design neural network models to generate diverse questions in a controllable way. Compared with existing answer-aware question generation, our approach essentially converts the one-to-many mapping problem into a one-to-one mapping problem, and is thus scalable by controlling the assistive information fed to the neural network while in the meantime ensuring the generation quality. Specifically, we have solved multiple challenges in the ACS-aware question generation system:

What to ask given an unlabeled passage? Given an input passage such as a

sentence, randomly selecting the combinations of *<answer, clue, style>* information will cause type mismatches and input volume explosion. On one hand, answer, clue, and style are dependent on each other. Without taking their correlations into account, we may select "how" or "when" as the target question style (or type) while a person's name is selected as the answer. Such errors are called type mismatches. On the other hand, input volume explosion means that we may randomly sample thousands of different *<answer, clue, style>* combinations without limitation while most of these inputs lead to meaningless questions.

To overcome these challenges, we design and implement an information extractor to efficiently sample meaningful inputs from a given passage. We learn the joint distribution of the three types of information from existing abundant reading comprehension datasets, such as SQuAD [Rajpurkar *et al.*, 2016]. In the meantime, we decompose the joint probability distribution of an *<answer, clue, style>* tuple into three components, and apply a three-step sampling mechanism to the input passage to select reasonable combinations of input information to feed into ACS-aware question generation. Based on this strategy, we can alleviate the type mismatch problem and narrow down the combinations of assistive information for meaningful question generation.

How to learn a model to ask ACS-aware questions? Existing neural approaches are mostly designed for answer-aware question generation, while there exists no dataset available for the task of ACS-aware question generation. In our system, we propose effective strategies to automatically construct datasets from existing reading comprehension datasets without any human labeling effort. We formally define "clue" as a semantic chunk in an input passage, which will be included or rephrased in the target question. Based on this definition, we perform syntactic parsing and chunking on input text, and select the chunk which is most relevant to the target question as the clue information. In this manner, we have leveraged the abundance of reading comprehension datasets to automatically construct training data for our new problem. On the other hand, we categorize different questions into 9 styles, including "what", "how", "yes-no" and so forth, and extract such information from existing datasets to train ACS-aware question generation models.

We propose two deep neural models for ACS-aware question generation, and show their superior performance in generating diverse and high-quality questions. The first model employs sequence-to-sequence framework with copy and attention mechanism [Sutskever *et al.*, 2014; Bahdanau *et al.*, 2014; Cao *et al.*, 2017], incorporating the information of answer, clue and style into the encoder and decoder. Furthermore, it differentiates content words and function words in the input, and utilizes vocabulary

reduction which downsizes the vocabularies for both the encoder and decoder to encourage aggressive copying. In the second model, we fine-tune a GPT2-small model [Radford *et al.*, 2019]. We train our ACS-aware QG models using the input passage, answer, clue, and question style as the language modeling context. As a result, we reduce the repeating output word phenomenon which usually exists in sequence-to-sequence models, and can generate questions with better readability. With multi-aspect assistive input information, our models are able to ask a variety of high-quality questions based on an input passage, while making the generation process controllable.

How to ensure the quality of generated QA pairs? We construct a data filter, which consists of an entailment model and a question answering model. In our filtering process, we input questions generated in the aforementioned manner into a BERT-based [Devlin *et al.*, 2018] question answering model to get its predicted answer span, and measure the answer overlap between our input answer span and the predicted answer span. In addition, we also classify the entailment relationship between the original sentence and the question-answer concatenation. These components allow us to remove low-quality QA pairs. By combining the input selector, ACS-aware question generator, and the data filter, we have constructed a pipeline that is able to generate a large number of QA pairs from unlabeled text without extra human labeling efforts.

We perform extensive experiments based on the SQuAD dataset [Rajpurkar *et al.*, 2016] and Wikipedia, and compare our ACS-aware question generation model with different existing approaches. Results show that both the content-separated seq2seq model with aggressive copying mechanism and the extra input information bring substantial benefits to question generation. Our method outperforms the state-of-the-art models significantly in terms of various metrics such as BLEU-4, ROUGE-L and METEOR.

With models trained on $86,635$ of SQuAD data, we can automatically generate two large datasets containing 1.33 million and 1.45 million QA pairs from a corpus of top-ranked Wikipedia articles. We perform quality evaluation on the generated datasets and identify their strengths and weaknesses. Finally, we also observe how our generated QA data performs in training question-answering models in machine reading comprehension, as an alternative means to assess the quality of the question-answer pairs generated.

9.2 Problem Formulation

In this section, we formally introduce the problem of ACS-aware question generation.

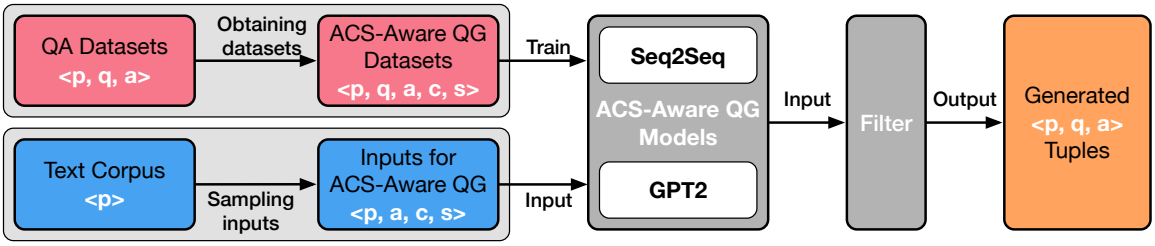


Figure 9.2: An overview of the system architecture. It contains a dataset constructor, information sampler, ACS-aware question generator and a data filter.

Denote a passage by p, where it can be either a sentence or a paragraph (in our work, it is a sentence). Let q denotes a question related to this passage, and a denotes the answer of that question. A passage consists of a sequence of words $p = \{p_t\}_{t=1}^{|p|}$ where $|p|$ denotes the length of p. A question $q = \{q_t\}_{t=1}^{|q|}$ contains words from either a predefined vocabulary V or from the input text p. Our objective is to generate different question-answer pairs from it. Therefore, we aim to model the probability $P(q, a|p)$.

In our work, we factorize the generation process into multiple steps to select different inputs for question generation. Specifically, given a passage p, we will select three types of information as input to a generative model, which are defined as follows:

- **Answer**: here we define an answer a as a span in the input passage p. Specifically, we select a semantic chunk of p as a target answer from a set of chunks given by parsing and chunking.

- **Clue**: denote a clue as c. As mentioned in Sec. 9.1, a clue is a semantic chunk in input p which will be copied or rephrased in the target question. It is related to the answer a, and providing it as input can help reducing the uncertainty when generating questions. This helps to alleviate the one-to-many mapping problem of question generation, makes the generation process more controllable, as well as improves the quality of generated questions.

- **Style**: denote a question style as s. We classify each question into nine styles: *"who"*, *"where"*, *"when"*, *"why"*, *"which"*, *"what"*, *"how"*, *"yes-no"*, and *"other"*. By providing the target style to the question generation model, we further reduce the uncertainty of generation and increase the controllability.

We shall note that our definition of clue is different with [Liu *et al.*, 2019*b*]. In our work, given a passage p and a question q, we identify a clue as a consistent

chunk in p instead of being the overlapping non-stop words between p and q. On one hand, this allows the clue to be expressed in different ways in a question. On the other hand, given unlabeled text corpus, we can sample clue chunks for generating questions according to the same distribution in training datasets to avoid discrepancy between training and generating.

Given the above definitions, our generation process is decomposed into input sampling and ACS-aware question generation:

$$P(q, a|p) = \sum_{c,s} P(a, c, s|p)P(q|a, c, s, p) \tag{9.1}$$

$$= \sum_{c,s} P(a|p)P(s|a, p)P(c|s, a, p)P(q|c, s, a, p), \tag{9.2}$$

where $P(a|p)$, $P(s|a, p)$, and $P(c|s, a, p)$ model the process of input sampling to get answer, question style, and clue information for a target question; and $P(q|c, s, a, p)$ models the process of generating the target question.

9.3 Model Description

In this section, we present our overall system architecture for generating questions from unlabeled text corpus. We then introduce the details of each component.

Figure 9.2 shows the pipelined system we build to train ACS-aware question generation models and generate large-scale datasets which can be utilized for different applications. Our system consists of four major components: i) dataset constructor, which takes existing QA datasets as input, and constructs training datasets for ACS-aware question generation; ii) information sampler (extractor), which samples answer, clue and style information from input text and feed them into ACS-aware QG models; iii) ACS-aware question generation models, which are trained on the constructed datasets to generate questions; and iv) data filter, which controls the quality of generated questions.

9.3.1 Obtaining Training Data for Question Generation

Our first step is to acquire a training dataset to train ACS-aware question generation models. Existing answer-aware question generation methods [Serban *et al.*, 2016; Du *et al.*, 2017; Liu *et al.*, 2019*b*; Zhou *et al.*, 2017] utilize reading comprehension datasets such as SQuAD [Rajpurkar *et al.*, 2016], as these datasets contain $< p, q, a >$ tuples. However, for our problem, the input and output consists of $< p, q, a, c, s >$, where

ALGORITHM 8: Clue Extraction

Input: passage p, answer a, question q, related words dictionary R.

Output: clue c.

get candidate chunks $C = \{c_1, c_2, \cdots, c_{|C|}\}$ of passage p by parsing and chunking;
remove function words, tokenize p and q to get $p_{t,c}$ and $q_{t,c}$ and stemming p and q to get $p_{m,c}$ and $q_{m,c}$;
for $c \in C$ **do**
    get tokenized clue $c_{t,c}$ and stemmed clue $c_{m,c}$ with only content words;
    $n^o_{t,c} \leftarrow$ number of overlapping tokens between $c_{t,c}$ and $q_{t,c}$;
    $n^o_{m,c} \leftarrow$ number of overlapping stems between $c_{m,c}$ and $q_{m,c}$;
    $n^{soft-o}_{t,c} \leftarrow$ number of soft copied tokens between $c_{t,c}$ and $q_{t,c}$;
    binary $x \leftarrow$ whether q contains the chunk text c;
    $score(c) = n^o_{t,c} + n^o_{m,c} + n^{soft-o}_{t,c} + x$
end for
select the chunk c with maximum $score(c)$ as the clue chunk;

the clue c and style s information are not directly provided in existing datasets. To address this issue, we design effective strategies to automatically extract clue and style information without involving human labeling.

Rules for Clue Identification. As mentioned in Sec. 9.2, given $< p, q, a >$, we define a semantic chunk c in input p which is copied or rephrased in the output question q as clue. We identify c by the method shown in Algorithm 8.

First, we parse and chunk the input passage to get all candidate chunks. Second, we get the tokenized and stemmed passage and question, and only keep the content words in the results. Third, we calculate the similarities between each candidate chunk and the target question according to different criteria. The final score of each chunk is the sum of different similarities. Finally, we select the chunk with the maximum score as the identified clue chunk c.

To estimate the similarities between each candidate chunk and the question, we calculate the number of overlapping tokens and stems between each chunk and the question, as well as checking whether the chunk is fully contained in the question. In addition, we further define "soft copy" relationship between two words to take rephrasing into consideration. Specifically, a word $w_q \in q$ is considered as *soft-copied* from input passage p if there exist a word $w_p \in p$ which is semantically coherent with w_q. To give an instance, consider a passage "Selina left her hometown at the age of 18" and a question "How old was Selina when she left?", the word "old" is soft-copied from "age" in the input passage.

To identify the soft-copy relationship between any pair of words, we utilize synonyms and word vectors, such as Glove [Pennington *et al.*, 2014], to construct a related

ALGORITHM 9: Style Classification

Input: question q, style set $S = \{who, where, when, why, which, what, how, yes\text{-}no, other\}$, yes-no feature words set $Y = \{am, is, was, were, are, does, do, did, have, had, has, could, can, shall, should, will, would, may, might\}$.

Output: style $s \in S$.

```
for s ∈ S\{yes-no, other} do
    if word s is contained in q then
        return s
    end if
end for
for y ∈ Y do
    if word y is the first word of q then
        return yes-no
    end if
end for
return other
```

words dictionary R, where $R(w) = \{w_1, w_2, \cdots, w_{|R(w)|}\}$ returns a set of words that is closely related to w. For each word w, $R(w)$ is composed of the synonyms of w, as well as the top N most similar words estimated by word vector representations (we set $N = 5$). In our work, we utilize Glove word vectors, and construct R based on Genism [Řehůřek and Sojka, 2010] and WordNet [Miller, 1995].

Rules for Style Classification. Algorithm 9 presents our method for question style classification. We classify a given question into 9 classes based on a few heuristic strategies. If q contains *who, where, when, why, which, what*, or *how*, we classify it as the corresponding type. For *yes-no* type questions, we define a set of feature words. If q starts with any word belonging to the set of feature words, we classify it as type *yes-no*. For all other cases, we label it as *other*.

9.3.2 ACS-Aware Question Generation

After obtained training datasets, we design two models for ACS-aware question generation. The first model is based on Seq2Seq framework with attention and copy mechanism [Sutskever *et al.*, 2014; Bahdanau *et al.*, 2014; Gu *et al.*, 2016]. In addition, we exploit clue embedding, content embedding, style encoding and aggressive copying to improve the performance of question generation. The second model is based on pre-trained language models. We fine-tune a GPT2-small model [Radford *et al.*, 2019] using the constructed training datasets.

Seq2Seq-based ACS-aware question generation

Given a passage, an answer span, a clue span, and a desired question style, we train a neural encoder-decoder model to generate appropriate questions.

Encoder. We utilize a bidirectional Gated Recurrent Unit (BiGRU) [Chung *et al.*, 2014] as our encoder. For each word p_i in input passage p, we concatenate the different features to form a concatenated embedding vector w_i as the input to the encoder. Specifically, for each word, it is represented by the concatenation of its word vector, embeddings of its Named Entity Recognition (NER) tag, Part-of-Speech (POS) tag, and whether it is a content word. In addition, we can know whether each word is within the span of answer a or clue c, and utilize binary features to indicate the positions of answer and clue in input passage. All tag features and binary features are casted into 16-dimensional vectors by different embedding matrices that are trainable.

Suppose the embedding of passage p is $(w_1, w_2, \cdots, w_{|p|})$. Our encoder will read the input sequence and produce a sequence of hidden states $h_1, h_2, \cdots, h_{|p|}$, where each hidden state is a concatenation of a forward representation and a backward representation:

$$h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i], \tag{9.3}$$

$$\overrightarrow{h}_i = \mathbf{BiGRU}(w_i, \overrightarrow{h}_{i-1}), \tag{9.4}$$

$$\overleftarrow{h}_i = \mathbf{BiGRU}(w_i, \overleftarrow{h}_{i+1}). \tag{9.5}$$

The $\overrightarrow{h}_i$ and $\overleftarrow{h}_i$ are the forward and backward hidden states of the i-th token in p, respectively.

Decoder. Our decoder is another GRU with attention and copy mechanism. Denote the embedding vector of desired question style s as h_s. We initialize the hidden state of our decoder GRU by concatenating h_s with the last backward encoder hidden state $\overleftarrow{h}_1$ to a linear layer:

$$s_l = tanh(\mathbf{W}_0 \overleftarrow{h}_1 + b), \tag{9.6}$$

$$s_0 = [h_s; s_l]. \tag{9.7}$$

At each decoding time step t, the decoder calculates its current hidden state based on the word vector of the previous predicted word w_{t-1}, previous attentional context vector c_{t-1}, and its previous hidden state s_{t-1}:

$$s_t = \mathbf{GRU}([w_{t-1}; c_{t-1}], s_{t-1}), \tag{9.8}$$

where the context vector c_t at time step t is a weighted sum of input hidden states, and the weights are calculated by the concatenated attention mechanism [Luong *et*

al., 2015]:

$$e_{t,i} = v^{\intercal} tanh(\mathbf{W}_s s_t + \mathbf{W}_h h_i), \tag{9.9}$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j=1}^{|p|} \exp(e_{t,j})}, \tag{9.10}$$

$$c_t = \sum_{i=1}^{|p|} \alpha_{t,i} h_i. \tag{9.11}$$

To generate an output word, we combine w_{t-1}, s_t and c_t to calculate a readout state r_t by an MLP maxout layer with dropouts [Goodfellow *et al.*, 2013], and pass it to a linear layer and a softmax layer to predict the probabilities of the next word over a vocabulary:

$$r_t = \mathbf{W}_{rw} w_{t-1} + \mathbf{W}_{rc} c_t + \mathbf{W}_{rs} s_t \tag{9.12}$$

$$m_t = [\max\{r_{t,2j-1}, r_{t,2j}\}]^{\intercal}_{j=1,\ldots,d} \tag{9.13}$$

$$p(y_t|y1, \cdots, y_{t-1}) = \text{softmax}(\mathbf{W}_o m_t), \tag{9.14}$$

where r_t is a 2-D vector.

For copy or point mechanism [Gulcehre *et al.*, 2016], the probability to copy a word from input p at time step t is given by:

$$g_c = \sigma(\mathbf{W}_{cs} s_t + \mathbf{W}_{cc} c_t + b), \tag{9.15}$$

where σ is the Sigmoid function, and g_c is the probability of performing copying. The copy probability of each input word is given by the attention weights in Equation (10).

It has been reported that the generated words in a target question are usually from frequent words, while the majority of low-frequency words in the long tail are copied from the input instead of generated [Liu *et al.*, 2019*b*]. Therefore, we reduce the vocabulary size to be the top N_V high-frequency words at both the encoder and the decoder, where N_V is a predefined threshold that varies among different datasets. This helps to encourage the model to learn aggressive copying and improves the performance of question generation.

GPT2-based ACS-aware question generation

Pre-trained large-scale language models, such as BERT [Devlin *et al.*, 2018], GPT2 [Radford *et al.*, 2019] and XLNet [Yang *et al.*, 2019], have significantly boosted the performance of a series of NLP tasks including text generation. They are mostly

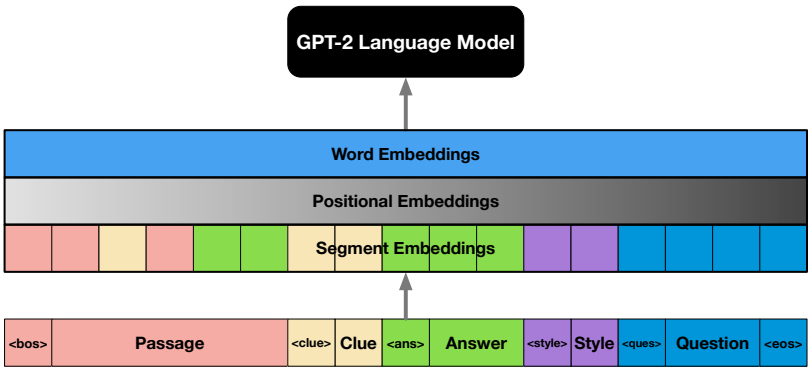


Figure 9.3: The input representations we utilized for fine-tuning GPT-2 Transformer-based language model.

based on the Transformer architecture [Vaswani *et al.*, 2017] and have been shown to capture many facets of language relevant for downstream tasks [Clark *et al.*, 2019]. Compared with Seq2Seq models which often generate text containing repeated words, the pre-trained language models acquire knowledge from large-scale training dataset and are able to generate text of high-quality. In our work, to produce questions with better quality and to compare with Seq2Seq-based models, we further fine-tune the publicly-available pre-trained GPT2-small model [Radford *et al.*, 2019] according to our problem settings.

Specifically, to obtain an ACS-question generation model with GPT2, we concatenate passage, answer, clue and question style as the context of language modeling. Specifically, the input sequence is organized in the form of "*<bos> ..passage text.. <clue> .. clue chunk .. <ans> .. answer chunk .. <style> .. question style .. <ques> .. question text .. <eos>*". During the training process, we learn a language model with above input format. When generating, we sample different output questions by starting with an input in the form of "*<bos> ..passage text.. <clue> .. clue chunk .. <ans> .. answer chunk .. <style> .. question style .. <ques>*". Figure 9.3 illustrates the input representation for fine-tuning GPT-2 language model. Similar to [Krishna and Iyyer, 2019], we leverage GPT-2's segment embeddings to denote the specificity of the passage, clue, answer, style and question. We also utilize answer segment embeddings and clue segment embedding in place of passage segment embeddings at the location of the answer or clue in the passage to denote the position of the answer span and clue span. During the generation process, the trained model uses top-p nucleus sampling with $p = 0.9$ [Holtzman *et al.*, 2019] instead of beam search and top-k sampling. For implementation, we utilize the code base of [Krishna and Iyyer, 2019] as a starting point, as well as the Transformers library from HuggingFace [Wolf *et al.*, 2019].

9.3.3 Sampling Inputs for Question Generation

As mentioned in Sec. 9.2, the process of ACS-aware question generation consists of input sampling and text generation. Given an unlabeled text corpus, we need to extract valid <*passage, answer, clue, style*> combinations as inputs to generate questions with an ACS-aware question generation model.

In our work, we decompose the sampling process into three steps to sequentially sample the candidate answer, style and clue based on a given passage. We make the following assumptions: i) the probability of a chunk a be selected as an answer only depends on its Part-of-Speech (POS) tag, Named Entity Recognition (NER) tag and the length (number of words) of the chunk; ii) the style s of the target question only depends on the POS tag and NER tag of the selected answer a; and iii) the probability of selecting a chunk c as clue depends on the POS tag and the NER tag of c, as well as the dependency distance between chunk c and a. We calculate the length of the shortest path between the first word of c and that of a as the dependency distance. The intuition for the last designation is that a clue chunk is usually closely related to the answer, and is often copied or rephrased into the target question. Therefore, the dependency distance between a clue and an answer will not be large [Liu *et al.*, 2019*b*].

With above assumptions, we will have:

$$P(a|p) = P(a|POS(a), NER(a), length(a)), \tag{9.16}$$

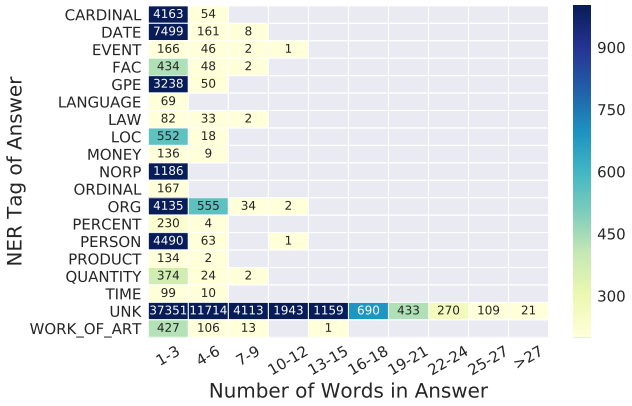
$$P(s|a, p) = P(s|POS(a), NER(a)), \tag{9.17}$$

$$P(c|s, a, p) = P(c|POS(c), NER(c), DepDist(c, a)), \tag{9.18}$$

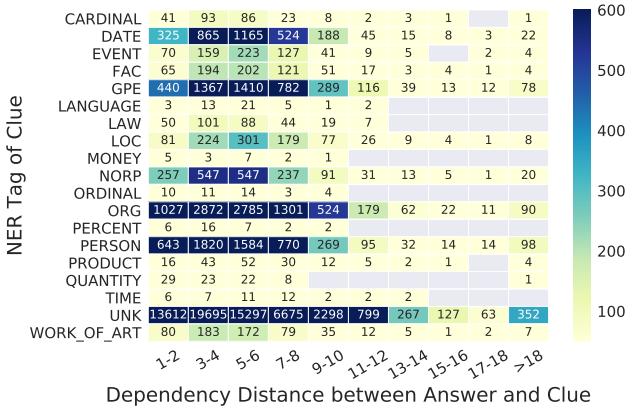
where $DepDist(c, a)$ represents the dependency distance between the first token of c and that of a.

The above conditional probabilistic distributions can be learned from an existing dataset (such as SQuAD), named as reference dataset. Given a reference dataset consisting of <*passage, question, answer*> triplets, first, we perform POS tagging, NER tagging, parsing and chunking on the input passages. Second, we recognize the clue and style information according to the steps described in Sec. 9.3.1, get the NER tag and the POS tag of both the answer chunk and the clue chunk, and calculate the dependency distance between the clue chunk and the answer chunk. Finally, we calculate the conditional distributions according to the extracted information.

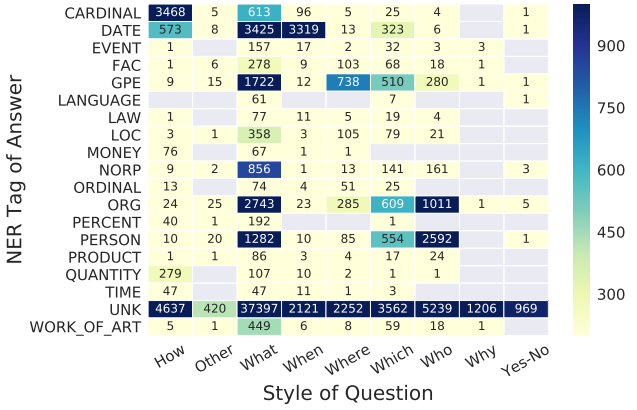
We set the maximum length of an candidate answer to 30, and split the range of length into 10 bins of equal size to calculate $P(a|POS(a), NER(a), length(a))$. Similarly, we set the maximum dependency distance between a clue chunk and an



(a) Joint Distribution of Answer NER and Answer Length



(b) Joint Distribution of Clue NER and Dependency Distance



(c) Joint Distribution of Answer NER and Question Style

Figure 9.4: The input join distributions we get using SQuAD1.1 training dataset as reference data.

answer chunk to be 20, and split the range of distance into 10 bins of equal size to calculate $P(c|POS(c), NER(c), DepDist(c, a))$. Figure 9.4 shows the marginal distributions we get by utilizing the SQuAD1.1 training dataset as our reference data. The NER tagging [1] is performed by spaCy [Honnibal and Montani, 2017], and the "UNK" tag means the chunk is not recognized as a named entity. From Figure 9.4(a), we can see that most of the answers are short, and a majority of them are entities such as person (PERSON), organization (ORG) or date (DATE). From Figure 9.4(b), we can see the syntactical dependency distance between a clue chunk and an answer is usually less than 8, which matches with our intuition that a clue shall be correlated with the answer so that it will be copied or rephrased in the target question. Finally, Figure 9.4(c) shows most of the questions are "What" style, and the followings are "Who", "How" and "When". The NER tags of answers are highly correlated with the style of questions. Also, we shall notice that the NER performance of spaCy is not perfect. Therefore, we may observe weird cases such as organization (ORG) matches with "Who". Determining different conditional probabilities with a reference dataset instead of following pre-defined rules helps us to take such kind of noises into account.

After calculating the above distributions according to an reference dataset, we can sample different information given a passage p. First, we get all candidate chunks $K = \{k_1, k_2, \cdots, k_{|K|}\}$ by parsing and chunking over p. Second, we sample a chunk k_i as answer according to the normalized probability distribution over all chunks:

$$P(k_i) = \frac{P(k_i|POS(k_i), NER(k_i), length(k_i))}{\sum_{j=1}^{|K|} P(k_j|POS(k_j), NER(k_j), length(k_j))}. \tag{9.19}$$

Third, we sample a question style s_i over all possible questions styles $S = \{s_1, s_2, \cdots, s_{|S|}\}$ by the normalized probability:

$$P(s_i) = \frac{P(s_i|POS(k_i), NER(k_i))}{\sum_{j=1}^{|S|} P(s_j|POS(k_i), NER(k_i))}. \tag{9.20}$$

Finally, we sample a chunk k_l as clue according to:

$$P(k_l) = \frac{P(k_l|POS(k_l), NER(k_l), DepDist(k_l, k_i))}{\sum_{j=1}^{|K|} P(k_j|POS(k_j), NER(k_j), DepDist(k_j, k_i))}. \tag{9.21}$$

We can repeat the above steps for multiple times to get different inputs from the same passage and generate diverse questions. In our work, for each input passage, we sample 5 different chunks as answer spans, 2 different question styles for each answer, and 2 different clues for each answer. In this way, we over-generate questions by sampling 20 questions for each sentence.

[1] The meaning of NER labels can be found at: https://spacy.io/api/annotation

9.3.4 Data Filtering for Quality Control

After sampled multiple inputs from each sentence, we can generate different questions based on the inputs. However, it is hard to ask each sentence 20 meaningful and different questions even given 20 different inputs derived from it, as the questions may be duplicated due to similar inputs, or the questions can be meaningless if the *<answer, clue, style>* combination is not reasonable. Therefore, we further utilize a filter to remove low-quality QA pairs.

We leverage an entailment model and a QA model based on BERT [Devlin *et al.*, 2018]. For the entailment model, as the SQuAD 2.0 dataset [Rajpurkar *et al.*, 2018] contains unanswerable questions, we utilize it to train a classifier which tells us whether a pair of *<question, answer>* matches with the content in the input passage. For the question answering model, we fine-tuned another BERT-based QA model utilizing the SQuAD 1.1 dataset [Rajpurkar *et al.*, 2016].

Given a sample *<passage, question, answer>*, we keep it if this sample satisfies two criteria: first, it is classified as positive according to the BERT-based entailment model; second, the F1 similarity score between the gold answer span and the answer span predicted by BERT-based QA is above 0.9. Note that we do not choose to fine-tune a BERT-based QA model over SQuAD 2.0 to perform entailment and question answering at the same time. That is because we get better performance by separating the entailment step with the QA filtering step. Besides, we can utilize extra datasets from other entailment tasks to enhance the entailment model and further improve the data filter.

9.4 Evaluation

In this section, we compare our proposed ACS-aware question generation with answer-aware question generation models to show the benefits. Besides, we generate large number of QA pairs from unlabeled text corpus using our models, and perform a variety of evaluations to analyze the quality of the generated dataset. Furthermore, we test the performance of QA models trained on our generated dataset, and show the potential applications and future directions of our work.

9.4.1 Evaluate ACS-aware Question Generation

Datasets, Metrics and Baselines. We evaluate the performance of ACS-aware question generation based on the SQuAD dataset [Rajpurkar *et al.*, 2016]. It is a reading comprehension dataset which contains questions derived from Wikipedia

articles, and the answer to every question is a segment of text from the corresponding reading passage. In our work, we use the data split proposed by [Zhou *et al.*, 2017], where the input is the sentence that contains the answer. The training set contains 86,635 samples, and the original dev set that contains 17,929 samples is randomly split into a dev test and a test set of equal size. The average lengths (number of words) of sentences, questions and answers are 32.72, 11.31, and 3.19, respectively.

The performance of question generation is evaluated by the following metrics.

- **BLEU** [Papineni *et al.*, 2002]. BLEU measures precision by how much the words in predictions appear in reference sentences. BLEU-1 (B1), BLEU-2 (B2), BLEU-3 (B3), and BLEU-4 (B4), use 1-gram to 4-gram for calculation, respectively.

- **ROUGE-L** [Lin, 2004]. ROUGE-L measures recall by how much the words in reference sentences appear in predictions using Longest Common Subsequence (LCS) based statistics.

- **METEOR** [Denkowski and Lavie, 2014]. METEOR is based on the harmonic mean of unigram precision and recall, with recall weighted higher than precision.

We compare our methods with the following baselines.

- **PCFG-Trans** [Heilman, 2011]: a rule-based answer-aware question generation system.

- **SeqCopyNet** [Zhou *et al.*, 2018], **NQG++** [Zhou *et al.*, 2017], **AFPA** [Sun *et al.*, 2018], **seq2seq+z+c+GAN** [Yao *et al.*, 2018], and **s2sa-at-mp-gsa** [Zhao *et al.*, 2018]: answer-aware neural question generation models based on Seq2Seq framework.

- **NQG-Knowledge** [Gupta *et al.*, 2019], **DLPH** [Gao *et al.*, 2018]: auxiliary-information-enhanced question generation models with extra inputs such as knowledge or difficulty.

- **Self-training-EE** [Sachan and Xing, 2018], **BERT-QG-QAP** [Zhang and Bansal, 2019], **NQG-LM** [Zhou *et al.*, 2019*a*], **CGC-QG** [Liu *et al.*, 2019*b*] and **QType-Predict** [Zhou *et al.*, 2019*b*]: multi-task question generation models with auxiliary tasks such as question answering, language modeling, question type prediction and so on.

The reported performance of baselines are directly copied from their papers or evaluated by their published code on GitHub.

For our models, we evaluate the following versions:

- **CS2S-VR-A**. Content separated Seq2Seq model with Vocabulary Reduction for Answer-aware question generation. In this variant, we incorporate content embeddings in word representations to indicate whether each word is a content word or a function word. Besides, we reduce the size of vocabulary by only keeping the top 2000 frequent words for encoder and decoder. In this way, low-frequency words are represented by its NER, POS embeddings and feature embeddings. We also add answer position embedding to indicate the answer span in input passages.

- **CS2S-AS**. This model adds question style embedding to initialize decoder, without vocabulary reduction (vocabulary size is $20,000$ when we do not exploit vocabulary reduction).

- **CS2S-AC**. The variant adds clue embedding in encoder to indicate the span of clue chunk.

- **CS2S-ACS**. This variant adds both clue embedding in encoder and style embedding in decoder.

- **CS2S-VR-ACS**. This is the fully featured model with answer, clue and style embedding, as well as vocabulary reduction.

- **GPT2-ACS**. This is our fine-tuned GPT2-small model for ACS-aware question generation.

Experiment Settings. We implement our models in PyTorch 1.1.0 [Paszke *et al.*, 2017] and Transformers 2.0.0 [Wolf *et al.*, 2019], and train the model with two Tesla P40. We utilize spaCy [Matthew Honnibal, 2015] to perform dependency parsing and extract lexical features for tokens. For Seq2Seq-based models, we set word embeddings to be 300-dimensional and initialize them by GloVe, and set them trainable. The out-of-vocabulary words are initialized randomly. All other features are embedded to 16-dimensional vectors. The encoder is a single layer BiGRU with hidden size 512, and the decoder is a single layer undirected GRU with hidden size 512. We set dropout rate $p = 0.1$ for the encoder, decoder, and the attention module. We train the models by Cross-Entropy loss for question generation and question copying, and perform gradient descent by the Adam [Kingma and Ba, 2014] optimizer with

Model	B1	B2	B3	B4	ROUGE-L	METEOR
PCFG-Trans	28.77	17.81	12.64	9.47	31.68	18.97
SeqCopyNet	–	–	–	13.02	44.00	–
seq2seq+z+c+GAN	44.42	26.03	17.60	13.36	40.42	17.70
NQG++	42.36	26.33	18.46	13.51	41.60	18.18
AFPA	43.02	28.14	20.51	15.64	–	–
s2sa-at-mp-gsa	44.51	29.07	21.06	15.82	44.24	19.67
NQG-Knowledge	–	–	–	13.69	42.13	18.50
DLPH	44.11	29.64	21.89	16.68	46.22	20.94
NQG-LM	42.80	28.43	21.08	16.23	–	–
QType-Predict	43.11	29.13	21.39	16.31	–	–
Self-training-EE	–	–	–	14.28	42.97	18.79
CGC-QG	46.58	30.90	22.82	17.55	44.53	21.24
BERT-QG-QAP	–	–	–	18.65	46.76	22.91
CS2S-VR-A	45.28	29.58	21.45	16.13	43.98	20.59
CS2S-AS	45.79	29.12	20.59	15.09	45.84	20.14
CS2S-AC	48.13	32.51	24.08	18.40	47.45	22.27
CS2S-ACS	50.72	34.60	25.79	19.84	51.08	23.58
CS2S-VR-ACS	**52.30**	**36.70**	**28.00**	**22.05**	**53.25**	25.11
GPT2-ACS	42.60	31.23	24.00	18.87	43.63	**25.15**

Table 9.1: Evaluation results of different models on SQuAD dataset.

an initial learning rate $l_r = 0.001$, two momentum parameters are $\beta_1 = 0.8$ and $\beta_2 = 0.999$ respectively, and $\epsilon = 10^{-8}$. The mini-batch size for each update is set to 32 and model is trained for up to 10 epochs. Gradient clipping with range $[-5, 5]$ is applied to Adam. Beam width is set to be 20 for decoding. The decoding process stops when the $<EOS>$ token (represents end-of-sentence) is generated.

For GPT2-ACS model, we fine-tune the GPT2-small model using SQuAD 1.1 training dataset from [Zhou *et al.*, 2017]. We fine-tune the model for 4 epochs with batch size 2, and apply top-p nucleus sampling with $p = 0.9$ when decoding. For BERT-based filter, we fine-tune the BERT-large-uncased model from HuggingFace [Wolf *et al.*, 2019] with parameters suggested by [Wolf *et al.*, 2019] for training on SQuAD 1.1 and SQuAD 2.0. Our code will be published for research purpose[2].

Main Results. Table. 9.1 compares our models with baseline approaches. We can see that our CS2S-VR-ACS achieves the best performance in terms of the evaluation metrics and outperforms baselines by a great margin. Comparing CS2S-VR-A with Seq2Seq-based answer-aware QG baselines, we can see that it outperforms all the baseline approaches in that category with the same input information (input passage and answer span). This is because that our content separation strategy and

[2]https://github.com/BangLiu/ACS-QG

Experiments		CS2S-VR-ACS	GPT2-ACS	GOLD
Question is Well-formed	No	28.5%	6.0%	2.0%
	Understandable	31.5%	19.5%	9.0%
	Yes	40.0%	74.5%	89.0%
Question is Relevant	No	6.3%	11.7%	7.1%
	Yes	93.7%	88.3%	92.9%
Answer is Correct	No	7.4%	3.6%	2.2%
	Partially	12.7%	15.1%	15.4%
	Yes	79.9%	81.3%	82.4%

Table 9.2: Human evaluation results about the quality of generated QA pairs.

vocabulary reduction operation help the model to better learn what words to copy from the inputs. Comparing our ACS-aware QG models and variants with auxiliary-information-enhanced models (such as NQG-Knowledge and DLPH) and auxiliary-task-enhanced baselines (such as BERT-QG-QAP), we can see that the clue and style information helps to generate better results than models with knowledge or difficulty information. That is because our ACS-aware setting makes the question generation problem closer to one-to-one mapping, and greatly reduces the task difficulty.

Comparing GPT2-ACS with CS2S-VR-ACS, we can see that GPT2-ACS achieves better METEOR score, while CS2S-VR-ACS performs better over BLEU scores and ROUGE-L. That is because GPT2-ACS has no vocabulary reduction. Hence, the generated words are more flexible. However, metrics such as BLEU scores are not able to evaluate the quality of generated QA pairs semantically. Therefore, in the following section, we further analyze the quality of QA pairs generated by CS2S-VR-ACS and GPT2-ACS to identify their strengths and weaknesses.

9.4.2 Qualitative Analysis

After training our CS2S-VR-ACS and GPT2-ACS models, we generate large-scale *<passage, question, answer>* datasets from unlabeled text corpus. Specifically, we obtain the top $10,000$ English Wikipedia articles with Project Nayuki's Wikipedia's internal PageRanks. After that, we split each article in the corpus into sentences, and filter out sentences with lengths shorter than 5 or longer than 100. Based on these sentences, we perform input sampling to sample *<answer, clue, style>* triplets for each sentence according to the steps described in Section 9.3.3, and feed them into our models to generate questions. After filtering, we create two datasets utilizing the two models, where each of the dataset contains around 1.4 million generated questions.

We first evaluate the quality of generated QA pairs by running pilot user study.

We ask 10 graduate students to evaluate 500 <*passage, question, answer*> samples: 200 samples generated by the CS2S-VR-ACS model, 200 samples generated by the GPT2-ACS model, and 100 gold samples from the SQuAD 1.1 training dataset. All the samples are randomly shuffled, and each sample will be evaluated by 3 workers. We acquire judgments of the following questions:

- **Is the question well-formed?** This is to check whether a given question is both grammatical and meaningful [Krishna and Iyyer, 2019]. Workers will select *yes*, *no*, or *understandable*. The option *understandable* is selected if a question is not totally grammatically correct, but we can infer its meaning.
- **If the question is well-formed or understandable, is the question relevant to the passage?** Workers will select *yes* if we can find the answer to the generated question in the passage.
- **If the question is relevant to the passage, is the answer actually a valid answer to the generated question?** Workers will select *yes*, *no* or *partially*. The last option represents that the answer in our generated sample partially overlaps with the true answer in the passage.

Table 9.2 shows the evaluation results based on the sampled data. First, we can see that even the gold samples from the SQuAD dataset are not totally well-formed, relevant and answer correct. That is because we only use the sentence which contains the answer span as context. However, about 20% questions in SQuAD require paragraph-level information to be asked [Du *et al.*, 2017]. Second, 94% of the questions generated by GPT2-ACS are well-formed or understandable, while the percentage is 71.5% for the CS2S-VR-ACS model. We can see that although the BLEU scores of the GPT2-ACS model are lower than that of CS2S-VR-ACS, the results of GPT2-ACS are semantically better due to the knowledge learned from large-scale pre-training. Third, we can see that most of the questions generated by both CS2S-VR-ACS and GPT2-ACS are relevant to the input passages. Finally, most of the answers are also correct given the generated questions. This demonstrates the high-quality of the question-answer pairs generated by our models.

Figure 9.5 is a running example to show the properties of generated questions. We can see that our ACS-aware question generation models are able to generate various questions based on different answers, question styles, and clue chunks. Compared with answer-aware question generation, our generation process is more controllable, and our generated questions are more diverse and of high quality. In some cases, the answer span does not match with the question. We further performed pilot user

Figure 9.5: Showcases of generated questions-answer pairs by our system.

studies to analyze the bad cases in our generated samples. For each question that is not well-formed, we ask workers to label the weakness of it. The results of the study show that most of the errors are grammatically errors, type mismatches, meaningless, or incomplete information. For CS2S-VR-ACS, about 56.7% of the bad cases are grammatically incorrect; 29.2% of them have the problem of type mismatch, e.g., the generated question starts with "When" when asking questions about a person; 14.2% of them are grammatically correct, but meaningless; and 17.5% of the questions do not express their meaning explicitly due to missing words or fuzzy pronouns. Similarly, for GPT2-ACS, the percentages of the above four problems are 40.4% (grammatically incorrect), 46.2% (type mismatches), 15.4% (meaningless) and 11.6% (incomplete or fuzzy information). Note that the sum of these percentages does not equal to 1. That is because each question may be labeled with multiple types of weaknesses.

In order to reduce different errors and further improve the quality of generated questions, first, we need to incorporate the knowledge of natural language by methods such as large-scale pre-training. We can observe from Table 9.2 that most of the questions generated by GPT2-ACS are well-formed and grammatically correct. Second, we can utilize a better named entity recognition model to provide more accurate information about the named entity types of answer chunks. In this way, the type mismatch errors will be reduced. Last but not the least, the problem of semantic mismatching, meaningless, or information incompleteness can be reduced by training a better entailment model to enhance the data filter.

Experiments	CS2S-VR-ACS		GPT2-ACS	
	EM	F1	EM	F1
SQuAD	86.72	92.97	86.72	92.97
Generated	71.14	83.53	74.47	85.64
Generated + SQuAD	86.12	92.36	85.87	92.33

Table 9.3: Evaluate the performance of question answering with different training datasets.

9.4.3 Apply to Question Answering

We also perform quality test of the generated question-answer pairs by applying them to downstream machine reading comprehension tasks. Specifically, we train different BERT-based question answering models based on the following settings:

- **SQuAD**: in this experiment, we utilize the original SQuAD 1.1 training dataset to train a BERT-based QA model, and test the performance on the dev dataset. The performance is evaluated by exact match (EM) and F1-Score (F1) between predicted answer span and the true answer span [Rajpurkar *et al.*, 2016].

- **Generated**: in this experiment, we sample a training dataset from our generated questions, where the size is equal to the training dataset of SQuAD 1.1. Although our questions are generated from sentences, we utilize the paragraphs the sentences belong to as contexts when training QA models.

- **Generated + SQuAD**: in this experiment, we combine the original SQuAD training dataset with our generated training dataset to train the BERT-based QA model.

For all the QA experiments, the configurations are the same except the training datasets.

Table 9.3 compares the performance of the resulting BERT-based QA models trained by above settings. Our implementation gives 86.72% EM and 92.97% F1 when trained on the original SQuAD dataset. In comparison, the model trained by our generated dataset gives 74.47% and 85.64% F1. When we combine the generated dataset with the SQuAD training set to train the QA model, the performance is not further improved. The results are reasonable. First, the generated dataset contains noises which will influence the performance. Second, simply increasing the size of training dataset will not always help with improving the performance. If most of the generated training samples are already answerable by the model trained over the original SQuAD dataset, they are not very helpful to further enhance the generalization ability of the model.

There are at least two methods to leverage the generated dataset to improve the performance of QA models. First, we can utilize curriculum learning algorithms [Bengio *et al.*, 2009] to select samples during training. We can select samples according to the current state of the model and the difficulties of the samples to further boost up the model's performance. Note that this requires us to remove the BERT-based QA model from our data filter, or set the threshold of filtering F1-score to be smaller. Second, similar to [Gao *et al.*, 2018], we can further incorporate the difficulty information into our question generation models, and encourage the model to generate more difficult question-answer pairs. We leave these to our future works.

Aside from machine reading comprehension, our system can be applied to many other applications. First, we can utilize it to generate exercises for educational purposes. Second, we can utilize our system to generate training datasets for a new domain by fine-tuning it with a small amount of labeled data from that domain. This will greatly reduce the human effort when we need to construct a dataset for a new domain. Last but not the least, our pipeline can be adapted to similar tasks such as comment generation, query generation and so on. The key insight is that we will need to decompose the inputs for a new task to help reducing the uncertainty of generation process, as well as propose suitable strategies to extract the required inputs from unlabeled text corpus.

9.5 Conclusion

In this chapter, we propose ACS-aware question generation, a novel text generation task which learns to generate questions based on input passages, clues, answers and question styles. Our proposed task significantly reduces the difficulty of question generation and provides more controllability over the generation process. In addition, we further propose effective strategies to sample meaningful combinations of different inputs from unlabeled text corpus, as well as filtering strategies to control the quality of generated question-answer pairs. Our system automatically generate two large datasets containing 1.33 million and 1.45 million QA pairs from Wikipedia articles. We present and evaluate our system and models to demonstrate the performance. Compared with existing answer-aware question generation models or models with auxiliary inputs or tasks, our ACS-aware QG model achieves significantly better performance, which highlights the importance of clue and style information. We further run pilot user studies to evaluate the quality of generated data. The evaluation results show that our model is able to generate diverse and high-quality questions from the same input sentence. Finally, we point out potential future directions to

further improve the performance of our pipeline.

Chapter 10

Conclusions

10.1 Summary

In this dissertation, we explore the idea of graph modeling for text pieces and propose a variety of approaches for different tasks.

In Chapter 3, different with traditional document clustering or topic detection, we propose to cluster extreme fine-grained hot events from news articles, which matches with the granularity of user interests. Furthermore, we organize correlated events into tree structures story trees to show the logical connections between them. We aim to design and implement a new generation of intelligent news organization system in the era of information explosion. Such a system helps to organize massive documents into structured story trees which are intuitive to human beings. Besides, it helps to reduce information redundancy by clustering documents with homogeneous contents into events. Furthermore, it tracks both short and long term stories by connecting correlated events into story trees and updating them in an online manner. We propose the novel *EventX* clustering algorithm for fine-grained event and story clustering in our *Story Forest* system. We implement our system in Tencent QQ Browser and show the performance of Story Forest through extensive simulations.

A key problem in event clustering is: how to classify the relationship between two documents and judge whether they are essentially talking about the same event or topic. Existing research works are mainly focus on sentence matching, and few research works are about matching two long documents. In Chapter 4 and 5, we study the problem of text matching. Specifically, we study both long document matching and short sentence matching. For long document matching, we propose the Concept Interaction Graph representation to represent one or a pair of documents. Each vertex in the graph is essentially a sub-topic in the document (pairs). By constructing such a graph representation, we divide a long document into short pieces over different

sub-topics. We then utilize Siamese encoding for local matching over vertices, and combine the local matching results through graph convolutional networks to give the final matching result.

Concept Interaction Graph decomposes the problem of long document matching into the problem of short text matching over different vertices. For short text matching, in Chapter 5, we further propose hierarchical sentence factorization technique to factorize a sentence into hierarchical of semantic units. Each layer in our factorized sentence tree contains the full information of the original sentence. With the increase of depth, the semantic units factorize into smaller granularity, starting from the original sentence and until they are factorized into words. We organize the semantic units in different layers by following the general "predict-argument" structure. Finally, such a hierarchical and ordered tree representation helps to align the semantic units in two sentences from different granularity. Based on the idea of factorization and alignment, we propose new unsupervised semantic distance metric between two sentences, as well as extend existing deep neural matching models to multi-granularity matching models.

The graph-based modeling of different text pieces, including sentences, documents and corpora, significantly improves the performance of many NLP tasks. However, for natural language understanding, we need to model the knowledge of the world and perform reasoning with them. Therefore, instead of connecting pure text elements, discovering new entities and concepts, and modeling the relationships between them to construct a model of world is of great value to improve the understanding of natural language. In Chapter 6 and 7, we propose our *ConcepT* and *GIANT* systems for user-centered concept mining and topic/event mining. Different with traditional knowledge graphs, our system discovers user-centered concepts as well as hot topics/events from large scale user queries and search click graphs. In this way, the discovered phrases are conforming to the language style of real world users, instead of expressed in "writer-perspective". Our system discovers concepts and construct a large-scale user interests ontology to depict the hierarchical relationships between different user interests. It is of great value to both user modeling and text understanding.

A structured concept base characterizes the world in a simplified form. To further improve the performance of language understanding tasks, such as machine reading comprehension, we still need a large amount of training data. However, such kind of training datasets are expensive and requires a lot of time to construct. Besides, it is also hard to guarantee the quality of constructed datasets. In Chapter 8 and 9, we look into the problem of automatic question generation which aims to generate context-question-answer triplets for various applications. We notice that given the

context sentence and even a specific answer, the problem of question generation is still a one-to-many mapping problem. Therefore, it is hard to generate questions exactly like the given ones in the training dataset. In our proposed approach, we emphasize on the concept of "clue words" in question generation. Specifically, clue words indicate what aspect a question is asking about. Given an answer and the related clue words, we can narrow the way how we can ask questions about a sentence. As the clue words are not given in any existing question answering and generation datasets, we propose to predict the potential clue words by constructing a clue predictor which combines the syntactical structure of sentences with graph convolutional networks. We further proposed a strategy to encourage aggressive copying from input sentences when generating questions. Our final model significantly improves the performance of question generation when compared with exiting approaches.

In Chapter 9, we introduce the "answer-clue-style" aware question generation task, and further propose efficient strategies to sample different input elements for this task. Our new paradigm of question generation dramatically reduces the difficulty of generating high-quality questions by providing more input information to the generation system. Combing our new task with large pretrained language models such as GPT-2, we can generate a large-scale question-answer pairs dataset for machine reading comprehension with a relative small size of training data.

10.2 Directions for Future Work

10.2.1 Extending Current Research Work

Our work on information organization, information recommendation and reading comprehension can be further extended in the following directions.

Story Forest 2.0: User-Centered and Real-Time Story Construction. We will extend our proposed Story Forest and EventX framework to make it applicable to a continuous stream of news documents, and combine it with the user attention ontology created by our GIANT system. On one hand, currently, we extract the events for each batch of documents collected in different hours, and merge a new event with an existing event if they are talking about the same one. We can further improve Story Forest by maintaining a dynamic keyword graph, performing dynamic keyword community detection, and run incremental event extraction based on such a dynamic keyword graph. Whenever new document stream comes in, we update the keyword graph to find out new keyword communities, and cluster new documents into new events or assign them to existing events. In this way, our framework can be directly

applied to a continuous stream of news documents, rather than processing them in batches. On the other hand, the attention ontology created by the GIANT system contains the relationships between different events and topics. For each event, it also shows its arguments such as involved entities, location and trigger words. Based on these information, we can improve the story structure generation algorithm by taking the event arguments into account. We can also generate user-centered stories by taking each user's interest into account. In this way, the generated stories will mainly contain important and user interested events. Furthermore, we can guide users to learn and grow with the constructed story trees. That is to say, with the growth of story trees, users will learn more useful and advanced topics based on what he/she has learned previously.

Long Document Understanding with Attention Graph. Existing works on machine reading comprehension do not perform well for long documents. Compared with the task of SQuAD question answering, we need to retrieve relevant documents which may contain the answer to a specific question from a large corpus, and extract or generate an answer from selected documents. The attention graph constructed by our GIANT system is able to characterize the topic coverages of a document. Based on it, we can better retrieve relevant documents given a specific user query. We can also improve the performance of question answering by utilizing the attention tags associated to a document as prior knowledge to depicting and organizing the content of a document.

Joint Learning of Question Answering and Question Generation. Our ACS-aware question generation can help to generate large-scale high quality question-answer pairs from unlabeled text corpus, which can serve as the training dataset for machine reading comprehension. However, increasing the size of training dataset may not help improve performance if a large amount of high-quality data is available. To further improve machine reading comprehension tasks such as SQuAD QA, one may need to perform bad-case-directed curriculum/active learning to select appropriate training samples given the QA model. Furthermore, we can jointly learn the models of QA and QG by sharing the states between the two models. In this way, the training state of the QA model can guide the QG model to generate QA pairs it needs. The QG model can potentially further improve the performance of the QA model by generating QA pairs that the QA model cannot answer correctly at present.

10.2.2 Long-Term Research Goals

My long-term research aims to develop theories and algorithms for machine intelligence, with applications to various domains. Deep learning models are good at learning representations of inputs and mappings from inputs to outputs. However, they are data-intensive and compute-intensive, as well as lack the ability of explainable reasoning and inference. Graph neural networks (GNNs) are capable of incorporating a sparse and discrete dependency structure of input data. However, they lack the ability of understanding or automatically establishing the implicit relationships between different things. To fuse the advantage of deep learning and relational reasoning to learn from heterogeneous data, I am planning to focus on *relational and discrete representation learning and reasoning with graphs*.

Our current work propose a unified methodology to transform different text pieces into suitable graph representations, and combine such a representation with deep learning models to improve the performance of various NLP and text mining tasks. However, first, the designation of graph representations are specific to each task and requires a lot human experience. How to automatically design or learn appropriate graph structures from raw input text remains a challenging problem. Second, existing works are mostly learning the hidden representation of graph nodes by a multi-layer graph neural network. To the best of our knowledge, there is few research work on how to learn different and higher-level nodes and edges with multi-layer graph neural network. From our point of view, such a multi-layered graph transformation can serve as a model of logical reasoning and inference. Last but not least, our work has not take take the relationships between different tasks into account.

Based on above thoughts, in the future, we are going to explore the following potential research directions:

Representation and Relation Learning with Graph Neural Networks. Existing GNNs can only learn from input when a graph-structure of input data is available. However, real-world graphs are often noisy and incomplete or might not be available at all. Our study aims to design effective models and algorithms to automatically learn the relational structure in input data with limited structured inductive biases. Instead of manually designing specific graph representations of data for different applications, we will enable models to automatically identify the implicit relationships between input data points, and learn the graph structure and representations of inputs.

Joint Learning of Graph Structure and Graph Embedding. Graph/Network embedding [Cui *et al.*, 2018] assigns nodes in a network to low-dimensional represen-

tations and effectively preserves the network structure. Our research work is mainly focusing on designing the graph representations of different data, and solving different tasks with manually designed graph representation and node/edge feature vectors. To reduce the human effort in designation, we can jointly learn the structure of input data and the embedding of graph nodes/edges.

Hierarchical Relational Learning for Reasoning and Inference. Learning and reasoning with graph-structured representations naturally supports interpretability, causality, and transferability / inductive generalization. Existing deep neural models are mostly learning over fixed relational topologies of input data. In this study, we are aiming to perform hierarchical representation and relation learning over both the data points (nodes) and the relationships (edges). It automatically groups and aggregates related data points into semantic units of diverse granularities, as well as learns to capture the implicit and high-level semantic relationships between data points in a hierarchical manner. A potential way to achieve this is combining representation learning with symbolic inference over graph-structured data to enable a model to perform explainable reasoning and inference.

Task Relation Modeling for Continual Lifelong Meta-Learning. In this study, we propose to marriage the universal representation power of graph structures with multi-task learning to integrate diverse input data, such as images, text pieces, and knowledge bases, and jointly learn a unified and structured representation for various tasks. Furthermore, we learn the relationships or correlations between different tasks, and exploit the learned relationship for curriculum learning to accelerate the rate of convergence in different tasks. Finally, with the universal representation and integration of different data, as well as the joint and curriculum learning of different tasks, our artificial learning systems will gain the ability to continually acquire, fine-tune, and transfer knowledge and skills throughout their lifespan.

Bibliography

Adomavicius, Gediminas and Alexander Tuzhilin (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering* (6), 734–749.

Adomavicius, Gediminas and Alexander Tuzhilin (2011). Context-aware recommender systems. In: *Recommender systems handbook*. pp. 217–253. Springer.

Aggarwal, Charu C and ChengXiang Zhai (2012). A survey of text clustering algorithms. In: *Mining text data*. pp. 77–128. Springer.

Albregtsen, Fritz et al. (2008). Statistical texture measures computed from gray level coocurrence matrices. *Image processing laboratory, department of informatics, university of oslo*.

Allan, James (2012). *Topic detection and tracking: event-based information organization*. Vol. 12. Springer Science & Business Media.

Allan, James, Ron Papka and Victor Lavrenko (1998). On-line new event detection and tracking. In: *SIGIR*. ACM. pp. 37–45.

Aone, Chinatsu and Mila Ramos-Santacruz (2000). Rees: a large-scale relation and event extraction system. In: *Proceedings of the sixth conference on Applied natural language processing*. Association for Computational Linguistics. pp. 76–83.

Atefeh, Farzindar and Wael Khreich (2015). A survey of techniques for event detection in twitter. *Computational Intelligence* **31**(1), 132–164.

Auer, Sören, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak and Zachary Ives (2007). Dbpedia: A nucleus for a web of open data. *The semantic web* pp. 722–735.

Bahdanau, Dzmitry, Kyunghyun Cho and Yoshua Bengio (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Baker, Collin and Michael Ellsworth (2017). Graph methods for multilingual framenets. In: *Proceedings of TextGraphs-11: the Workshop on Graph-based Methods for Natural Language Processing*. pp. 45–50.

Baker, Collin F, Charles J Fillmore and John B Lowe (1998). The berkeley framenet project. In: *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics. pp. 86–90.

Balinsky, Helen, Alexander Balinsky and Steven Simske (2011). Document sentences as a small world. In: *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*. IEEE. pp. 2583–2588.

Banarescu, Laura, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer and Nathan Schneider (2012). Abstract meaning representation (amr) 1.0 specification. In: *Parsing on Freebase from Question-Answer Pairs. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle: ACL*. pp. 1533–1544.

Banarescu, Laura, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer and Nathan Schneider (2013). Abstract meaning representation for sembanking. In: *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. pp. 178–186.

Battaglia, Peter W, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Baudiš, Petr, Jan Pichl, Tomáš Vyskočil and Jan Šedivỳ (2016). Sentence pair scoring: Towards unified framework for text comprehension. *arXiv preprint arXiv:1603.06127*.

Beil, Florian, Martin Ester and Xiaowei Xu (2002). Frequent term-based text clustering. In: *KDD*. ACM. pp. 436–442.

Bejan, Cosmin Adrian and Sanda Harabagiu (2010). Unsupervised event coreference resolution with rich linguistic features. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. pp. 1412–1422.

Bengio, Yoshua, Jérôme Louradour, Ronan Collobert and Jason Weston (2009). Curriculum learning. In: *Proceedings of the 26th annual international conference on machine learning*. ACM. pp. 41–48.

Berant, Jonathan and Percy Liang (2014). Semantic parsing via paraphrasing.. In: *ACL (1)*. pp. 1415–1425.

Blei, David M, Andrew Y Ng and Michael I Jordan (2003). Latent dirichlet allocation. *JMLR* **3**(Jan), 993–1022.

Bobadilla, Jesús, Fernando Ortega, Antonio Hernando and Abraham Gutiérrez (2013). Recommender systems survey. *Knowledge-based systems* **46**, 109–132.

Bollacker, Kurt, Colin Evans, Praveen Paritosh, Tim Sturge and Jamie Taylor (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM. pp. 1247–1250.

Brin, Sergey (1998). Extracting patterns and relations from the world wide web. In: *International Workshop on The World Wide Web and Databases*. Springer. pp. 172–183.

Brown, Peter F, Vincent J Della Pietra, Stephen A Della Pietra and Robert L Mercer (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics* **19**(2), 263–311.

Bruggermann, Daniel, Yannik Hermey, Carsten Orth, Darius Schneider, Stefan Selzer and Gerasimos Spanakis (2016). Storyline detection and tracking using dynamic latent dirichlet allocation. In: *Proceedings of the 2nd Workshop on Computing News Storylines (CNS 2016)*. pp. 9–19.

Buchta, Christian, Martin Kober, Ingo Feinerer and Kurt Hornik (2012). Spherical k-means clustering. *Journal of Statistical Software* **50**(10), 1–22.

Büttcher, Stefan, Charles LA Clarke and Brad Lushman (2006). Term proximity scoring for ad-hoc retrieval on very large text collections. In: *SIGIR*. ACM. pp. 621–622.

Cao, Ziqiang, Chuwei Luo, Wenjie Li and Sujian Li (2017). Joint copying and restricted generation for paraphrase.. In: *AAAI*. pp. 3152–3158.

Carlson, Andrew, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka and Tom M Mitchell (2010). Toward an architecture for never-ending language learning. In: *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Cer, Daniel, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio and Lucia Specia (2017). Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Chali, Yllias and Sadid A Hasan (2015). Towards topic-to-question generation. *Computational Linguistics* **41**(1), 1–20.

Chang, Pi-Chuan, Michel Galley and Christopher D Manning (2008). Optimizing chinese word segmentation for machine translation performance. In: *ACL*. pp. 224–232.

Chen, Jinxiu, Donghong Ji, Chew Lim Tan and Zhengyu Niu (2006). Relation extraction using label propagation based semi-supervised learning. In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. pp. 129–136.

Chen, Kuang-hua and Hsin-Hsi Chen (1994). Extracting noun phrases from large-scale texts: A hybrid approach and its automatic evaluation. In: *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics. pp. 234–241.

Chen, Yubo, Liheng Xu, Kang Liu, Daojian Zeng and Jun Zhao (2015). Event extraction via dynamic multi-pooling convolutional neural networks. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. pp. 167–176.

Chen, Yubo, Shulin Liu, Xiang Zhang, Kang Liu and Jun Zhao (2017). Automatically labeled data generation for large scale event extraction. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pp. 409–419.

Chollet, François et al. (2015). Keras. `https://github.com/fchollet/keras`.

Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho and Yoshua Bengio (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Clark, Kevin, Urvashi Khandelwal, Omer Levy and Christopher D Manning (2019). What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

Cordeiro, Mário and João Gama (2016). Online social networks event detection: a survey. In: *Solving Large Scale Learning Tasks. Challenges and Algorithms*. pp. 1–41. Springer.

Cui, Peng, Xiao Wang, Jian Pei and Wenwu Zhu (2018). A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering* **31**(5), 833–852.

Culotta, Aron and Jeffrey Sorensen (2004). Dependency tree kernels for relation extraction. In: *Proceedings of the 42nd annual meeting on association for computational linguistics*. ACL. p. 423.

Damonte, Marco, Shay B Cohen and Giorgio Satta (2016). An incremental parser for abstract meaning representation. *arXiv preprint arXiv:1608.06111*.

Danon, Guy and Mark Last (2017). A syntactic approach to domain-specific automatic question generation. *arXiv preprint arXiv:1712.09827*.

De Sa, Christopher, Alex Ratner, Christopher Ré, Jaeho Shin, Feiran Wang, Sen Wu and Ce Zhang (2016). Deepdive: Declarative knowledge base construction. *ACM SIGMOD Record* **45**(1), 60–67.

Deerwester, Scott, Susan T Dumais, George W Furnas, Thomas K Landauer and Richard Harshman (1990). Indexing by latent semantic analysis. *Journal of the American society for information science* **41**(6), 391.

Defferrard, Michaël, Xavier Bresson and Pierre Vandergheynst (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In: *Advances in Neural Information Processing Systems*. pp. 3844–3852.

Denkowski, Michael and Alon Lavie (2014). Meteor universal: Language specific translation evaluation for any target language. In: *Proceedings of the ninth workshop on statistical machine translation*. pp. 376–380.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee and Kristina Toutanova (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dhillon, Inderjit S, Subramanyam Mallela and Dharmendra S Modha (2003). Information-theoretic co-clustering. In: *KDD*. ACM. pp. 89–98.

Ding, Chris, Xiaofeng He and Horst D Simon (2005). On the equivalence of nonnegative matrix factorization and spectral clustering. In: *SDM*. SIAM. pp. 606–610.

Doddington, George R, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel and Ralph M Weischedel (2004). The automatic content extraction (ace) program-tasks, data, and evaluation.. In: *Lrec*. Vol. 2. Lisbon. p. 1.

Du, Xinya and Claire Cardie (2018). Harvesting paragraph-level question-answer pairs from wikipedia. *arXiv preprint arXiv:1805.05942*.

Du, Xinya, Junru Shao and Claire Cardie (2017). Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*.

El-Kishky, Ahmed, Yanglei Song, Chi Wang, Clare R Voss and Jiawei Han (2014). Scalable topical phrase mining from text corpora. *Proceedings of the VLDB Endowment* **8**(3), 305–316.

Erkan, Günes and Dragomir R Radev (2004). Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research* **22**, 457–479.

Ester, Martin, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise.. In: *KDD*. Vol. 96. pp. 226–231.

Estévez, Pablo A, Michel Tesmer, Claudio A Perez and Jacek M Zurada (2009). Normalized mutual information feature selection. *IEEE TNN* **20**(2), 189–201.

Fader, Anthony, Stephen Soderland and Oren Etzioni (2011). Identifying relations for open information extraction. In: *Proceedings of the conference on empirical methods in natural language processing*. ACL. pp. 1535–1545.

Fan, Yixing, Liang Pang, JianPeng Hou, Jiafeng Guo, Yanyan Lan and Xueqi Cheng (2017). Matchzoo: A toolkit for deep text matching. *arXiv preprint arXiv:1707.07270*.

Flanigan, Jeffrey, Sam Thomson, Jaime G Carbonell, Chris Dyer and Noah A Smith (2014). A discriminative graph-based parser for the abstract meaning representation.

Frantzi, Katerina, Sophia Ananiadou and Hideki Mima (2000). Automatic recognition of multi-word terms:. the c-value/nc-value method. *International journal on digital libraries* **3**(2), 115–130.

Fung, Benjamin CM, Ke Wang and Martin Ester (2003). Hierarchical document clustering using frequent itemsets. In: *SDM*. SIAM. pp. 59–70.

Gao, Yifan, Jianan Wang, Lidong Bing, Irwin King and Michael R Lyu (2018). Difficulty controllable question generation for reading comprehension. *arXiv preprint arXiv:1807.03586*.

Gilmer, Justin, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals and George E Dahl (2017). Neural message passing for quantum chemistry. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. pp. 1263–1272.

Goodfellow, Ian J, David Warde-Farley, Mehdi Mirza, Aaron Courville and Yoshua Bengio (2013). Maxout networks. *arXiv preprint arXiv:1302.4389*.

Goodfellow, Ian, Yoshua Bengio and Aaron Courville (2016). *Deep learning*. MIT press.

Grishman, Ralph (1997). Information extraction: Techniques and challenges. In: *Information extraction a multidisciplinary approach to an emerging information technology*. pp. 10–27. Springer.

Grishman, Ralph, David Westbrook and Adam Meyers (2005). Nyu's english ace 2005 system description. *ACE*.

Gu, Jiatao, Zhengdong Lu, Hang Li and Victor OK Li (2016). Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Gulcehre, Caglar, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou and Yoshua Bengio (2016). Pointing the unknown words. *arXiv preprint arXiv:1603.08148*.

GuoDong, Zhou, Su Jian, Zhang Jie and Zhang Min (2005). Exploring various knowledge in relation extraction. In: *Proceedings of the 43rd annual meeting on association for computational linguistics*. ACL. pp. 427–434.

Gupta, Deepak, Kaheer Suleman, Mahmoud Adada, Andrew McNamara and Justin Harris (2019). Improving neural question generation using world knowledge. *arXiv preprint arXiv:1909.03716*.

Han, Fred.X, Di Niu, Kunfeng Lai, Weidong Guo, Yancheng He and Yu Xu (2019). Inferring search queries from web documents via a graph-augmented sequence to attention network. pp. 2792–2798.

He, Hua and Jimmy J Lin (2016). Pairwise word interaction modeling with deep neural networks for semantic similarity measurement.. In: *HLT-NAACL*. pp. 937–948.

He, Xiaofei, Deng Cai, Yuanlong Shao, Hujun Bao and Jiawei Han (2011). Laplacian regularized gaussian mixture model for data clustering. *IEEE TKDE* **23**(9), 1406–1418.

Heilman, Michael (2011). Automatic factual question generation from text. *Language Technologies Institute School of Computer Science Carnegie Mellon University*.

Heilman, Michael and Noah A Smith (2010). Good question! statistical ranking for question generation. In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics. pp. 609–617.

Helsgaun, Keld (2000). An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research* **126**(1), 106–130.

Hensman, Svetlana (2004). Construction of conceptual graph representation of texts. In: *Proceedings of the Student Research Workshop at HLT-NAACL 2004*. Association for Computational Linguistics. pp. 49–54.

Hofmann, Thomas (1999). Probabilistic latent semantic indexing. In: *SIGIR*. ACM. pp. 50–57.

Holtzman, Ari, Jan Buys, Maxwell Forbes and Yejin Choi (2019). The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Honnibal, Matthew and Ines Montani (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Hu, Baotian, Zhengdong Lu, Hang Li and Qingcai Chen (2014). Convolutional neural network architectures for matching natural language sentences. In: *Advances in neural information processing systems*. pp. 2042–2050.

Hu, Wenpeng, Bing Liu, Jinwen Ma, Dongyan Zhao and Rui Yan (2018). Aspect-based question generation.

Hua, Ting, Xuchao Zhang, Wei Wang, Chang-Tien Lu and Naren Ramakrishnan (2016). Automatical storyline generation with help from twitter. In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM. pp. 2383–2388.

Hua, Wen, Zhongyuan Wang, Haixun Wang, Kai Zheng and Xiaofang Zhou (2015). Short text understanding through lexical-semantic analysis. IEEE. pp. 495–506.

Huang, Lifu and Lian'en Huang (2013). Optimized event storyline generation based on mixture-event-aspect model.. In: *EMNLP*. pp. 726–735.

Huang, Po-Sen, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero and Larry Heck (2013). Learning deep structured semantic models for web search using click-through data. In: *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM. pp. 2333–2338.

Huang, Ruihong and Ellen Riloff (2012). Bootstrapped training of event extraction classifiers. In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics. pp. 286–295.

Huang, Zhiheng, Wei Xu and Kai Yu (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Jain, Anil K (2010). Data clustering: 50 years beyond k-means. *Pattern recognition letters* **31**(8), 651–666.

Jang, Eric, Shixiang Gu and Ben Poole (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Ji, Heng and Ralph Grishman (2008). Refining event extraction through cross-document inference. In: *Proceedings of ACL-08: HLT*. pp. 254–262.

Jiang, Chuntao, Frans Coenen, Robert Sanderson and Michele Zito (2010). Text classification using graph mining-based feature extraction. *Knowledge-Based Systems* **23**(4), 302–308.

Jiang, Jyun-Yu, Mingyang Zhang, Cheng Li, Mike Bendersky, Nadav Golbandi and Marc Najork (2019). Semantic text matching for long-form documents.

Joulin, Armand, Edouard Grave, Piotr Bojanowski and Tomas Mikolov (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Kim, Yanghoon, Hwanhee Lee, Joongbo Shin and Kyomin Jung (2018). Improving neural question generation using answer separation. *arXiv preprint arXiv:1809.02393*.

Kim, Yanghoon, Hwanhee Lee, Joongbo Shin and Kyomin Jung (2019). Improving neural question generation using answer separation. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. pp. 6602–6609.

Kingma, Diederik P and Jimmy Ba (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingsbury, Paul and Martha Palmer (2002). From treebank to propbank.. In: *LREC*. pp. 1989–1993.

Kipf, Thomas N and Max Welling (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Knight, Philip A (2008). The sinkhorn–knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications* **30**(1), 261–275.

Konstan, Joseph A (2008). Introduction to recommender systems. In: *2008 ACM SIGMOD International Conference on Management of Data 2008, SIGMOD'08*. p. 1376776.

Koo, Terry, Xavier Carreras and Michael Collins (2008). Simple semi-supervised dependency parsing. In: *Proceedings of ACL-08: HLT*. pp. 595–603.

Koutrika, Georgia (2018). Modern recommender systems: from computing matrices to thinking with neurons. In: *Proceedings of the 2018 International Conference on Management of Data*. ACM. pp. 1651–1654.

Krishna, Kalpesh and Mohit Iyyer (2019). Generating question-answer hierarchies. *arXiv preprint arXiv:1906.02622*.

Krizhevsky, Alex, Ilya Sutskever and Geoffrey E Hinton (2012). Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. pp. 1097–1105.

Kusner, Matt, Yu Sun, Nicholas Kolkin and Kilian Weinberger (2015). From word embeddings to document distances. In: *International Conference on Machine Learning*. pp. 957–966.

Lample, Guillaume and Alexis Conneau (2019). Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Lample, Guillaume, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami and Chris Dyer (2016). Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

LeCun, Yann, Yoshua Bengio and Geoffrey Hinton (2015). Deep learning. *nature* **521**(7553), 436.

Lee, Heeyoung, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu and Dan Jurafsky (2013). Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics* **39**(4), 885–916.

Lee, Heeyoung, Marta Recasens, Angel Chang, Mihai Surdeanu and Dan Jurafsky (2012). Joint entity and event coreference resolution across documents. In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics. pp. 489–500.

Lehmann, Jens, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer et al. (2015). Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web* **6**(2), 167–195.

Leskovec, Jure, Marko Grobelnik and Natasa Milic-Frayling (2004). Learning substructures of document semantic graphs for document summarization.

Li, Yinghao, Wing Pong Robert Luk, Kei Shiu Edward Ho and Fu Lai Korris Chung (2007). Improving weak ad-hoc queries using wikipedia asexternal corpus. In: *SIGIR*. ACM. pp. 797–798.

Lin, Chin-Yew (2004). Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Liu, Angli, Stephen Soderland, Jonathan Bragg, Christopher H Lin, Xiao Ling and Daniel S Weld (2016*a*). Effective crowd annotation for relation extraction. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pp. 897–906.

Liu, Bang, Di Niu, Haojie Wei, Jinghong Lin, Yancheng He, Kunfeng Lai and Yu Xu (2019*a*). Matching article pairs with graphical decomposition and convolutions. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. pp. 6284–6294.

Liu, Bang, Di Niu, Kunfeng Lai, Linglong Kong and Yu Xu (2017). Growing story forest online from massive breaking news. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM. pp. 777–785.

Liu, Bang, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei and Yu Xu (2019*b*). Learning to generate questions by learning what not to generate. In: *The World Wide Web Conference*. ACM. pp. 1106–1118.

Liu, Bang, Ting Zhang, Fred X Han, Di Niu, Kunfeng Lai and Yu Xu (2018*a*). Matching natural language sentences with hierarchical sentence factorization. In: *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. pp. 1237–1246.

Liu, Bang, Weidong Guo, Di Niu, Chaoyue Wang, Shunnan Xu, Jinghong Lin, Kunfeng Lai and Yu Xu (2019*c*). A user-centered concept mining system for query and document understanding at tencent. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '19. ACM. New York, NY, USA. pp. 1831–1841.

Liu, Jialu, Jingbo Shang, Chi Wang, Xiang Ren and Jiawei Han (2015). Mining quality phrases from massive text corpora. In: *SIGMOD*. ACM. pp. 1729–1744.

Liu, Luying, Jianchu Kang, Jing Yu and Zhongliang Wang (2005). A comparative study on unsupervised feature selection methods for text clustering. In: *Natural Language Processing and Knowledge Engineering, 2005. IEEE NLP-KE'05. Proceedings of 2005 IEEE International Conference on*. IEEE. pp. 597–601.

Liu, Shulin, Yubo Chen, Shizhu He, Kang Liu and Jun Zhao (2016*b*). Leveraging framenet to improve automatic event detection. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pp. 2134–2143.

Liu, Xiao, Zhunchen Luo and Heyan Huang (2018*b*). Jointly multiple events extraction via attention-based graph information aggregation. *arXiv preprint arXiv:1809.09078*.

Liu, Xueqing, Yangqiu Song, Shixia Liu and Haixun Wang (2012). Automatic taxonomy construction from keywords. In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. pp. 1433–1441.

Liu, Zhiyuan, Xinxiong Chen, Yabin Zheng and Maosong Sun (2011). Automatic keyphrase extraction by bridging vocabulary gap. In: *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*. ACL. pp. 135–144.

Luong, Minh-Thang, Hieu Pham and Christopher D Manning (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Maddison, Chris J, Daniel Tarlow and Tom Minka (2014). A* sampling. In: *Advances in Neural Information Processing Systems*. pp. 3086–3094.

Manning, Christopher D, Christopher D Manning and Hinrich Schütze (1999). *Foundations of statistical natural language processing*. MIT press.

Manning, Christopher, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard and David McClosky (2014). The stanford corenlp natural language processing toolkit. In: *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. pp. 55–60.

Marcheggiani, Diego and Ivan Titov (2017). Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv preprint arXiv:1703.04826*.

Marelli, Marco, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi and Roberto Zamparelli (2014). A sick cure for the evaluation of compositional distributional semantic models.. In: *LREC*. pp. 216–223.

Matthew Honnibal (2015). spacy: Industrial-strength natural language processing (nlp) with python and cython. `https://spacy.io`. [Online; accessed 3-November-2018].

McClosky, David, Mihai Surdeanu and Christopher D Manning (2011). Event extraction as dependency parsing. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics. pp. 1626–1635.

Mei, Qiaozhu and ChengXiang Zhai (2005). Discovering evolutionary theme patterns from text: an exploration of temporal text mining. In: *KDD*. ACM. pp. 198–207.

Mihalcea, Rada and Paul Tarau (2004). Textrank: Bringing order into texts. ACL.

Mikolov, Tomas, Kai Chen, Greg Corrado and Jeffrey Dean (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Miller, George A (1995). Wordnet: a lexical database for english. *Communications of the ACM* **38**(11), 39–41.

Mitra, Bhaskar, Fernando Diaz and Nick Craswell (2017). Learning to match using local and distributed representations of text for web search. In: *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. pp. 1291–1299.

Miwa, Makoto, Rune Sætre, Jin-Dong Kim and Jun'ichi Tsujii (2010). Event extraction with complex event classification using rich features. *Journal of bioinformatics and computational biology* **8**(01), 131–146.

Mostafazadeh, Nasrin, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He and Lucy Vanderwende (2016). Generating natural questions about an image. *arXiv preprint arXiv:1603.06059*.

Mueller, Jonas and Aditya Thyagarajan (2016). Siamese recurrent architectures for learning sentence similarity. In: *Thirtieth AAAI Conference on Artificial Intelligence*.

Nadeau, David and Satoshi Sekine (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes* **30**(1), 3–26.

Nallapati, Ramesh, Ao Feng, Fuchun Peng and James Allan (2004). Event threading within news topics. In: *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. ACM. pp. 446–453.

Navigli, Roberto, Paola Velardi and Stefano Faralli (2011). A graph-based algorithm for inducing lexical taxonomies from scratch. In: *Twenty-Second International Joint Conference on Artificial Intelligence*.

Neculoiu, Paul, Maarten Versteegh, Mihai Rotaru and Textkernel BV Amsterdam (2016). Learning text similarity with siamese recurrent networks. *ACL 2016* p. 148.

Nguyen, Thien Huu and Ralph Grishman (2016). Modeling skip-grams for event detection with convolutional neural networks. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pp. 886–891.

Nguyen, Thien Huu, Kyunghyun Cho and Ralph Grishman (2016). Joint event extraction via recurrent neural networks. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pp. 300–309.

Nikolentzos, Giannis, Polykarpos Meladianos, François Rousseau, Yannis Stavrakas and Michalis Vazirgiannis (2017). Shortest-path graph kernels for document similarity. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pp. 1890–1900.

Ohsawa, Yukio, Nels E Benson and Masahiko Yachida (1998). Keygraph: Automatic indexing by co-occurrence graph based on building construction metaphor. In: *Research and Technology Advances in Digital Libraries, 1998. ADL 98. Proceedings. IEEE International Forum on*. IEEE. pp. 12–18.

Page, Lawrence, Sergey Brin, Rajeev Motwani and Terry Winograd (1999). The pagerank citation ranking: Bringing order to the web.. Technical report. Stanford InfoLab.

Paltoglou, Georgios and Mike Thelwall (2010). A study of information retrieval weighting schemes for sentiment analysis. In: *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics. pp. 1386–1395.

Pang, Liang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan and Xueqi Cheng (2016). Text matching as image recognition.. In: *AAAI*. pp. 2793–2799.

Papineni, Kishore, Salim Roukos, Todd Ward and Wei-Jing Zhu (2002). Bleu: a method for automatic evaluation of machine translation. In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics. pp. 311–318.

Parameswaran, Aditya, Hector Garcia-Molina and Anand Rajaraman (2010). Towards the web of concepts: Extracting concepts from large datasets. *Proceedings of the VLDB Endowment* **3**(1-2), 566–577.

Park, Hae-Sang and Chi-Hyuck Jun (2009). A simple and fast algorithm for k-medoids clustering. *Expert systems with applications* **36**(2), 3336–3341.

Park, Youngja, Roy J Byrd and Branimir K Boguraev (2002). Automatic glossary extraction: beyond terminology identification. In: *COLING 2002: The 19th International Conference on Computational Linguistics*.

Paszke, Adam, Sam Gross, Soumith Chintala and Gregory Chanan (2017). Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration. *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*.

Paul, Christian, Achim Rettinger, Aditya Mogadala, Craig A Knoblock and Pedro Szekely (2016). Efficient graph-based document similarity. In: *European Semantic Web Conference*. Springer. pp. 334–349.

Pawar, Sachin, Girish K Palshikar and Pushpak Bhattacharyya (2017). Relation extraction: A survey. *arXiv preprint arXiv:1712.05191*.

Pennington, Jeffrey, Richard Socher and Christopher Manning (2014). Glove: Global vectors for word representation. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pp. 1532–1543.

Ponzanelli, Luca, Andrea Mocci and Michele Lanza (2015). Summarizing complex development artifacts by mining heterogeneous data. In: *Proceedings of the 12th Working Conference on Mining Software Repositories*. IEEE Press. pp. 401–405.

Poon, Hoifung and Pedro Domingos (2010). Unsupervised ontology induction from text. In: *Proceedings of the 48th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. pp. 296–305.

Postma, Marten, Filip Ilievski and Piek Vossen (2018). Semeval-2018 task 5: Counting events and participants in the long tail. In: *Proceedings of The 12th International Workshop on Semantic Evaluation*. pp. 70–80.

Pourdamghani, Nima, Yang Gao, Ulf Hermjakob and Kevin Knight (2014). Aligning english strings with abstract meaning representation graphs.. In: *EMNLP*. pp. 425–429.

Punyakanok, Vasin and Dan Roth (2001). The use of classifiers in sequential inference. In: *Advances in Neural Information Processing Systems*. pp. 995–1001.

Putra, Jan Wira Gotama and Takenobu Tokunaga (2017). Evaluating text coherence based on semantic similarity graph. In: *Proceedings of TextGraphs-11: the Workshop on Graph-based Methods for Natural Language Processing*. pp. 76–85.

Qiu, Xipeng and Xuanjing Huang (2015). Convolutional neural tensor network architecture for community-based question answering.. In: *IJCAI*. pp. 1305–1311.

Quirk, Chris, Chris Brockett and William Dolan (2004). Monolingual machine translation for paraphrase generation. In: *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei and Ilya Sutskever (2019). Language models are unsupervised multitask learners. *OpenAI Blog*.

Radicchi, Filippo, Claudio Castellano, Federico Cecconi, Vittorio Loreto and Domenico Parisi (2004). Defining and identifying communities in networks. *PNAS* **101**(9), 2658–2663.

Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev and Percy Liang (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Rajpurkar, Pranav, Robin Jia and Percy Liang (2018). Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Řehůřek, Radim and Petr Sojka (2010). Software Framework for Topic Modelling with Large Corpora. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA. Valletta, Malta. pp. 45–50. `http://is.muni.cz/publication/884893/en`.

Ren, Xiang, Zeqiu Wu, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, Tarek F Abdelzaher and Jiawei Han (2017). Cotype: Joint extraction of typed entities and relations with knowledge bases. In: *WWW*. International World Wide Web Conferences Steering Committee. pp. 1015–1024.

Rink, Bryan, Cosmin Adrian Bejan and Sanda M Harabagiu (2010). Learning textual graph patterns to detect causal event relations.. In: *FLAIRS Conference*.

Ritter, Alan, Oren Etzioni, Sam Clark et al. (2012). Open domain event extraction from twitter. In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. pp. 1104–1112.

Ritter, Alan, Sam Clark, Oren Etzioni et al. (2011). Named entity recognition in tweets: an experimental study. In: *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics. pp. 1524–1534.

Robertson, Stephen E and Steve Walker (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In: *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. Springer-Verlag New York, Inc.. pp. 232–241.

Robertson, Stephen, Hugo Zaragoza et al. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval* **3**(4), 333–389.

Rose, Stuart, Dave Engel, Nick Cramer and Wendy Cowley (2010). Automatic keyword extraction from individual documents. *Text Mining* pp. 1–20.

Rosenberg, Andrew and Julia Hirschberg (2007). V-measure: A conditional entropy-based external cluster evaluation measure.. In: *EMNLP-CoNLL*. Vol. 7. pp. 410–420.

Rousseau, François and Michalis Vazirgiannis (2013). Graph-of-word and tw-idf: new approach to ad hoc ir. In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM. pp. 59–68.

Rousseau, François, Emmanouil Kiagias and Michalis Vazirgiannis (2015). Text categorization as a graph classification problem.. In: *ACL (1)*. pp. 1702–1712.

Rubner, Yossi, Carlo Tomasi and Leonidas J Guibas (2000). The earth mover's distance as a metric for image retrieval. *International journal of computer vision* **40**(2), 99–121.

Ruder, Sebastian (2019). Neural Transfer Learning for Natural Language Processing. PhD thesis. National University of Ireland, Galway.

Sachan, Mrinmaya and Eric Xing (2018). Self-training for jointly learning to ask and answer questions. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. pp. 629–640.

Salle, Alexandre, Marco Idiart and Aline Villavicencio (2016). Enhancing the lexvec distributed word representation model using positional contexts and external memory. *arXiv preprint arXiv:1606.01283*.

Sayyadi, Hassan and Louiqa Raschid (2013). A graph analytical approach for topic detection. *ACM TOIT* **13**(2), 4.

Sayyadi, Hassan, Matthew Hurst and Alexey Maykov (2009). Event detection and tracking in social streams.. In: *Icwsm*.

Schenker, Adam, Mark Last, Horst Bunke and Abraham Kandel (2003). Clustering of web documents using a graph model. *SERIES IN MACHINE PERCEPTION AND ARTIFICIAL INTELLIGENCE* **55**, 3–18.

Schlichtkrull, Michael, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov and Max Welling (2017). Modeling relational data with graph convolutional networks. *arXiv preprint arXiv:1703.06103*.

Schlichtkrull, Michael, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov and Max Welling (2018). Modeling relational data with graph convolutional networks. In: *European Semantic Web Conference*. Springer. pp. 593–607.

Schuhmacher, Michael and Simone Paolo Ponzetto (2014). Knowledge-based graph document modeling. In: *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM. pp. 543–552.

Schuler, Karin Kipper (2005). Verbnet: A broad-coverage, comprehensive verb lexicon.

Serban, Iulian Vlad, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville and Yoshua Bengio (2016). Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *arXiv preprint arXiv:1603.06807*.

Settles, Burr (2009). Active learning literature survey. Technical report. University of Wisconsin-Madison Department of Computer Sciences.

Severyn, Aliaksei and Alessandro Moschitti (2015). Learning to rank short text pairs with convolutional deep neural networks. In: *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. ACM. pp. 373–382.

Sha, Lei, Feng Qian, Baobao Chang and Zhifang Sui (2018). Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In: *Thirty-Second AAAI Conference on Artificial Intelligence*.

Shahaf, Dafna, Carlos Guestrin and Eric Horvitz (2012). Trains of thought: Generating information maps. In: *Proceedings of the 21st international conference on World Wide Web*. ACM. pp. 899–908.

Shahaf, Dafna, Jaewon Yang, Caroline Suen, Jeff Jacobs, Heidi Wang and Jure Leskovec (2013). Information cartography: creating zoomable, large-scale maps of information. In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. pp. 1097–1105.

Shang, Jingbo, Jialu Liu, Meng Jiang, Xiang Ren, Clare R Voss and Jiawei Han (2018). Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering* **30**(10), 1825–1837.

Shao, Yang (2017). Hcti at semeval-2017 task 1: Use convolutional neural network to evaluate semantic textual similarity. In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pp. 130–133.

Shen, Yelong, Xiaodong He, Jianfeng Gao, Li Deng and Grégoire Mesnil (2014). Learning semantic representations using convolutional neural networks for web search. In: *Proceedings of the 23rd International Conference on World Wide Web*. ACM. pp. 373–374.

Shum, Heung-Yeung, Xiao-dong He and Di Li (2018). From eliza to xiaoice: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering* **19**(1), 10–26.

Singhal, Amit (2012). Introducing the knowledge graph: things, not strings. *Official google blog*.

Slonim, Noam and Naftali Tishby (2000). Document clustering using word clusters via the information bottleneck method. In: *SIGIR*. ACM. pp. 208–215.

Smirnova, Alisa and Philippe Cudré-Mauroux (2018). Relation extraction using distant supervision: A survey. *ACM Computing Surveys (CSUR)* **51**(5), 106.

Song, Linfeng, Zhiguo Wang, Wael Hamza, Yue Zhang and Daniel Gildea (2018*a*). Leveraging context information for natural question generation. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Vol. 2. pp. 569–574.

Song, Yan, Shuming Shi, Jing Li and Haisong Zhang (2018*b*). Directional skip-gram: Explicitly distinguishing left and right context for word embeddings. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. ACL. pp. 175–180.

Song, Yan, Shuming Shi, Jing Li and Haisong Zhang (2018*c*). Directional skip-gram: explicitly distinguishing left and right context for word embeddings. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. pp. 175–180.

Song, Yangqiu, Haixun Wang, Zhongyuan Wang, Hongsong Li and Weizhu Chen (2011). Short text conceptualization using a probabilistic knowledgebase. In: *Proceedings of the twenty-second international joint conference on artificial intelligence-volume volume three*. AAAI Press. pp. 2330–2336.

Spitzer, Frank (2013). *Principles of random walk*. Vol. 34. Springer Science & Business Media.

Su, Bing and Gang Hua (2017). Order-preserving wasserstein distance for sequence matching. In: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*. pp. 1049–1057.

Subramanian, Sandeep, Tong Wang, Xingdi Yuan, Saizheng Zhang, Yoshua Bengio and Adam Trischler (2017). Neural models for key phrase detection and question generation. *arXiv preprint arXiv:1706.04560*.

Suchanek, Fabian M, Gjergji Kasneci and Gerhard Weikum (2007). Yago: a core of semantic knowledge. In: *Proceedings of the 16th international conference on World Wide Web*. ACM. pp. 697–706.

Sun, Xingwu, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma and Shi Wang (2018). Answer-focused and position-aware neural question generation. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. pp. 3930–3939.

Sun, Yu, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu and Haifeng Wang (2019). Ernie 2.0: A continual pre-training framework for language understanding. *arXiv preprint arXiv:1907.12412*.

Sundermeyer, Martin, Ralf Schlüter and Hermann Ney (2012). Lstm neural networks for language modeling. In: *Thirteenth Annual Conference of the International Speech Communication Association*.

Sutskever, Ilya, Oriol Vinyals and Quoc V Le (2014). Sequence to sequence learning with neural networks. In: *Advances in neural information processing systems*. pp. 3104–3112.

Tanev, Hristo, Jakub Piskorski and Martin Atkinson (2008). Real-time news event extraction for global crisis monitoring. In: *International Conference on Application of Natural Language to Information Systems*. Springer. pp. 207–218.

Tang, Duyu, Nan Duan, Tao Qin, Zhao Yan and Ming Zhou (2017). Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*.

Tang, Duyu, Nan Duan, Zhao Yan, Zhirui Zhang, Yibo Sun, Shujie Liu, Yuanhua Lv and Ming Zhou (2018). Learning to collaborate for question answering and asking. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Vol. 1. pp. 1564–1574.

Trischler, Adam, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman and Kaheer Suleman (2016). Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Valenzuela-Escárcega, Marco A, Gus Hahn-Powell, Mihai Surdeanu and Thomas Hicks (2015). A domain-independent rule-based framework for event extraction. In: *Proceedings of ACL-IJCNLP 2015 System Demonstrations*. pp. 127–132.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser and Illia Polosukhin (2017). Attention is all you need. In: *Advances in neural information processing systems*. pp. 5998–6008.

Vossen, Piek, Tommaso Caselli and Yiota Kontzopoulou (2015). Storylines for structuring massive streams of news. In: *Proceedings of the First Workshop on Computing News Storylines*. pp. 40–49.

Wan, Shengxian, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang and Xueqi Cheng (2016). A deep architecture for semantic matching with multiple positional sentence representations.. In: *AAAI*. Vol. 16. pp. 2835–2841.

Wang, Chuan, Nianwen Xue and Sameer Pradhan (2015*a*). Boosting transition-based amr parsing with refined actions and auxiliary analyzers.. In: *ACL (2)*. pp. 857–862.

Wang, Dingding, Tao Li and Mitsunori Ogihara (2012). Generating pictorial storylines via minimum-weight connected dominating set approximation in multi-view graphs.. In: *AAAI*. Citeseer.

Wang, Lu, Claire Cardie and Galen Marchetti (2016). Socially-informed timeline generation for complex events. *arXiv preprint arXiv:1606.05699*.

Wang, Shuohang and Jing Jiang (2016). A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*.

Wang, Tong, Xingdi Yuan and Adam Trischler (2017*a*). A joint model for question answering and question generation. *arXiv preprint arXiv:1706.01450*.

Wang, Yujing, Xiaochuan Ni, Jian-Tao Sun, Yunhai Tong and Zheng Chen (2011). Representing document as dependency graph for document clustering. In: *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM. pp. 2177–2180.

Wang, Zhiguo, Wael Hamza and Radu Florian (2017*b*). Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*.

Wang, Zhongyuan, Kejun Zhao, Haixun Wang, Xiaofeng Meng and Ji-Rong Wen (2015*b*). Query understanding through knowledge-based conceptualization. In: *IJCAI*. pp. 3264–3270.

Watanabe, Kazufumi, Masanao Ochi, Makoto Okabe and Rikio Onai (2011). Jasmine: a real-time local-event detection system based on geolocation information propagated to microblogs. In: *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM. pp. 2541–2544.

Wikipedia (2017). Spearman's rank correlation coefficient — wikipedia, the free encyclopedia. [Online; accessed 31-October-2017].

Wikipedia contributors (2019). Bipartite graph — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Bipartite_graph&oldid=915607649`. [Online; accessed 29-September-2019].

Winograd, Terry (1972). Understanding natural language. *Cognitive psychology* **3**(1), 1–191.

Witten, Ian H and Olena Medelyan (2006). Thesaurus based automatic keyphrase indexing. In: *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'06)*. IEEE. pp. 296–297.

Witten, Ian H, Gordon W Paynter, Eibe Frank, Carl Gutwin and Craig G Nevill-Manning (2005). Kea: Practical automated keyphrase extraction. In: *Design and Usability of Digital Libraries: Case Studies in the Asia Pacific*. pp. 129–152. IGI Global.

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz and Jamie Brew (2019). Transformers: State-of-the-art natural language processing.

Wu, Ho Chung, Robert Wing Pong Luk, Kam Fai Wong and Kui Lam Kwok (2008). Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)* **26**(3), 13.

Wu, Wentao, Hongsong Li, Haixun Wang and Kenny Q Zhu (2012). Probase: A probabilistic taxonomy for text understanding. In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM. pp. 481–492.

Wu, Yu, Wei Wu, Can Xu and Zhoujun Li (2018). Knowledge enhanced hybrid neural network for text matching. In: *Thirty-Second AAAI Conference on Artificial Intelligence*.

Xu, Bo, Yong Xu, Jiaqing Liang, Chenhao Xie, Bin Liang, Wanyun Cui and Yanghua Xiao (2017). Cn-dbpedia: A never-ending chinese knowledge extraction system. In: *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer. pp. 428–438.

Xu, Shize, Shanshan Wang and Yan Zhang (2013). Summarizing complex events: a cross-modal solution of storylines extraction and reconstruction.. In: *EMNLP*. pp. 1281–1291.

Xu, Wei, Xin Liu and Yihong Gong (2003). Document clustering based on non-negative matrix factorization. In: *SIGIR*. ACM. pp. 267–273.

Yan, Rui, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li and Yan Zhang (2011). Evolutionary timeline summarization: a balanced optimization framework via iterative substitution. In: *SIGIR*. ACM. pp. 745–754.

Yan, Yulan, Naoaki Okazaki, Yutaka Matsuo, Zhenglu Yang and Mitsuru Ishizuka (2009). Unsupervised relation extraction by mining wikipedia texts using information from the web. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*. ACL. pp. 1021–1029.

Yang, Bishan and Tom Mitchell (2016). Joint extraction of events and entities within a document context. *arXiv preprint arXiv:1609.03632*.

Yang, Christopher C, Xiaodong Shi and Chih-Ping Wei (2009). Discovering event evolution graphs from news corpora. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* **39**(4), 850–863.

Yang, Yiming, Jaime Carbonell, Ralf Brown, John Lafferty, Thomas Pierce and Thomas Ault (2002). Multi-strategy learning for topic detection and tracking. In: *Topic detection and tracking*. pp. 85–114. Springer.

Yang, Zhilin, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov and Quoc V Le (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Yao, Kaichun, Libo Zhang, Tiejian Luo, Lili Tao and Yanjun Wu (2018). Teaching machines to ask questions.. In: *IJCAI*. pp. 4546–4552.

Yu, Lei, Karl Moritz Hermann, Phil Blunsom and Stephen Pulman (2014). Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*.

Yuan, Xingdi, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Sandeep Subramanian, Saizheng Zhang and Adam Trischler (2017). Machine comprehension by text-to-text neural question generation. *arXiv preprint arXiv:1705.02012*.

Zeng, Daojian, Kang Liu, Yubo Chen and Jun Zhao (2015). Distant supervision for relation extraction via piecewise convolutional neural networks. In: *EMNLP*. pp. 1753–1762.

Zhang, Chao, Fangbo Tao, Xiusi Chen, Jiaming Shen, Meng Jiang, Brian Sadler, Michelle Vanni and Jiawei Han (2018*a*). Taxogen: Unsupervised topic taxonomy construction by adaptive term embedding and clustering. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM. pp. 2701–2709.

Zhang, Shiyue and Mohit Bansal (2019). Addressing semantic drift in question generation for semi-supervised question answering. *arXiv preprint arXiv:1909.06356*.

Zhang, Ting, Bang Liu, Di Niu, Kunfeng Lai and Yu Xu (2018*b*). Multiresolution graph attention networks for relevance matching. In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM. pp. 933–942.

Zhang, Yuhao, Peng Qi and Christopher D Manning (2018*c*). Graph convolution over pruned dependency trees improves relation extraction. *arXiv preprint arXiv:1809.10185*.

Zhang, Ziqi, José Iria, Christopher Brewster and Fabio Ciravegna (2008). A comparative evaluation of term recognition algorithms.. In: *LREC*. Vol. 5.

Zhao, Yao, Xiaochuan Ni, Yuanyuan Ding and Qifa Ke (2018). Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. pp. 3901–3910.

Zheng, Zhaohui, Keke Chen, Gordon Sun and Hongyuan Zha (2007). A regression framework for learning ranking functions using relative relevance judgments. In: *SIGIR*. ACM. pp. 287–294.

Zhou, Deyu, Haiyang Xu and Yulan He (2015). An unsupervised bayesian modelling approach for storyline detection on news articles.. In: *EMNLP*. pp. 1943–1948.

Zhou, Qingyu, Nan Yang, Furu Wei and Ming Zhou (2018). Sequential copying networks. *arXiv preprint arXiv:1807.02301*.

Zhou, Qingyu, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao and Ming Zhou (2017). Neural question generation from text: A preliminary study. In: *National CCF Conference on Natural Language Processing and Chinese Computing*. Springer. pp. 662–671.

Zhou, Wenjie, Minghua Zhang and Yunfang Wu (2019*a*). Multi-task learning with language modeling for question generation. *arXiv preprint arXiv:1908.11813*.

Zhou, Wenjie, Minghua Zhang and Yunfang Wu (2019*b*). Question-type driven question generation. *arXiv preprint arXiv:1909.00140*.