Natural Language Processing with Deep Learning

IFT6289, Winter 2022

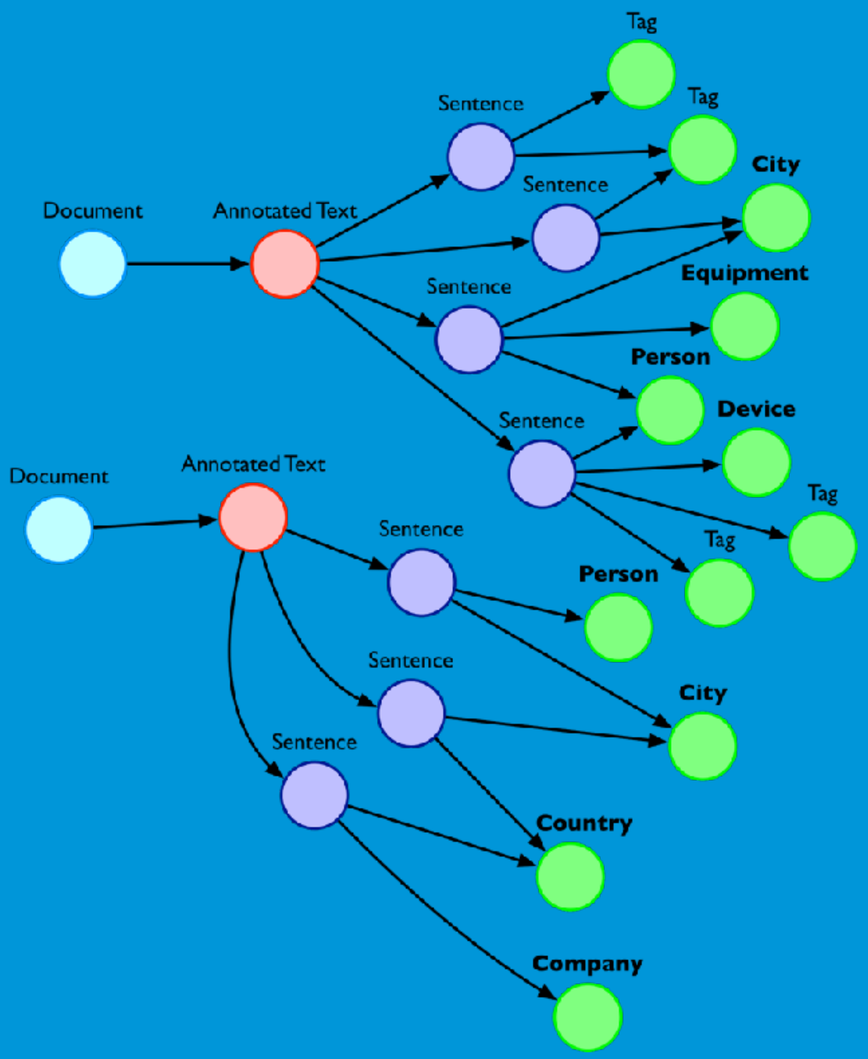
Lecture 13: GNNs and Graph-based NLP

Bang Liu

Lecture outline

1. Why graphs for NLP?
2. Modeling text as graphs
3. Graph neural networks
4. Text clustering: Story Forest for fine-grained events detection and organization
5. Text matching: matching article pairs with graphical decomposition and convolutions

Why Graphs for NLP?

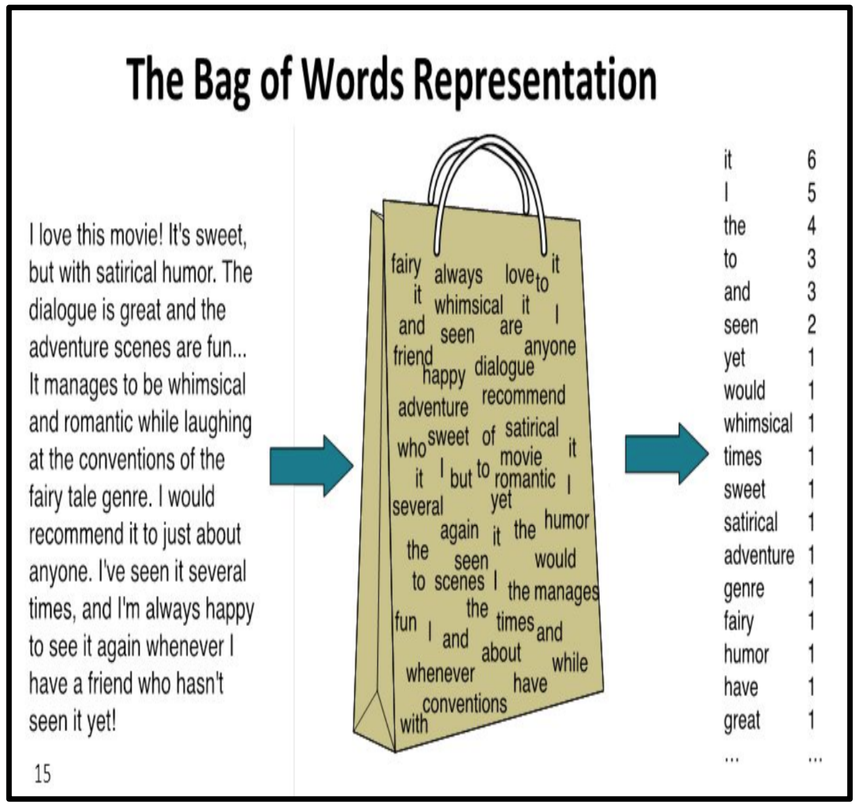


Representation and Computation

How to represent text

How to compute

Bag-of-Words

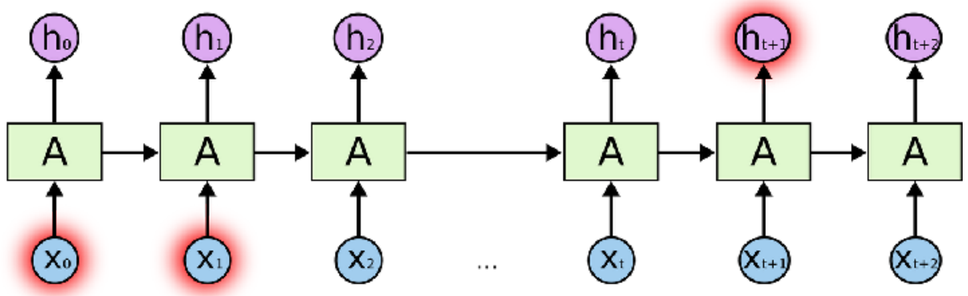


Bag-of-Words

Statistics

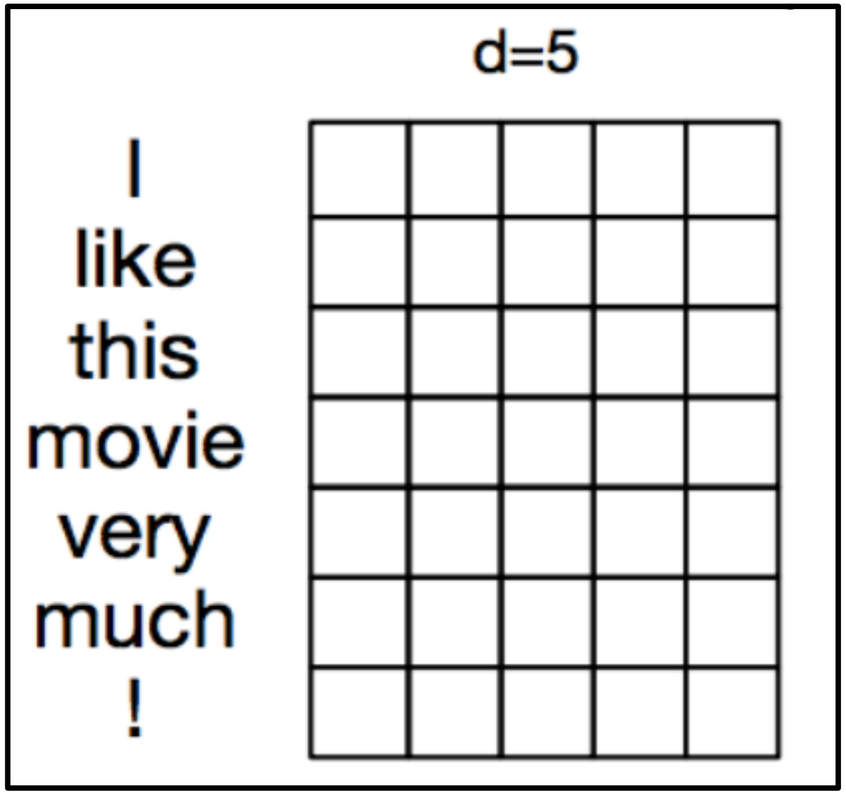
Word Vectors

Word Vector

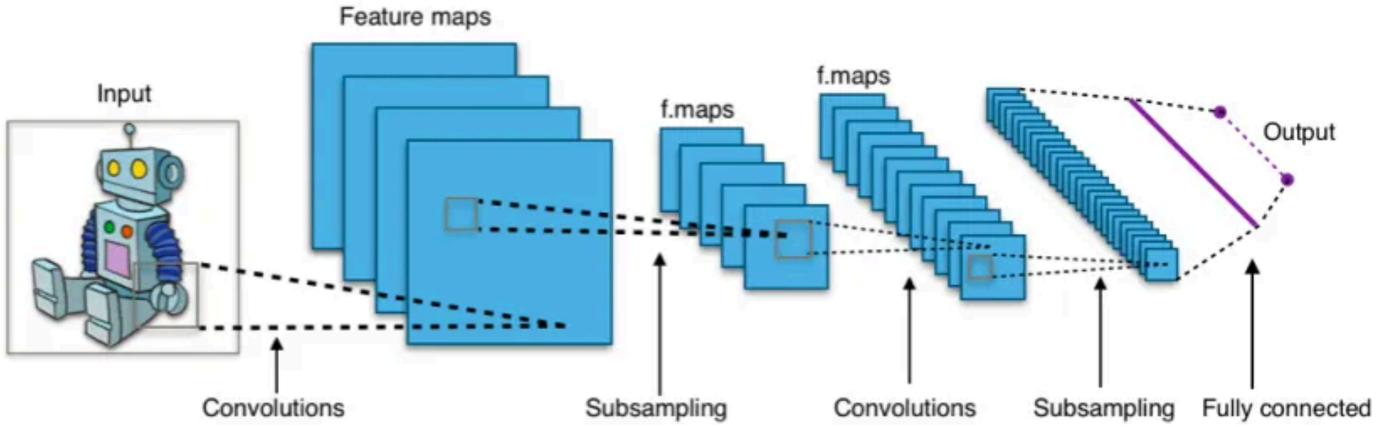


Recurrent Neural Networks (RNN)

Matrix



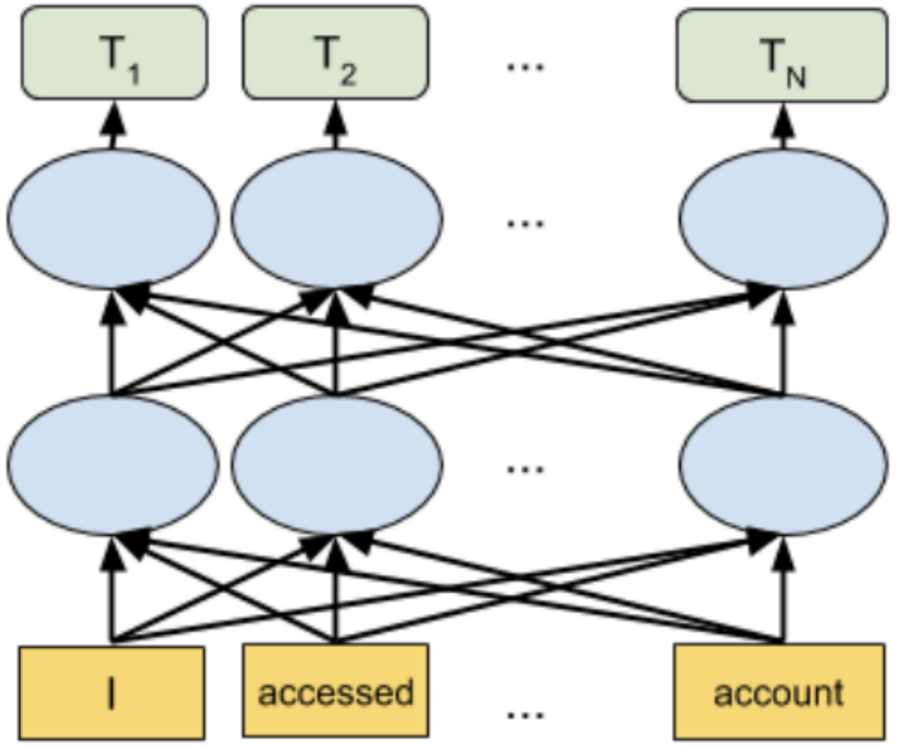
Matrix



Convolutional Neural Networks (CNN)

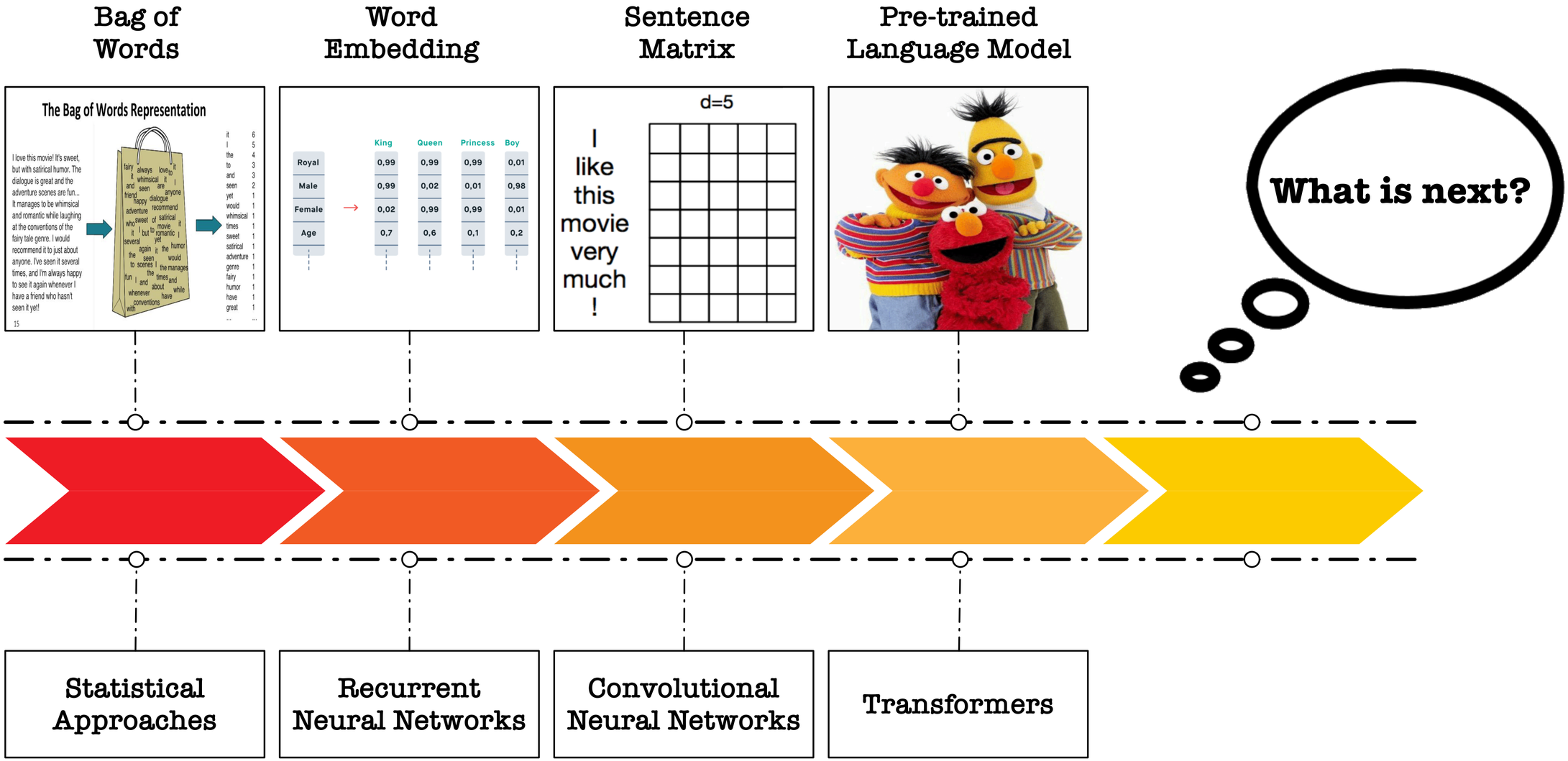
Language Models

Pre-trained Language Models

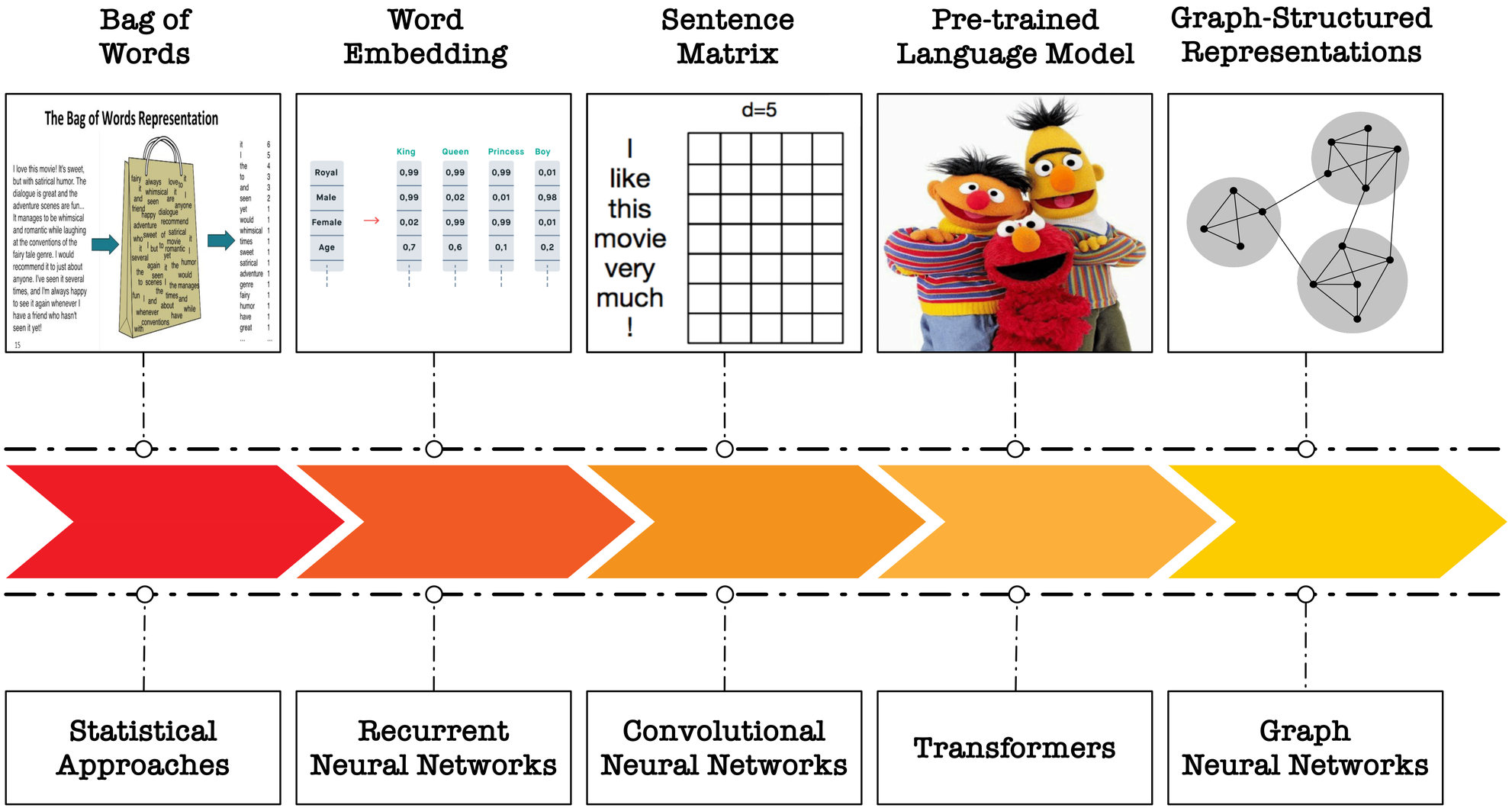


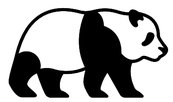
Transformers

Neural History of NLP



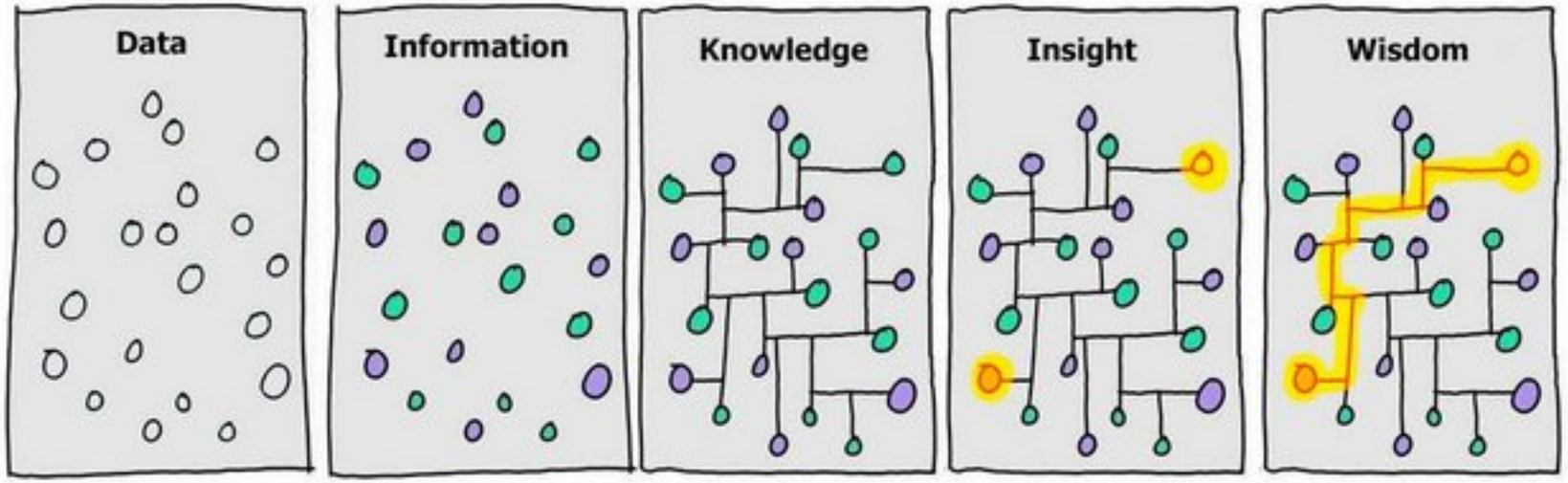
New Trends in NLP





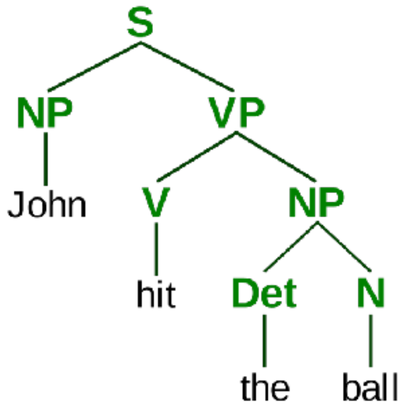
Why Graphs: Relation Matters

- Data in its native form is sparse, distributed and unstructured – it is **chaotic**.
- Transformation and organization produces **information**.
- **Knowledge** is connected information.
- Identify meaningful pieces of information and relate them to each other leads to **insight** and **wisdom**

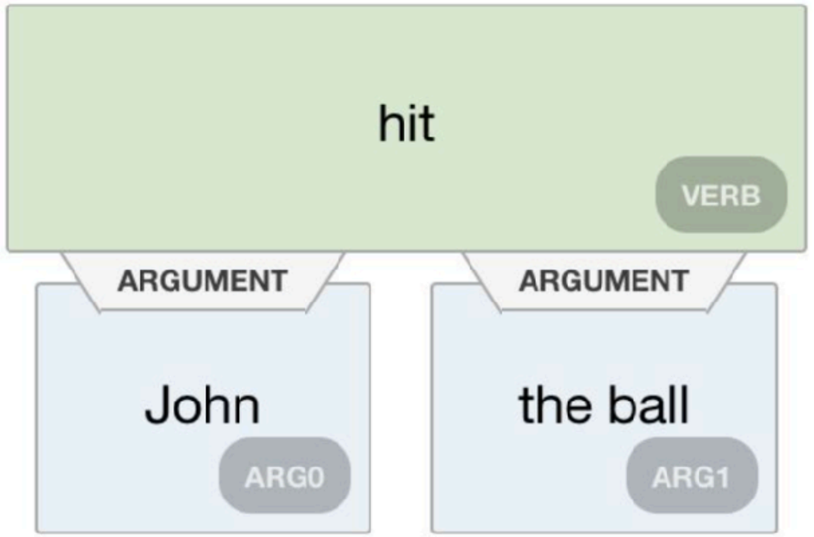


[Cartoon by David Somerville, based on a two pane version by Hugh McLeod.]

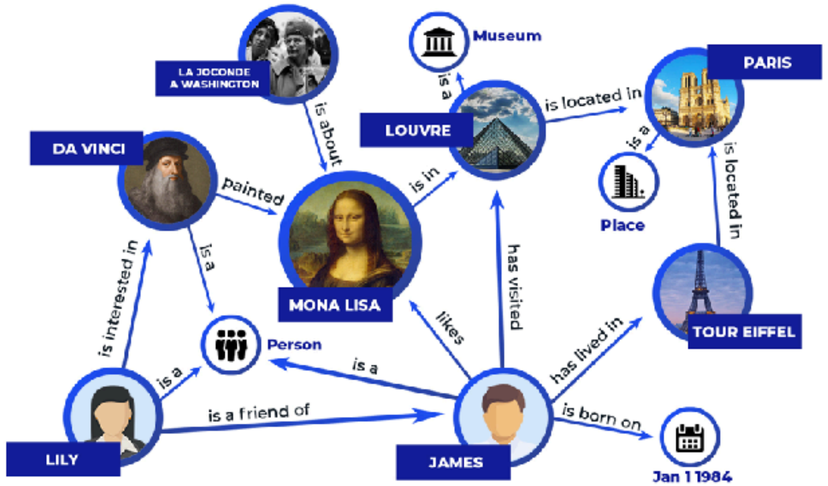
Why Graphs: Graphs are Everywhere in NLP



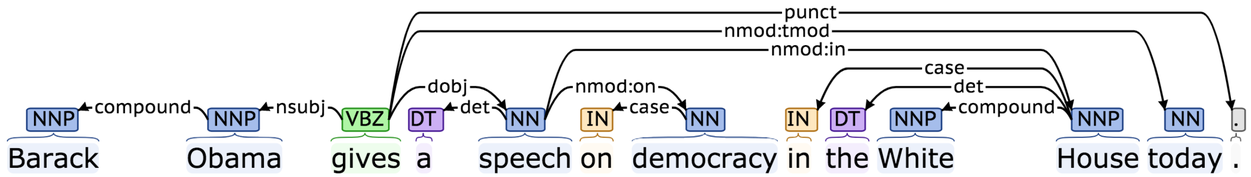
Constituency-based parse tree



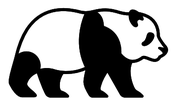
Semantic structure



Knowledge graph



Dependency-based parse tree



Why Graphs: Nature of Language

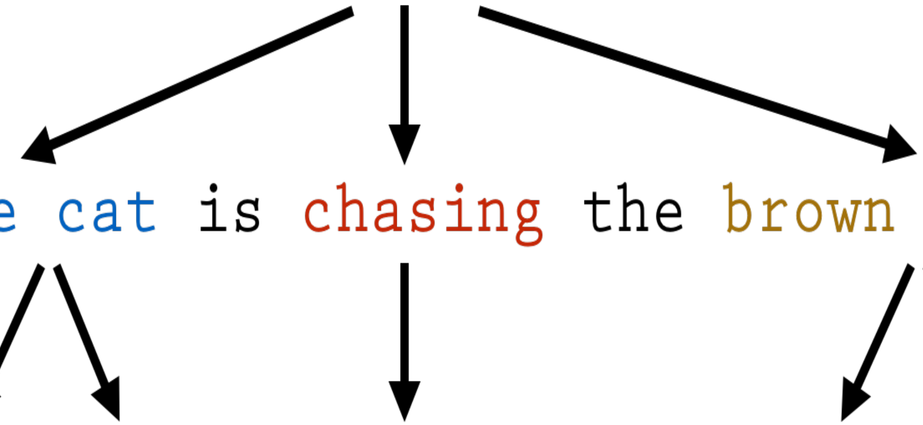
Natural language is flexible, compositional, hierarchical

Sentence A:

The brown mouse is being chased by the blue cat.

Sentence B:

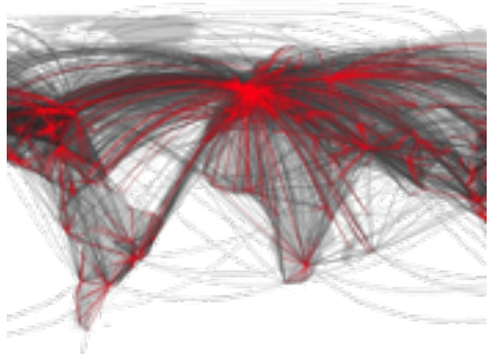
The blue cat is chasing the brown mouse.



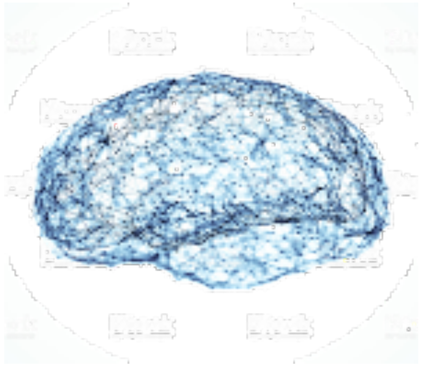
The blue cat is chasing the brown mouse.

Why Graphs: Nature of the World

Social Graphs

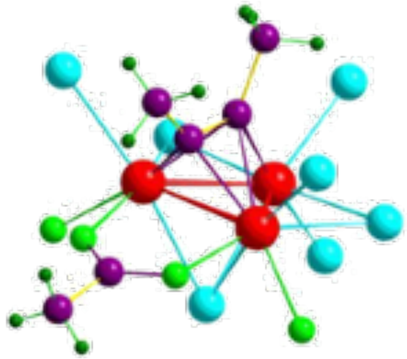


Transportation Graphs

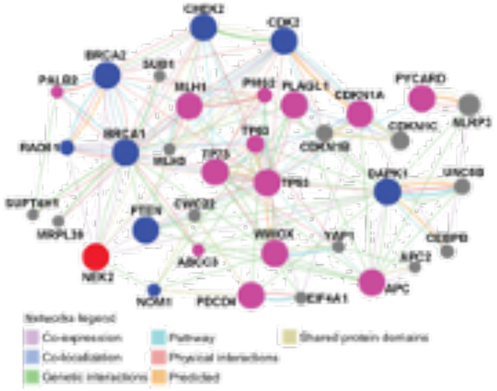


Brain Graphs

Web Graphs

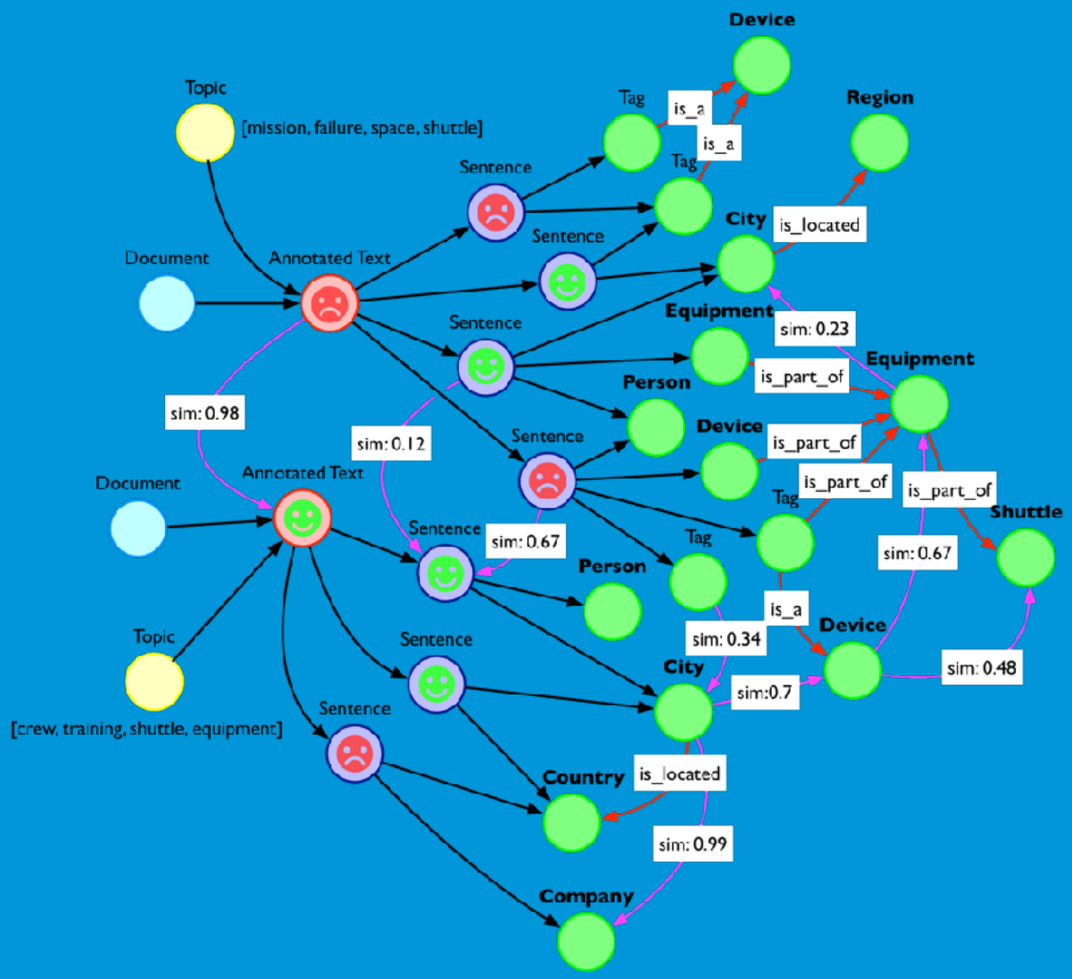


Molecular Graphs



Gene Graphs

Modeling Text as Graphs



NLP from a Graph Perspective

How to represent text
by graphs

How to compute
via graph modeling

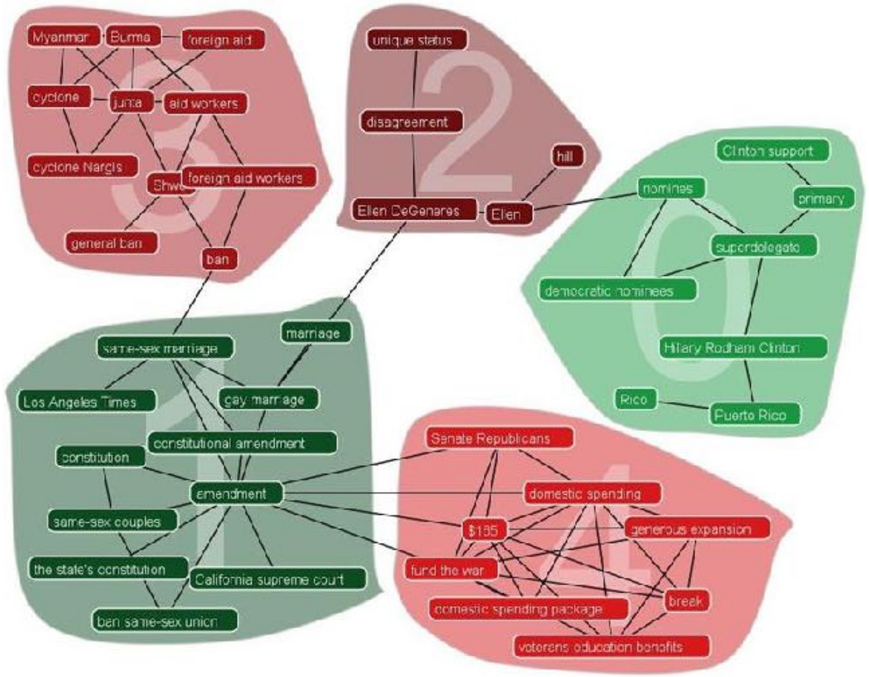
NLP from a Graph Perspective

Compute

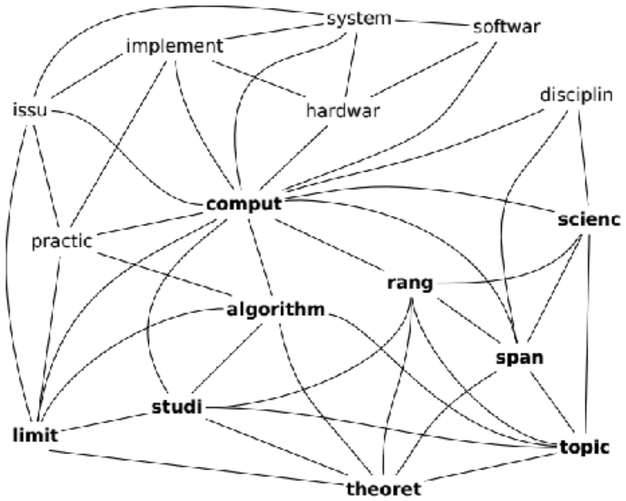
How to represent text
by graphs

How to compute
via graph modeling

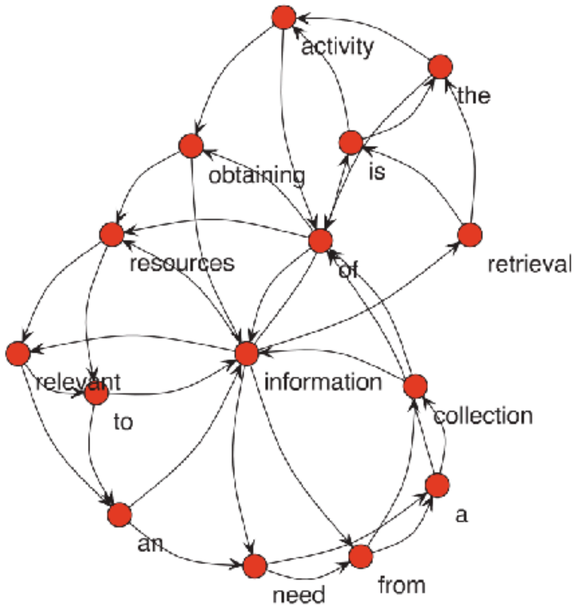
Word Graph



Sayyadi et al., TOIT 2013



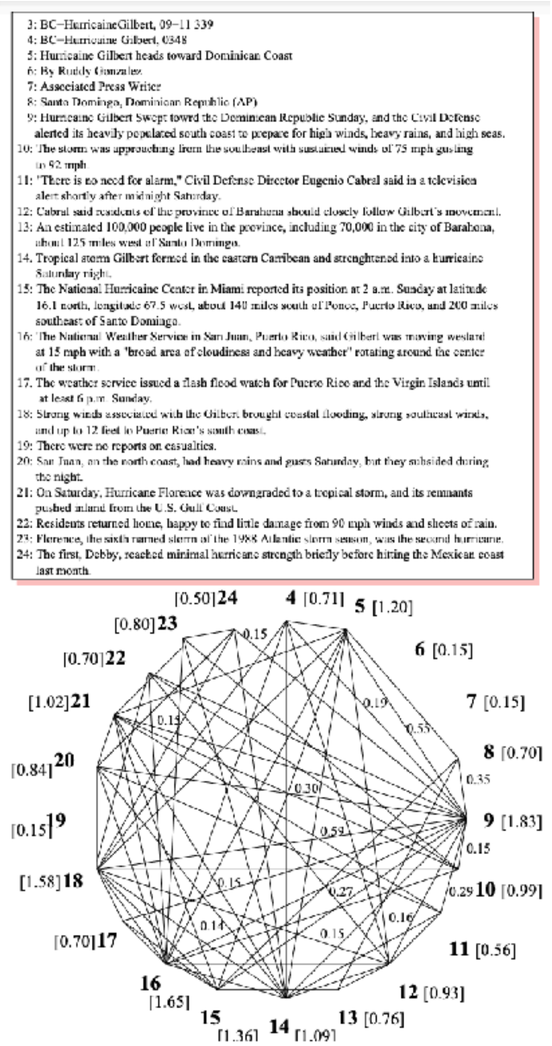
Rousseau et al., ACL 2015



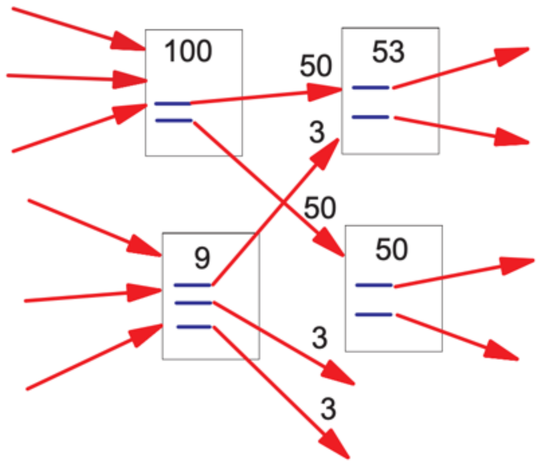
Rousseau et al., CIKM 2013

Node: words, keywords
Edge: word co-occurrence, directed/undirected

Text Graph

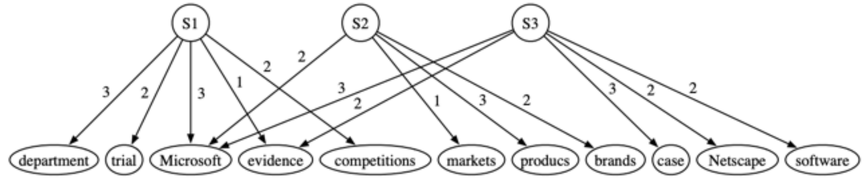


TextRank



PageRank

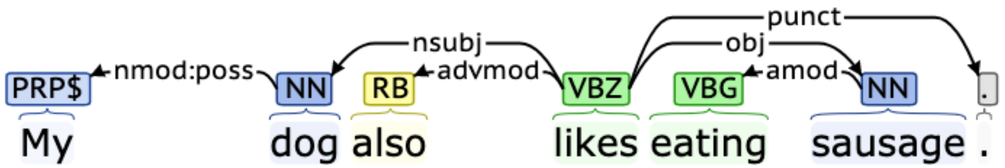
$_{S1}$[(The Justice Department)$_{S}$ is conducting an (anti-trust trial)$_{O}$ against (Microsoft Corp.)$_{X}$ with (evidence)$_{X}$ that (the company)$_{S}$ is increasingly attempting to crush (competitors)$_{O}$.] $_{S2}$[(Microsoft)$_{O}$ is accused of trying to forcefully buy into (markets)$_{X}$ where (its own products)$_{S}$ are not competitive enough to unseat (established brands)$_{O}$.] $_{S3}$[(The case)$_{S}$ revolves around (evidence)$_{O}$ of (Microsoft)$_{S}$ aggressively pressuring (Netscape)$_{O}$ into merging (browser software)$_{O}$.]



Putra et al., ACL 2017

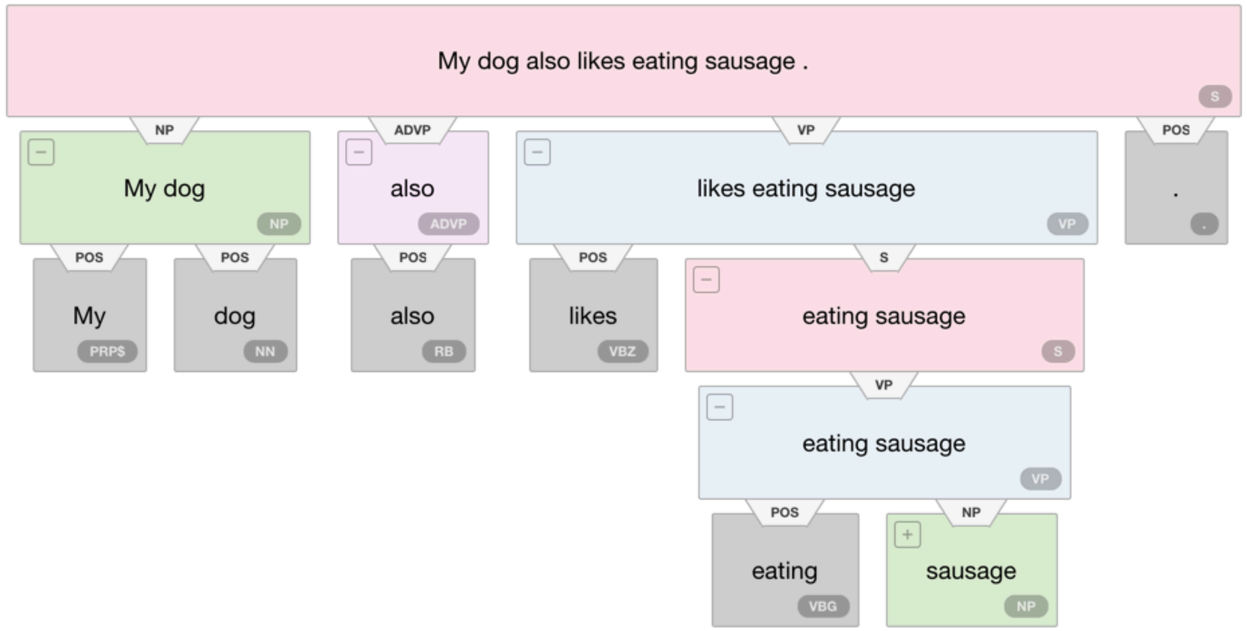
Node: sentences, paragraphs, documents
Edge: word co-occurrence, text similarities, location

Syntactic Graph



https://corenlp.run/

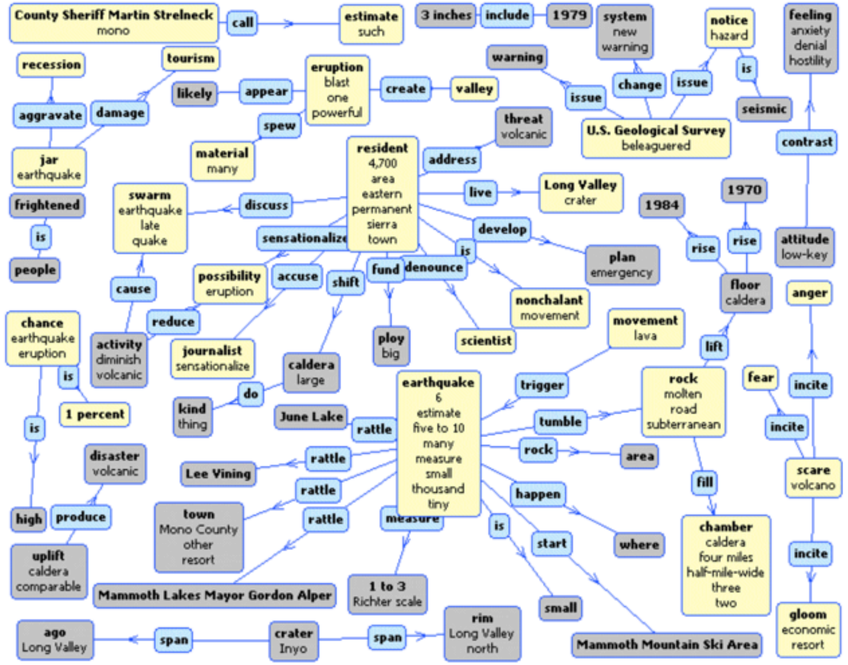
Dependency Parsing



https://demo.allennlp.org/constituency-parsing/constituency-parser

Constituency Parsing

Semantic Graph



Source: Leskovec et al., LinkKDD 2004

LOGIC format:

$\exists$ w, b, g:
instance(w, want-01) $\wedge$ instance(g, go-01) $\wedge$
instance(b, boy) $\wedge$ arg0(w, b) $\wedge$
arg1(w, g) $\wedge$ arg0(g, b)

AMR format (based on PENMAN):

```
(w / want-01
  :arg0 (b / boy)
  :arg1 (g / go-01
           :arg0 b))
```

GRAPH format:

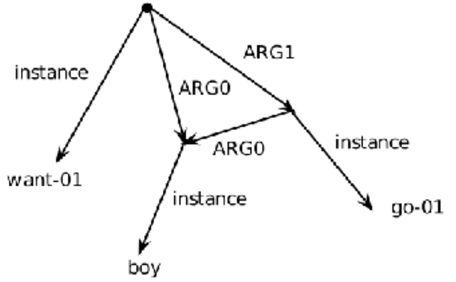
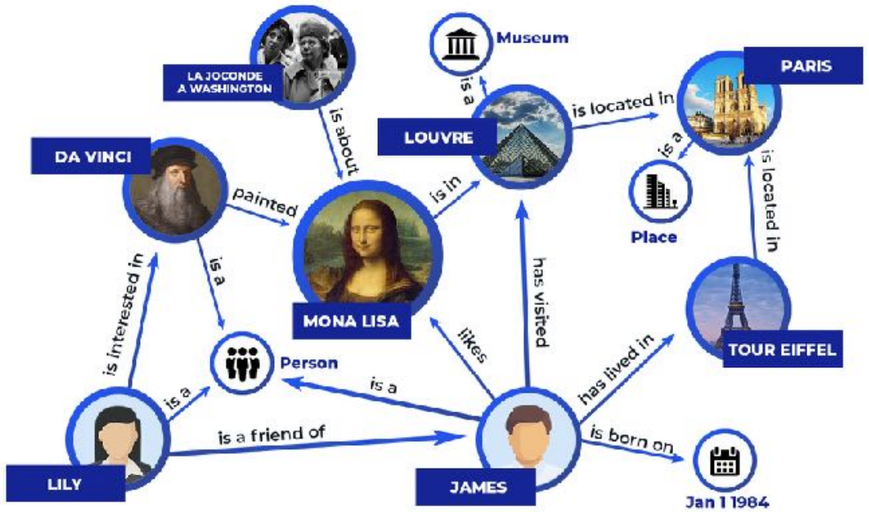


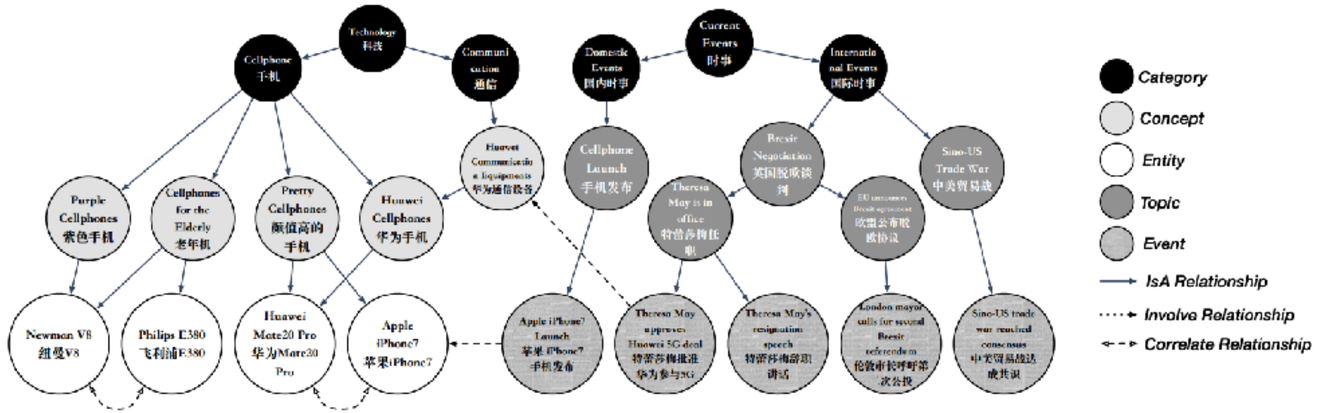
Figure 1: Equivalent formats for representating the meaning of "The boy wants to go".

Reveal the semantic structure between words or phrases

Abstract Meaning Representation for Sembanking

Knowledge Graph

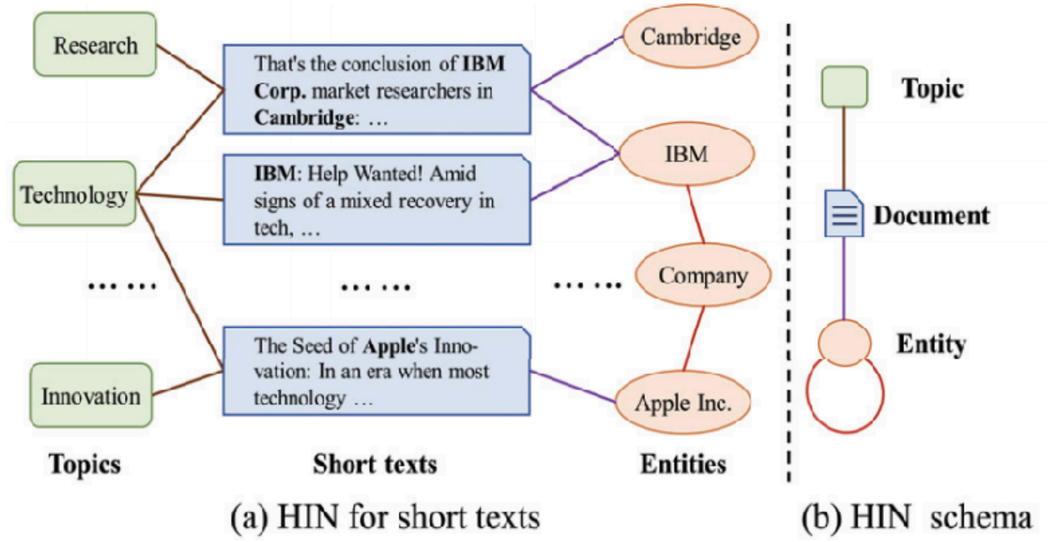




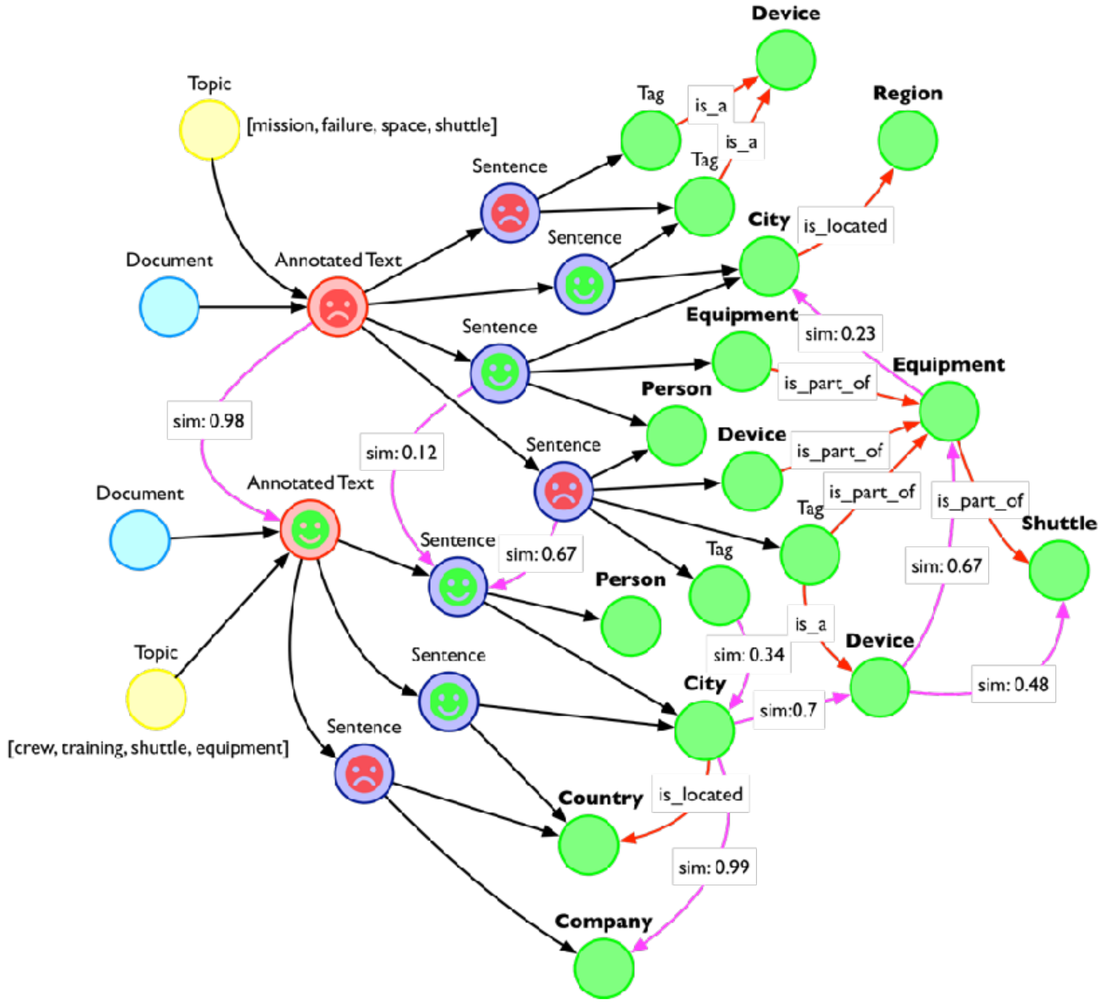
Knowledge Graph

Attention Ontology

Heterogeneous Graph



Hu et al., EMNLP 2019



https://graphaware.com/nlp/2018/09/26/bring-order-to-chaos.html

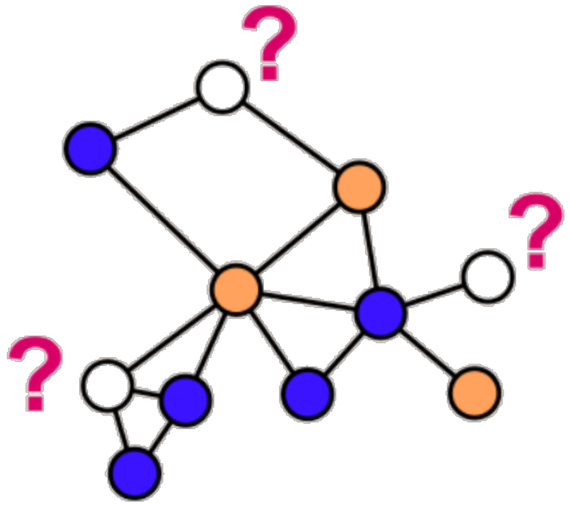
NLP from a Graph Perspective

How to represent text
by graphs

How to compute
via graph modeling

Node Classification

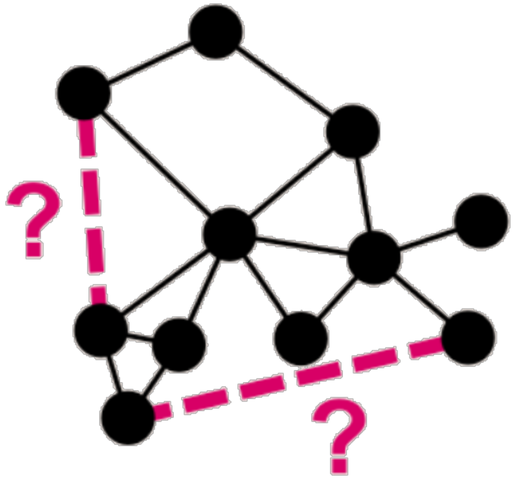
Assign labels to nodes



- Cetoli et al. TLT2016 - Graph Convolutional Networks for Named Entity Recognition
- Gui et al. EMNLP2019 - A Lexicon-Based Graph Neural Network for Chinese NER
- …

Link Prediction

Predict incomplete edges



- Rossi et al. TKDD2021 - Knowledge Graph Embedding for Link Prediction: A Comparative Analysis
- Broscheit et al. ACL 2020 - Can We Predict New Facts with Open Knowledge Graph Embeddings? A Benchmark for Open Link Prediction
- Zhang et al. NeurIPS 2018 - Link Prediction Based on Graph Neural Networks
- …

Relation Extraction

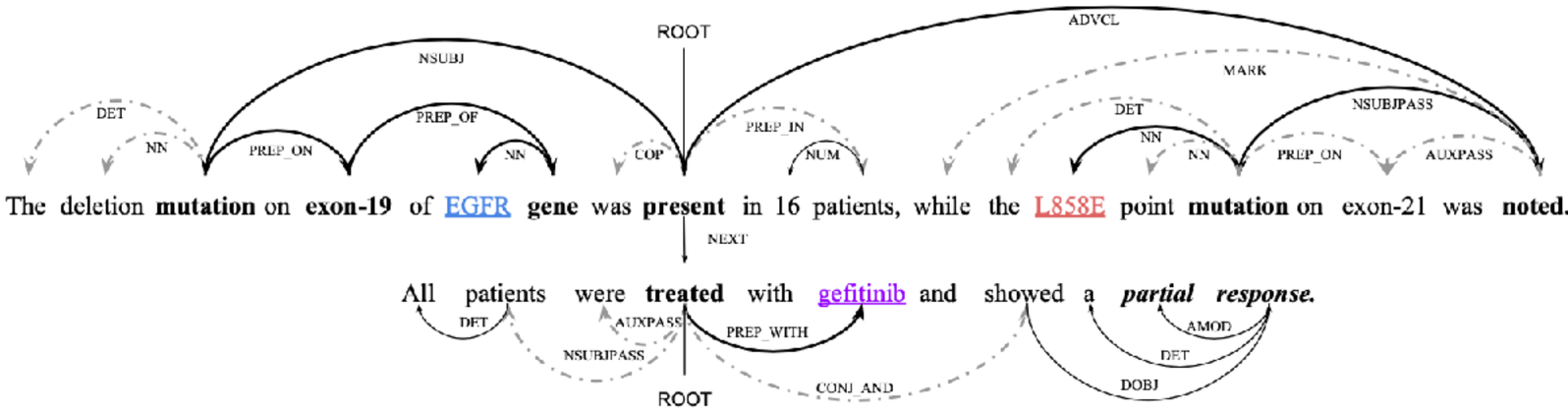
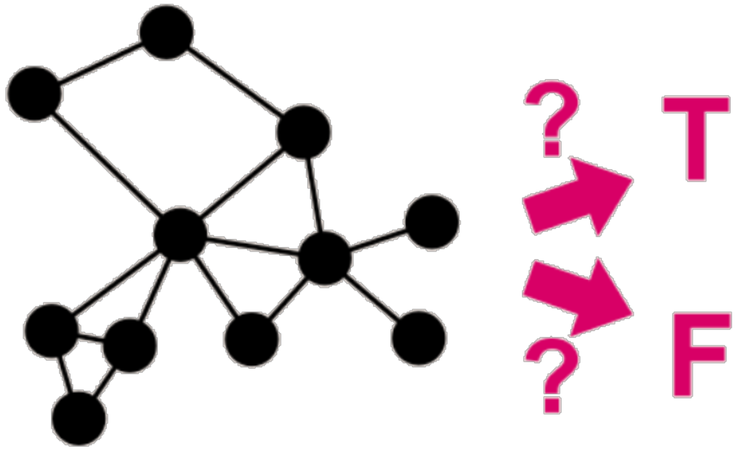


Figure 1: An example dependency tree for two sentences expressing a relation (sensitivity) among three entities. The shortest dependency path between these entities is highlighted in bold (edges and tokens). The root node of the LCA subtree of entities is *present*. The dotted edges indicate tokens K=1 away from the subtree. Note that tokens *partial response* off these paths (shortest dependency path, LCA subtree, pruned tree when K=1).

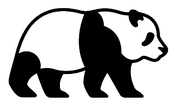
- Guo et al. ACL2019 - Attention Guided Graph Convolutional Networks for Relation Extraction
- Zhao et al. ACML2019 - Improving Relation Classification by Entity Pair Graph
- Zhang et al. EMNLP2018 - Graph Convolution over Pruned Dependency Trees Improves Relation Extraction
- Zhu et al. ACL2019 - Graph Neural Networks with Generated Parameters for Relation Extraction
- Marcheggiani et al. EMNLP2017 - Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling
- …

Graph Classification

Assign a label/score to the whole graph

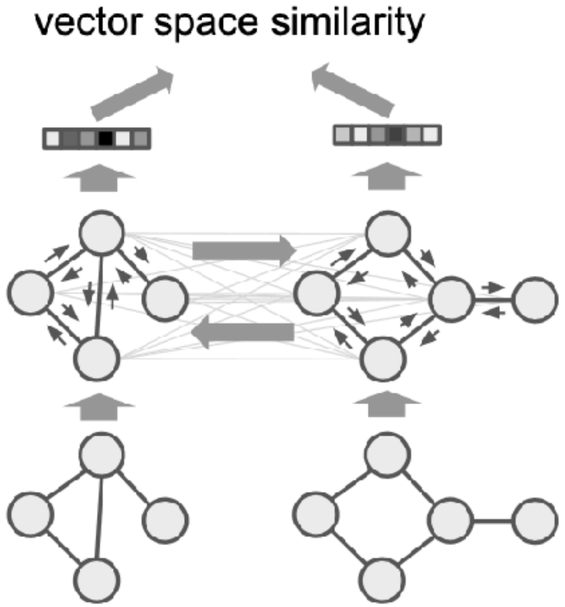


- Zhang et al. ACL2020 - Every Document Owns Its Structure: Inductive Text Classification via Graph Neural Networks
- Yao et al. AAAI2019 - Graph Convolutional Networks for Text Classification
- Huang et al. EMNLP2019 - Text Level Graph Neural Network for Text Classification
- Peng et al. WWW2018 - Large-Scale Hierarchical Text Classification with Recursively Regularized Deep Graph-CNN
- …



Graph Matching

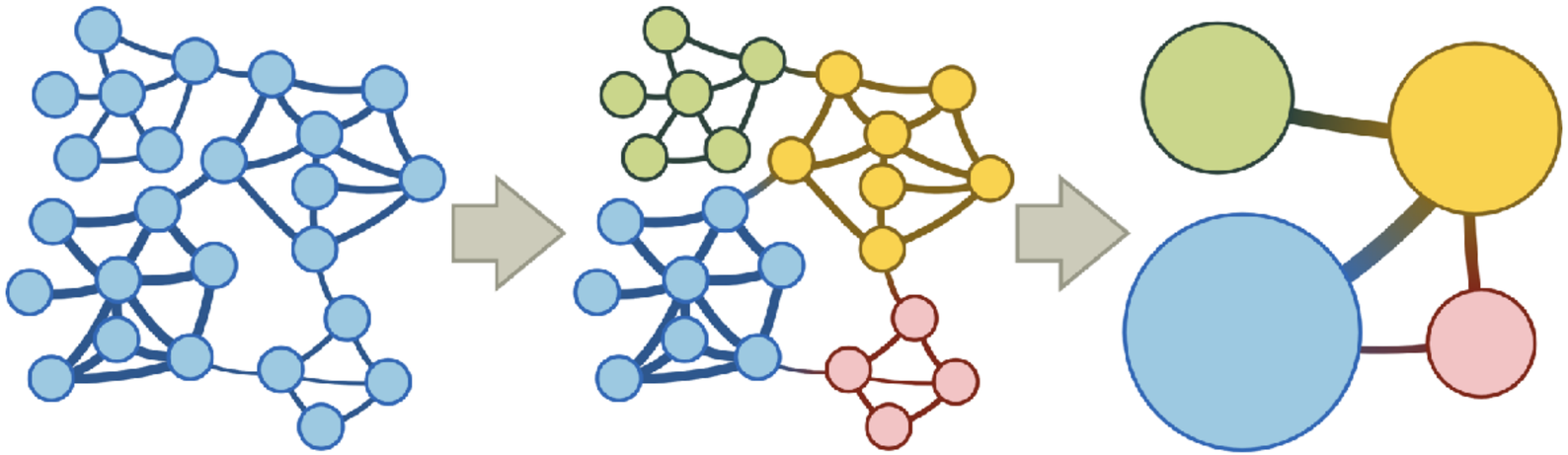
Assign a label/score to a pair of graphs



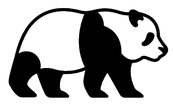
- Li et al. PMLR2019 - Graph Matching Networks for Learning the Similarity of Graph Structured Objects
- Liu et al. ACL2019 - Matching Article Pairs with Graphical Decomposition and Convolutions
- …

Community Detection

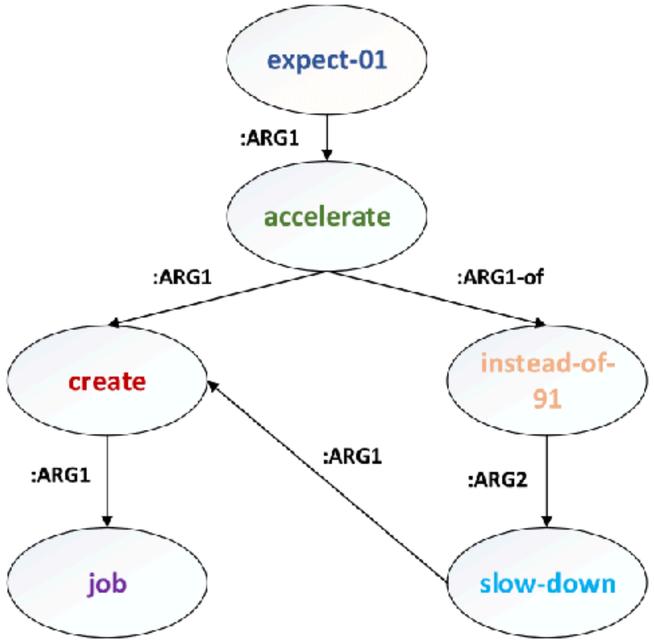
Group similar nodes



- Liu et al. TKDD2020 - Story Forest: Extracting Events and Telling Stories from Breaking News
- Chen et al. ICLR 2019 - Supervised Community Detection with Line Graph Neural Networks
- …



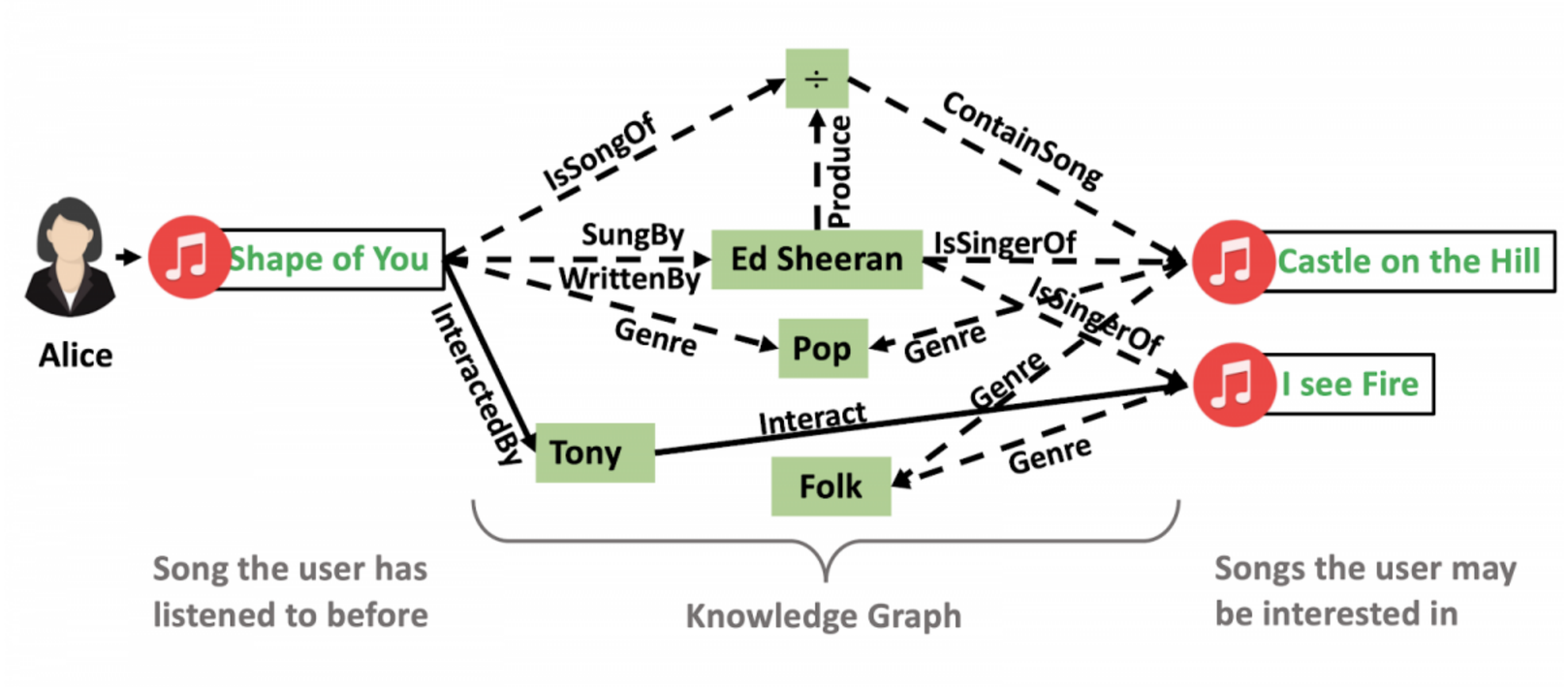
Graph to Text Generation



Job creation is not expected to slow down but will instead accelerate.

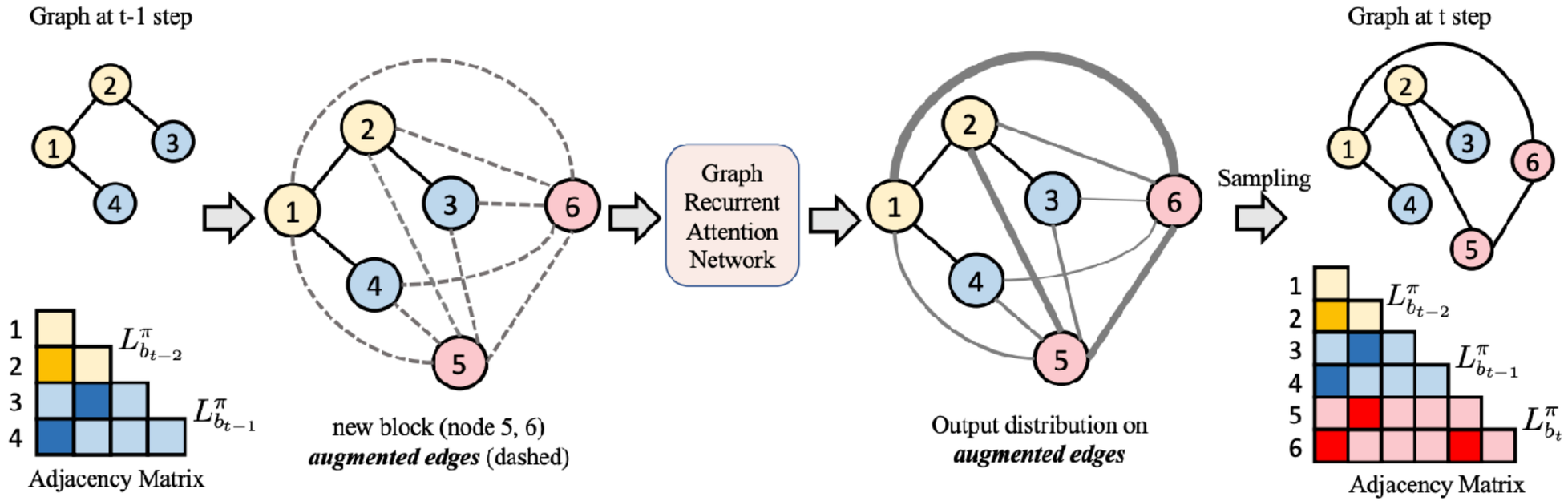
- Koncel-Kedziorski et al. NAACL2019 - Text Generation from Knowledge Graphs with Graph Transformers
- Wang et al. TACL2020 - AMR-To-Text Generation with Graph Transformer
- Song et al. ACL2018 - A Graph-to-Sequence Model for AMR-to-Text Generation
- Alon et al. ICLR2019 - Code2Seq: Generating Sequences from Structured Representations of Code
- …

Reasoning over Graphs



- Wang et al. AAAI2019 - Explainable Reasoning over Knowledge Graphs for Recommendation
- Ding et al. ACL2019 - Cognitive Graph for Multi-Hop Reading Comprehension at Scale
- Xiong et al. ACL2017 - DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning
- Chen et al. IJCAI2020 - GraphFlow: Exploiting Conversation Flow with Graph Neural Networks for Conversational Machine Comprehension
- Tu et al. ACL2019 - Multi-hop Reading Comprehension across Multiple Documents by Reasoning over Heterogeneous Graphs
- Asai et al. ICLR2020 - Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering
- Fang et al. EMNLP2020 - Hierarchical Graph Network for Multi-hop Question Answering
- …

Graph Transformation/Generation

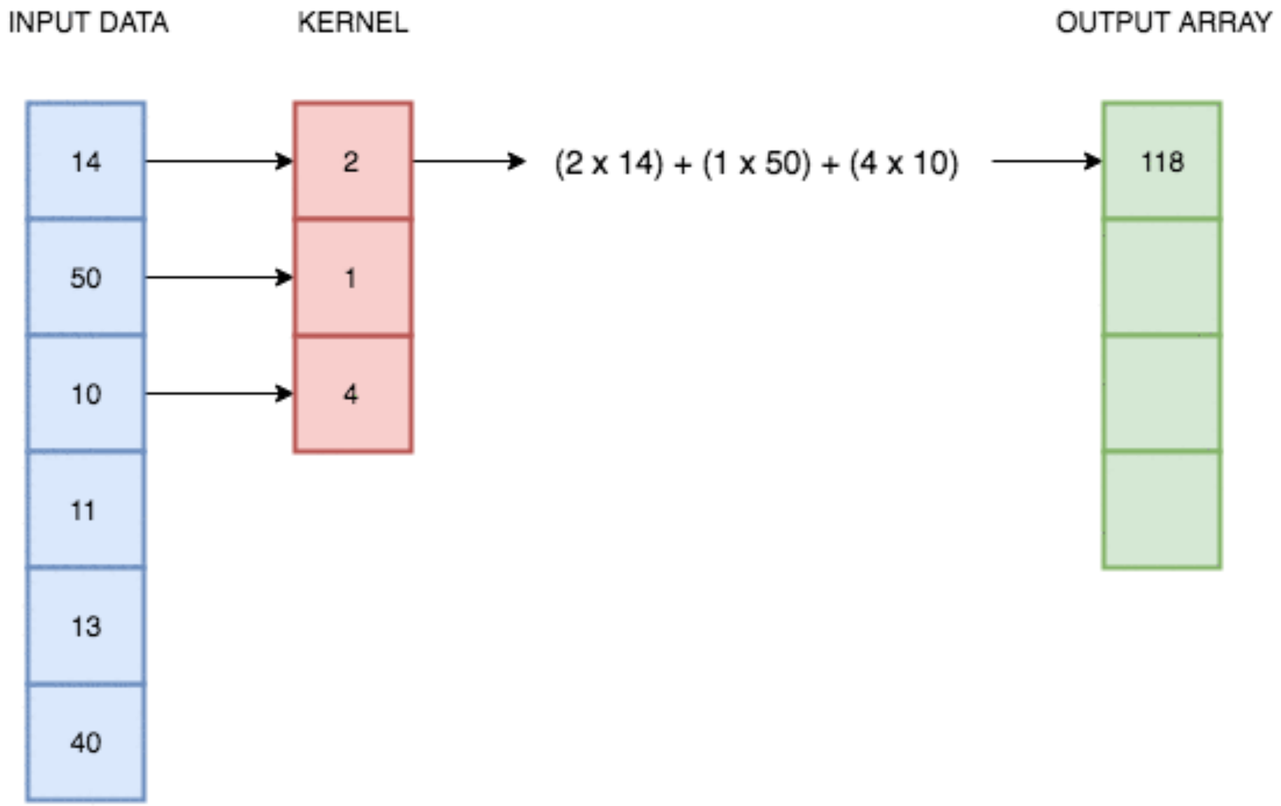


- Liao et al. NeurIPS2019 - Efficient Graph Generation with Graph Recurrent Attention Networks
- Do et al. KDD2019 - Graph Transformation Policy Network for Chemical Reaction Prediction
- Dinella et al. ICLR2020 - Hoppity: Learning Graph Transformations to Detect and Fix Bugs in Programs
- Brockschmidt et al. ICLR 2019 - Generative Code Modeling with Graphs
- …

Graph Neural Networks

First, let us recap
What is Convolution?

Understand Convolution: 1D



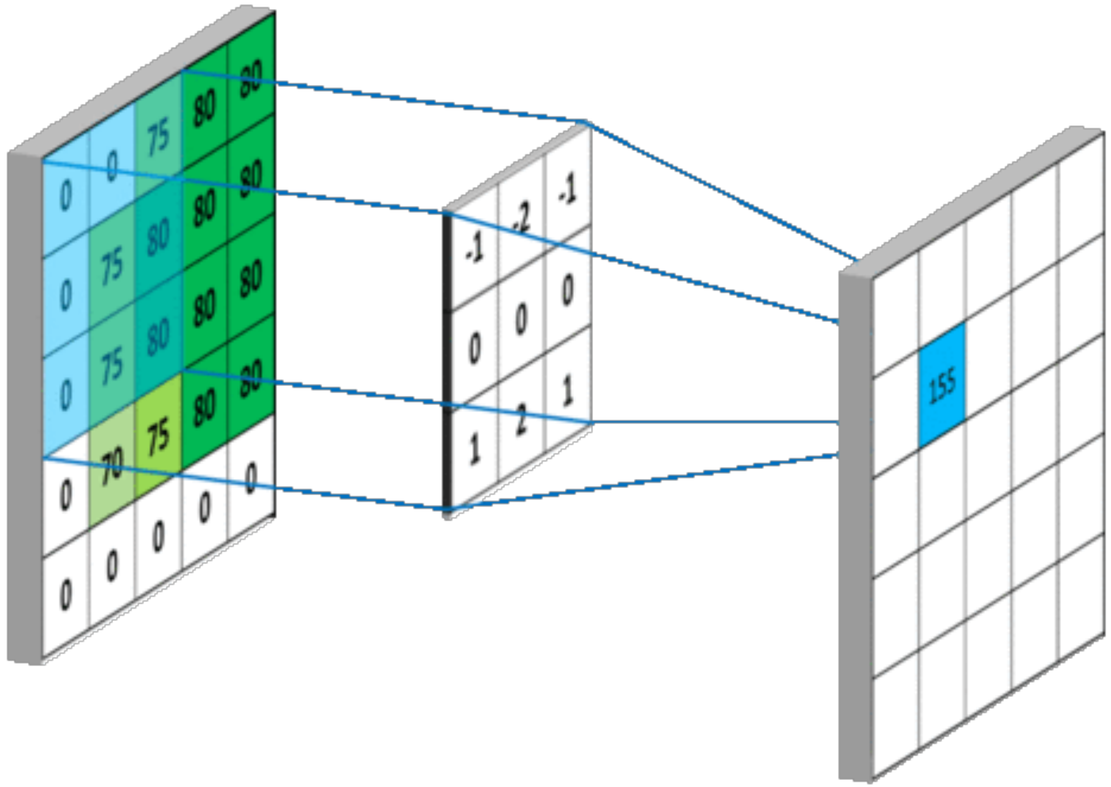
Continuous

$$(f * g)(n) = \int_{-\infty}^{\infty} f(\tau)g(n-\tau)d\tau$$

Discrete

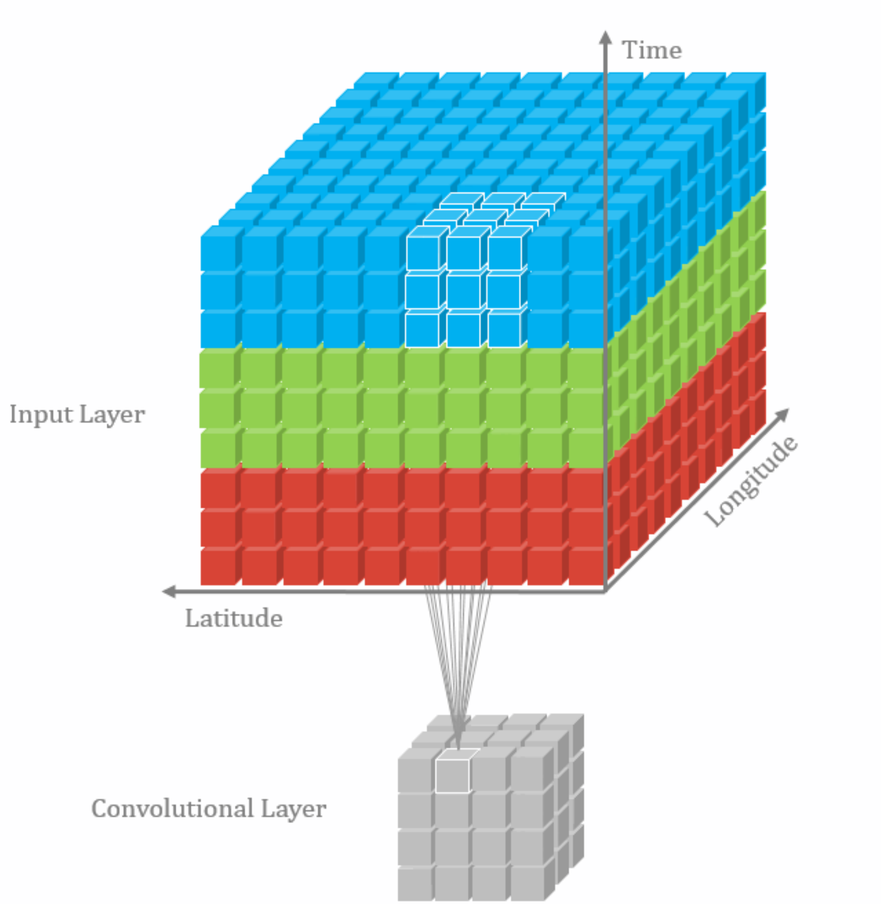
$$(f * g)(n) = \sum_{\tau=-\infty}^{\infty} f(\tau)g(n-\tau)$$

Understand Convolution: 2D



$$(f * g)(u, v) = \sum_i \sum_j f(i, j) g(u - i, v - j) = \sum_i \sum_j a_{i,j} b_{u-i,v-j}$$

Understand Convolution: 3D



$$(f * g)(u, v, w) = \sum_i \sum_j \sum_k f(i, j, k) g(u - i, v - j, w - k)$$

Understand Convolution: M-D

- Convolution is a **weighted sum with constraints**
- Convolution is a **filter / feature extractor**
- Convolution is a **transformation**

$$y(n_1, n_2, \ldots, n_M) = x(n_1, n_2, \ldots, n_M) * \overset{M}{\cdots} * h(n_1, n_2, \ldots, n_M)$$

$$\sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} \cdots \sum_{k_M=-\infty}^{\infty} h(k_1, k_2, \ldots, k_M) x(n_1 - k_1, n_2 - k_2, \ldots, n_M - k_M)$$

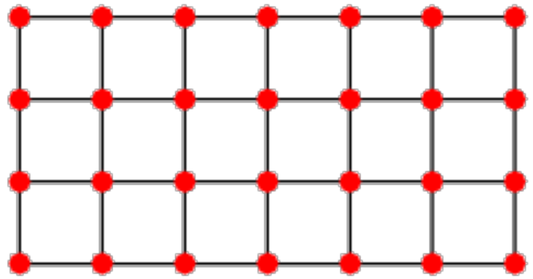
Graph is different

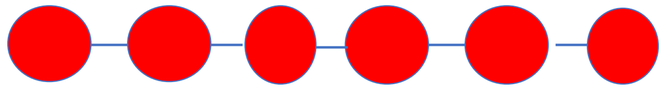
How to Define Graph Convolution?

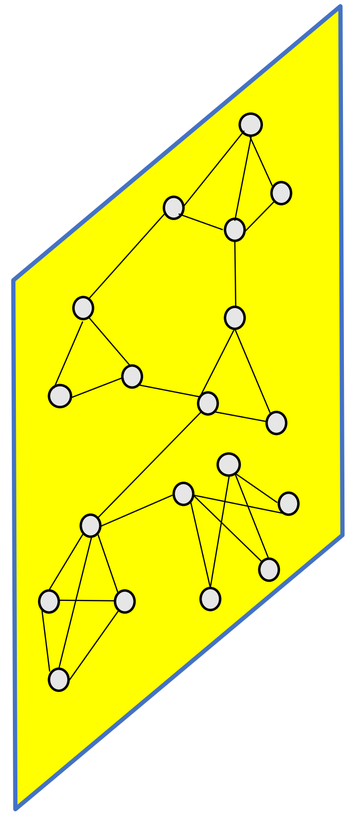
From Euclidean to Graphs

Traditional DL is designed for simple grids or sequences

- CNNs for fixed-size images/grids
- RNNs for text/sequences







But nodes on graphs have different connections

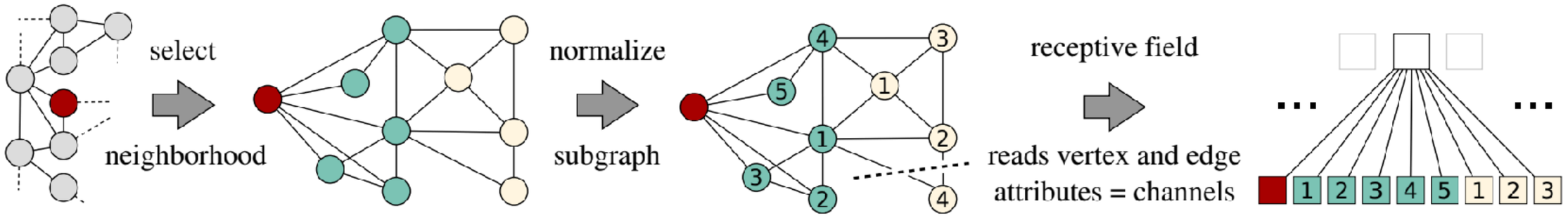
- Arbitrary neighbor size
- Complex topological structure
- No fixed node ordering

Why We Need Graph Neural Networks

- **CNN cannot process Non Euclidean Structure data**, and traditional discrete convolutions **cannot maintain translation invariance** on Non Euclidean Structure data (the number of adjacent vertices in each vertex in the topological graph may be different, so of course it is impossible to use the same size Convolution kernel for convolution operation)
- Since CNN cannot process Non Euclidean Structure data, and **hopes to effectively extract spatial features from such a data structure (topological map)** for machine learning, GCN has become the focus of research
- There is no topology network in my own research question, so is GNN not used at all? In fact, it is not. Broadly speaking, **any data can establish topological associations in the normed space.** Spectral clustering is the application of this idea. **So topological connection is a generalized data structure, and GNNs has a lot of application space.**

Two ways to extract features of topological graph

- **Spatial domain (vertex domain):** find out the neighbours of each vertex
  - How to define neighbours (or how to identify receptive field)?
  - How to process different numbers of neighbours?
- **Main limitations:**
  - Each vertex has different neighbors, so that the calculation must be done for each vertex
  - The effect of extracting features may not be as good as convolution



Niepert et al., Learning Convolutional Neural Networks for Graphs

Two ways to extract features of topological graph

- **How to define convolution on graphs?**
  - Different nodes have different number of neighbours
  - Nodes are unordered
- **Spectral domain**
  - We can define “convolution” operation based on **Convolution Theorem**
    - Convolution in one domain (e.g., time domain) equals point-wise multiplication in the other domain (e.g., frequency domain)

Convolution theorem

$$h(x) = f * g = \mathcal{F}^{-1}\{F \cdot G\},$$

Research History of Graph Convolution

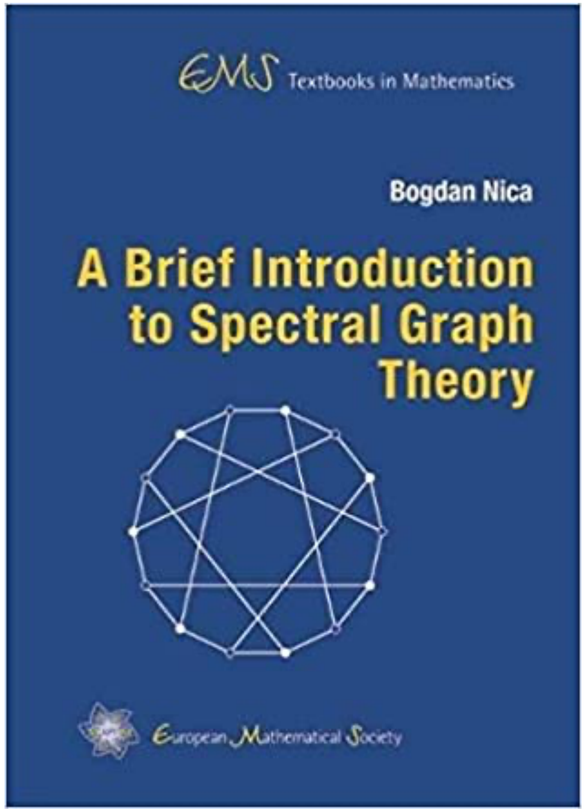
- The scholar who first studied **Graph Signal Processing (GSP)** defined **Fourier Transformation on graph**
- They then defines the **convolution on the graph**
- Finally, in combination with deep learning, **Graph Convolutional Network** was proposed

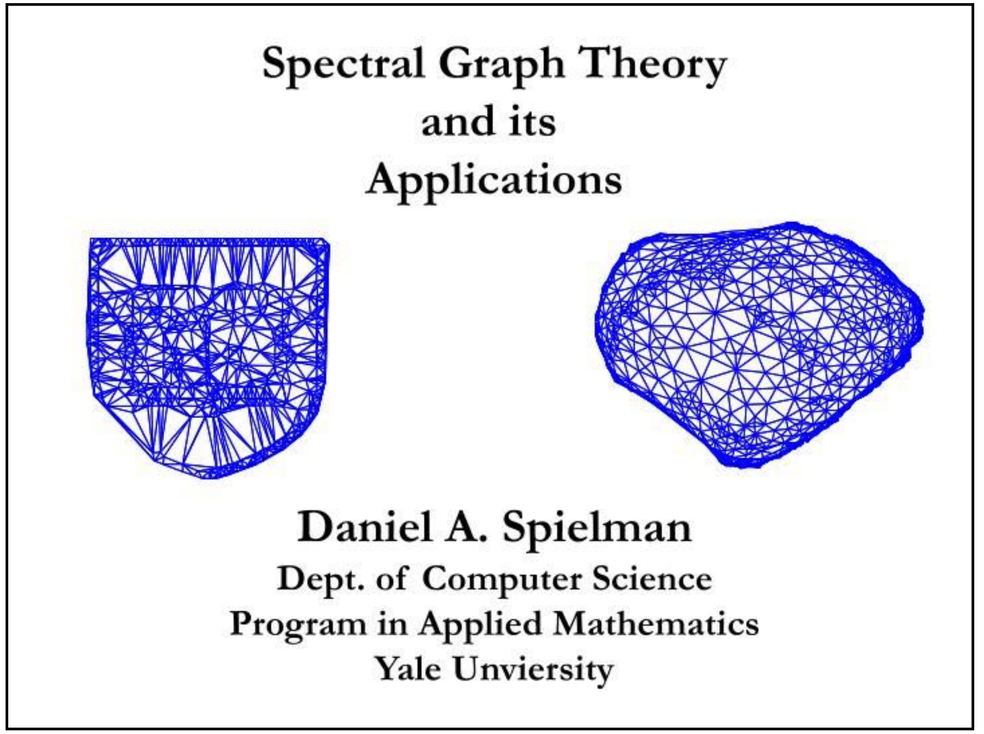
Q: How to represent graphs in Spectral Domain?

A: Spectral Graph Theory

Overview of Spectral Graph Theory

Study the properties of a graph based on the eigenvalues and eigenvectors of the Laplacian matrix of the graph.





Overview of GNNs

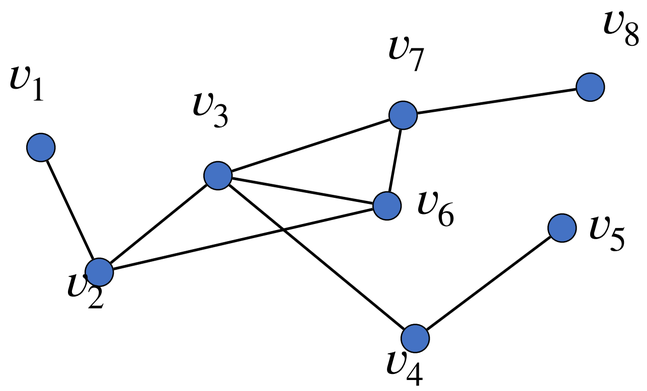
- **Graph Neural Networks**
  - The core work of GNNs is to perform convolution operations on the Embedding of nodes in the Spatial Domain (i.e., aggregate neighbor Embedding information)
  - However, the difference between graph data and image data is that the number and order of node neighbors are indefinite, so the traditional convolution operation (Convolution Operator) in the CNN model used on images cannot be directly used on the graph.

Overview of GNNs

- **Spectral domain**
  - We can **define "convolution" operation** based on **Convolution Theorem**
    - Convolution in one domain (e.g., time domain) equals point-wise multiplication in the other domain (e.g., frequency domain)
  - In order **to transform between the spectral domain and the spatial domain**, we use the Fourier formula and **define the Fourier transform on the graph** (from the spatial domain to the spectral domain) **and the inverse Fourier transform** on the graph (from the spectral domain back to Spatial domain) transformation formula
  - The specific operation is that we transform the Embedding of the node from the spatial domain to the spectral domain through forward Fourier transform, perform convolution operation with the convolution kernel in the spectral domain, and then transform the transformed node Embedding back into the inverse Fourier transform. Go to the spatial domain and participate in subsequent tasks such as classification.

Now, let’s learn some math:
Graph Signal Processing

Graphs and Graph Signals

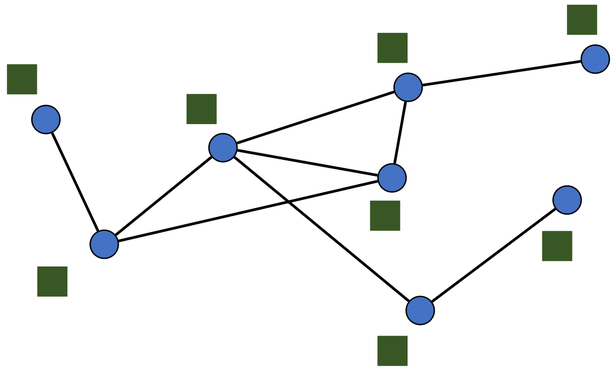


$$\mathcal{V} = \{v_1, \ldots, v_N\}$$

$$\mathcal{E} = \{e_1, \ldots, e_M\}$$

$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$$

Graphs and Graph Signals



$$\mathcal{V} = \{v_1, \ldots, v_N\}$$

$$\mathcal{E} = \{e_1, \ldots, e_M\}$$

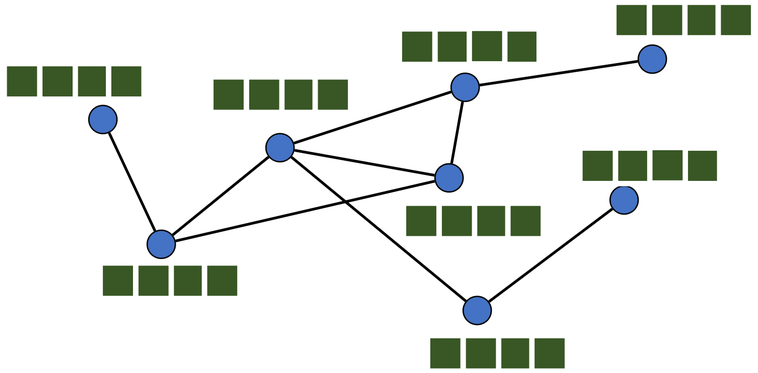
$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$$

Graph signal: a function map each vertex to a scalar

Graph Signal: $f : \mathcal{V} \rightarrow \mathbb{R}^N$

$$\mathcal{V} \longrightarrow \begin{bmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \\ f(8) \end{bmatrix}$$

Graphs and Graph Signals



$$\mathcal{V} = \{v_1, \ldots, v_N\}$$

$$\mathcal{E} = \{e_1, \ldots, e_M\}$$

$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$$

Can also make each vertex to a vector of d dimensional (node feature vector)

Graph Signal: $f : \mathcal{V} \rightarrow \mathbb{R}^{N \times d}$

$$\mathcal{V} \rightarrow \begin{bmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \\ f(8) \end{bmatrix}$$

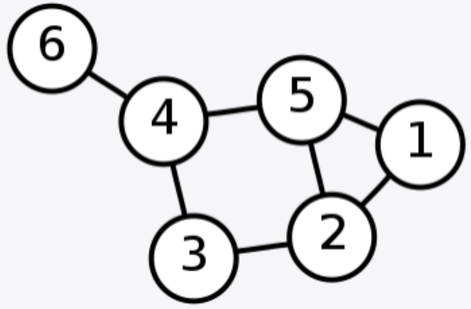
Interprete Laplacian Matrix

Laplacian Matrix

- Laplacian matrix is a matrix representation of a graph.
- It can be used to find many useful properties of a graph: calculate the number of spanning trees, construct low dimensional embeddings, etc.
- It can be interpreted as a matrix representation of a particular case of the discrete Laplace operator.

Degree Matrix Adjacency Matrix

G D A $L = D - A$

Labelled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

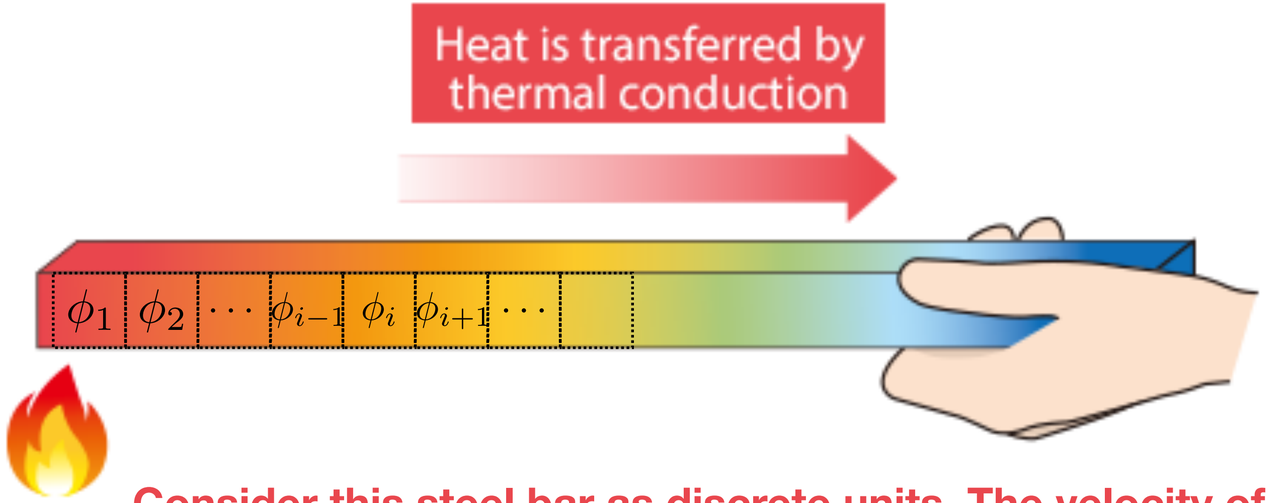
$\mathbf{D} = \mathrm{diag}(degree(v_1), \ldots, degree(v_N))$

$A[i,j] = 1$ if v_i is adjacent to v_j

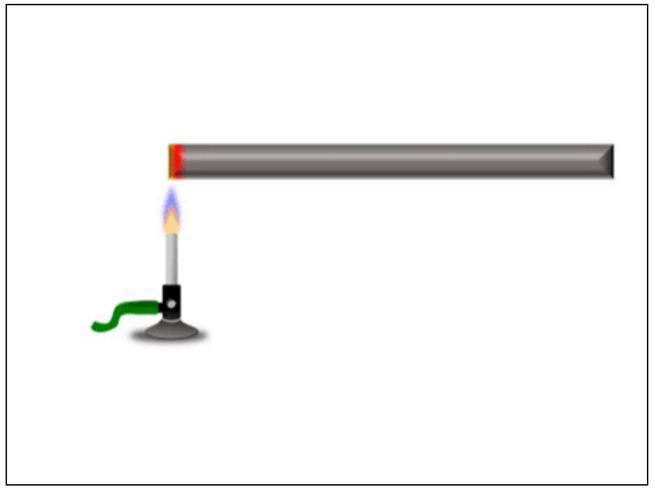
$A[i,j] = 0$, otherwise

https://en.wikipedia.org/wiki/Laplacian_matrix

Thermal Conduction



Consider this steel bar as discrete units. The velocity of heat transfer is proportional to the second derivative



This defines the 1-dimensional laplacian:

$$\frac{d\phi_i}{dt} = k(\phi_{i+1} - \phi_i) - k(\phi_i - \phi_{i-1})$$

$$\downarrow$$

$$\frac{d\phi_i}{dt} - k[(\phi_{i+1} - \phi_i) - (\phi_i - \phi_{i-1})] = 0$$

$$\downarrow$$

$$\frac{\partial \phi}{\partial t} - k\frac{\partial^2 \phi}{\partial x^2} = 0$$

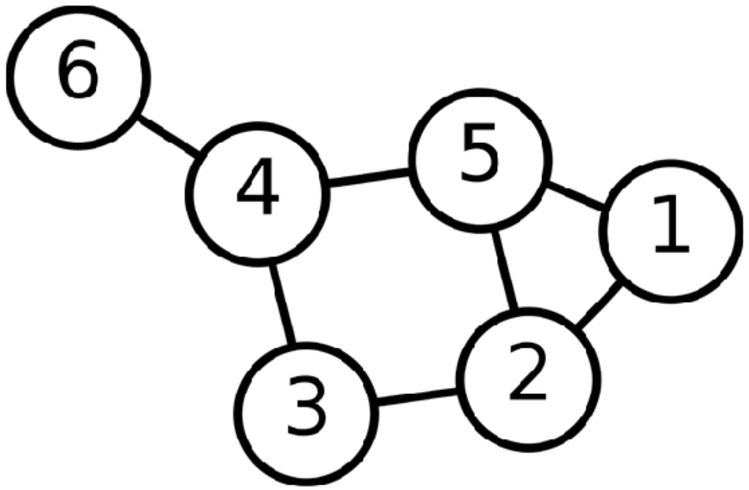
$$\downarrow$$

Heat equation: $$\frac{\partial \phi}{\partial t} - k\Delta\phi = 0$$

k: diffusivity constant

$\triangle$: Laplacian operator.
The sum of the second derivative of each coordinate

Thermal Conduction over Graphs



$$\frac{d\phi_i}{dt} = -k\sum_j A_{ij}(\phi_i - \phi_j)$$

$$\frac{d\phi_i}{dt} = -k[\phi_i \sum_j A_{ij} - \sum_j A_{ij}\phi_j]$$

$$= -k[deg(i)\phi_i - \sum_j A_{ij}\phi_j]$$

$$\begin{bmatrix} \frac{d\phi_1}{dt} \\ \frac{d\phi_2}{dt} \\ \cdots \\ \frac{d\phi_n}{dt} \end{bmatrix} = -k \begin{bmatrix} deg(1) \times \phi_1 \\ deg(2) \times \phi_2 \\ \cdots \\ deg(n) \times \phi_n \end{bmatrix} + kA \begin{bmatrix} \phi_1 \\ \phi_2 \\ \cdots \\ \phi_n \end{bmatrix}$$

$$\boldsymbol{\phi} = [\phi_1, \phi_2, \ldots, \phi_n]^T$$

$$D = diag(deg(1), deg(2), \ldots, deg(n))$$

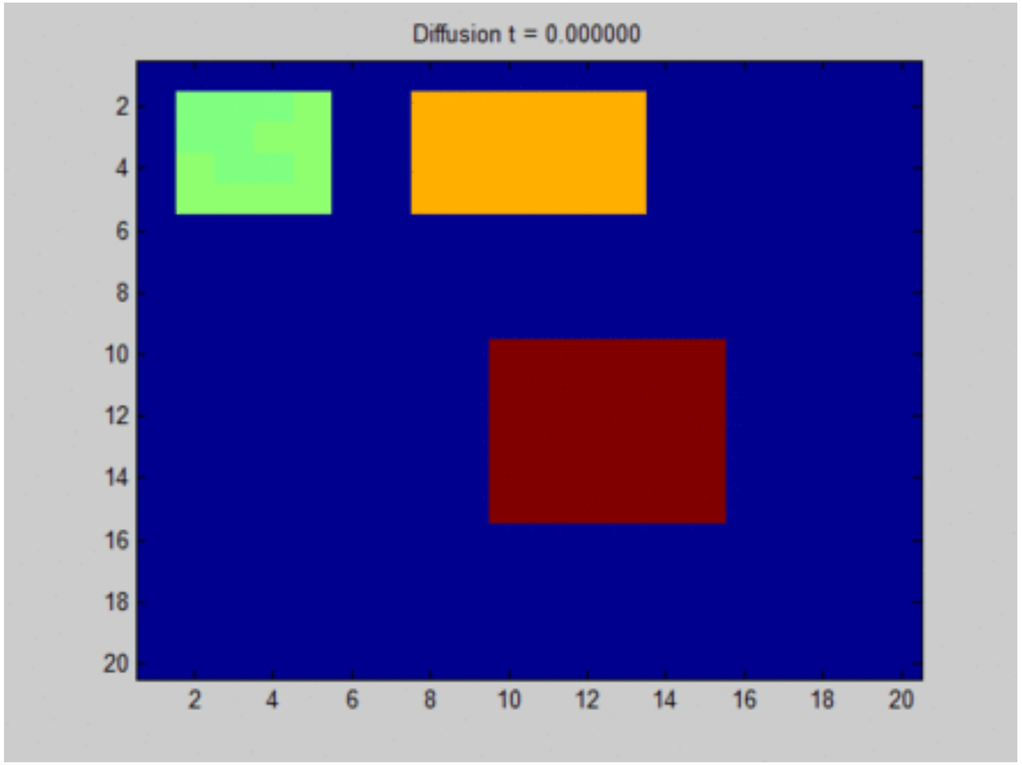
$$\frac{d\boldsymbol{\phi}}{dt} = -kD\boldsymbol{\phi} + kA\boldsymbol{\phi} = -k(D - A)\boldsymbol{\phi}$$

$$\frac{d\boldsymbol{\phi}}{dt} + kL\boldsymbol{\phi} = 0$$

$$\frac{\partial \phi}{\partial t} - k\Delta\phi = 0$$

Laplacian Matrix acts as discrete **Laplace operator over a graph**.

Thermal Conduction over Graphs



This GIF shows the progression of **diffusion**, as solved by the **graph laplacian** technique. A graph is constructed over a grid, where each pixel in the graph is connected to its 8 bordering pixels. **Values in the image then diffuse smoothly to their neighbors over time via these connections.** This particular image starts off with three strong point values which spill over to their neighbors slowly. The whole system eventually settles out to the same value at equilibrium.

What We Want: Message Passing over Graphs

- What we have seen?
  - We have a **space**: 1D, 2D, 2D discrete, graph, ….
  - Something can **transfer in the space**: heat, …
  - The **intensity of the transfer** between two adjacent points is proportional to the state difference between them.

- What we want now?
  - Space: a **graph**
  - Things to transfer: **feature** / **message**
  - Rule of transfer: the state change is proportional to the corresponding space (here is the Graph space) Laplacian operator acts on the current state.

- How to model it?
  - No need to follow Newton's law of cooling
  - E.g., you can use Neural Networks, kernal functions, etc., to calculate how things propagate based on adjacent node's states

Laplacian Matrix as an Operator

Laplacian matrix is a difference operator:

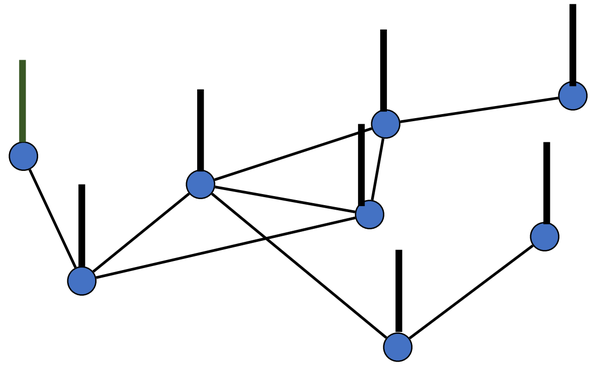
$$\mathbf{h} = \mathbf{L}\mathbf{f} = (\mathbf{D} - \mathbf{A})\mathbf{f} = \mathbf{D}\mathbf{f} - \mathbf{A}\mathbf{f}$$

$$\mathbf{h}(i) = \sum_{v_j \in \mathcal{N}(v_i)} (\mathbf{f}(i) - \mathbf{f}(j))$$

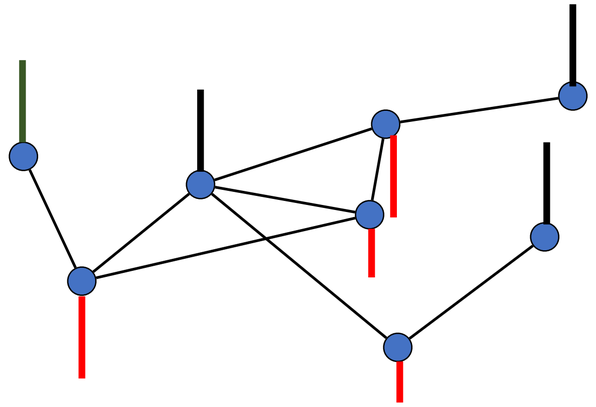
Laplacian quadratic form:

$$\mathbf{f}^T \mathbf{L}\mathbf{f} = \frac{1}{2} \sum_{i,j=1}^{N} \mathbf{A}[i,j](\mathbf{f}(i) - \mathbf{f}(j))^2$$

"Smoothness" or "Frequency" of the signal f



Low frequency graph signal



High frequency graph signal

Laplacian Matrix as an Operator

Laplacian matrix is positive semi-definite:

$$
\begin{aligned}
f^T L f &= f^T (D - A) f \\
&= f^T D f - f^T A f \\
&= f^T \ diag(d) \ f - f^T A f \\
&= \sum_{i=1}^{m} d_i f_i^2 - \sum_{j=1}^{m} \left[\sum_{i=1}^{m} f_i a_{ij} \right] f_j \\
&= \sum_{i=1}^{m} d_i f_i^2 - \sum_{i,j=1}^{m} f_i f_j a_{ij} \\
&= \frac{1}{2} \left[\sum_{i=1}^{m} d_i f_i^2 - 2 \sum_{i,j=1}^{m} f_i f_j a_{ij} + \sum_{i=1}^{m} d_i f_i^2 \right] \\
&= \frac{1}{2} \sum_{i,j=1}^{m} a_{ij} \left(f_i - f_j \right)^2 \ \text{note: } d_i = \sum_{j=1}^{m} a_{ij}
\end{aligned}
$$

$$
f^T L f \geqslant 0
$$

Important Properties of Laplacian Matrix

1. It is a real symmetric matrix with n linearly independent eigenvectors
2. The eigenvectors of the matrix are orthogonal to each other, that is, the matrix composed of all eigenvectors is an orthogonal matrix
3. All eigenvalues of positive semi-definite matrix are non-negative

Some Other Properties of Laplacian Matrix

- The number of occurrences of 0 in the eigenvalue is the number of connected regions in the graph.
- The minimum eigenvalue is 0, because in the Laplacian matrix (common form: L=D−A), the sum of each row is 0, and the eigenvector corresponding to the smallest eigenvalue is a vector whose value is all 1.
- The smallest non-zero eigenvalue is called the algebraic connectivity of the graph.

Eigen-decomposition of Laplacian Matrix

1. It is a real symmetric matrix with n linearly independent eigenvectors, so **Laplacian matrix must be able to perform eigen-decomposition (spectral decomposition)**

$$\mathbf{L} = \underbrace{\begin{bmatrix} | & & | \\ \mathbf{u}_0 & \cdots & \mathbf{u}_{N-1} \\ | & & | \end{bmatrix}}_{\boldsymbol{U}} \underbrace{\begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix}}_{\boldsymbol{\Lambda}} \underbrace{\begin{bmatrix} — & \mathbf{u}_0 & — \\ & \vdots & \\ — & \mathbf{u}_{N-1} & — \end{bmatrix}}_{\boldsymbol{U^T}}$$

Eigenvalues are sorted non-decreasingly: $0 = \lambda_0 < \lambda_1 \leq \cdots \lambda_{N-1}$

where $U = (u_0, \cdots, u_{N-1})$ is the **unit eigenvector** matrix composed of column vector u_i, λ_i are eigen values.

2. The eigenvectors of the matrix are orthogonal to each other, so **U is an orthogonal matrix**

$$UU^T = I \rightarrow U^T = U^{-1}$$

Fourier Transform on Graphs

Fourier Transform

The Fourier transform of the continuous domain is defined as:

$$F(w) = \mathcal{F}[f(t)] = \int f(t)e^{-iwt}dt$$

That is, the integral of the signal f(t) and the basis function e^{-iwt}.

Understand Fourier Transform: **3Blue1Brown:**

1. https://www.youtube.com/watch?v=spUNpyF58BY&ab_channel=3Blue1Brown
2. https://www.youtube.com/watch?v=r6sGWTCMz2k&t=83s&ab_channel=3Blue1Brown

So how do we **extend the Fourier transform** of the continuous domain **to the graph**? The most important thing is **how do we find the basis functions on the graph**? (The basis function of the continuous domain is known as e^{-iwt}.

Fourier Transform on Graphs

- The core work for **migrating the traditional Fourier transform and convolution to Graph**, is actually to **change the eigenfunction** e^{-iwt} **of the Laplacian operator into the eigenvector of the Laplacian matrix corresponding to Graph**.

- The relationship between the Fourier transform and the Laplacian matrix: **The base of the traditional Fourier transform is a set of eigenvectors of the Laplacian matrix.**

Fourier Transform on Graphs

- **We need a transformation**:
  - In the continuous domain, the basis function is e^{-iwt}.
  - When this transformation is placed on the graph, if there are eigenvectors and are orthogonal to each other, the eigenvectors can be used as basis vectors, so as the basis function of the discrete Fourier transform on the graph

Laplacian is the Transform Operator

- **The Laplacian Δ**, whether in the continuous domain or the discrete domain on the graph, just **satisfies these properties**
  - In the continuous domain:

$$\Delta e^{-iwt} = \frac{\partial^2}{\partial t^2} e^{-iwt} = -w^2 e^{-iwt}$$

  - Think of the definition of the generalized eigenvalue equation: $Au = \lambda u$
  - where A is a matrix (in linear algebra, the left multiplication of a matrix means linear transformation of a vector), u is an eigenvector or eigenfunction (vector of infinite dimensions), and λ is the eigenvalue corresponding to u. By analogy, we know that e^{-iwt} is the Eigenfunction of Δ, and ω is closely related to the corresponding eigenvalue.
- Therefore, if we want to transfer the Fourier transform to the graph, we have the Laplacian matrix (the Laplacian matrix is a discrete Laplacian operator), the next step is naturally to **find the eigenvectors of Laplacian matrix** (Equivalent to we have Δ, the next step is to find the corresponding e^{-iwt}).

To Summarize

- The core work of Fourier transform on graphs is to **map the eigenfunction e^{-iwt} of the Laplacian operator to the eigenvector of the graph Laplacian matrix.**
- Therefore, it is very important to **solve the eigenvector of the graph Laplacian matrix**.
- As introduced, the graph **Laplacian matrix is a positive semi-definite matrix**
- So the eigenvectors can be obtained through **eigen decomposition** (spectral decomposition), namely $u_1,...,u_n$.

	Traditional Fourier Transform	Graph Fourier Transform
Basises of Fourier Transform	$e^{-2\pi ixv}$	U^T
Basises of Inverse Fourier Transform	$e^{2\pi ixv}$	U
Dimension	∞	Number of vertices n

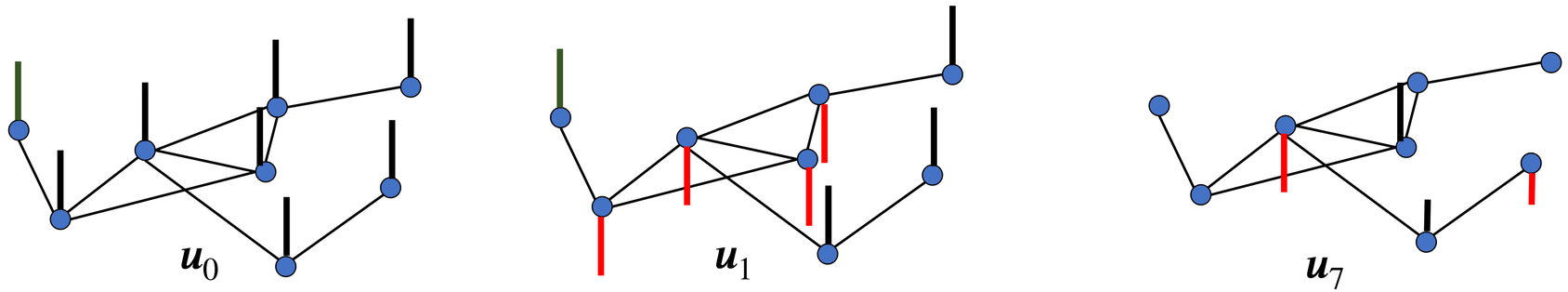
The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains

Eigenvectors as Graph Signals

The frequency of an eigenvector of Laplacian matrix is its corresponding eigenvalue:

$$\mathbf{u}_i^T \mathbf{L} \mathbf{u}_i = \mathbf{u}_i^T \lambda_i \mathbf{u}_i = \lambda_i$$

Frequency of the signal u_i



Low frequency

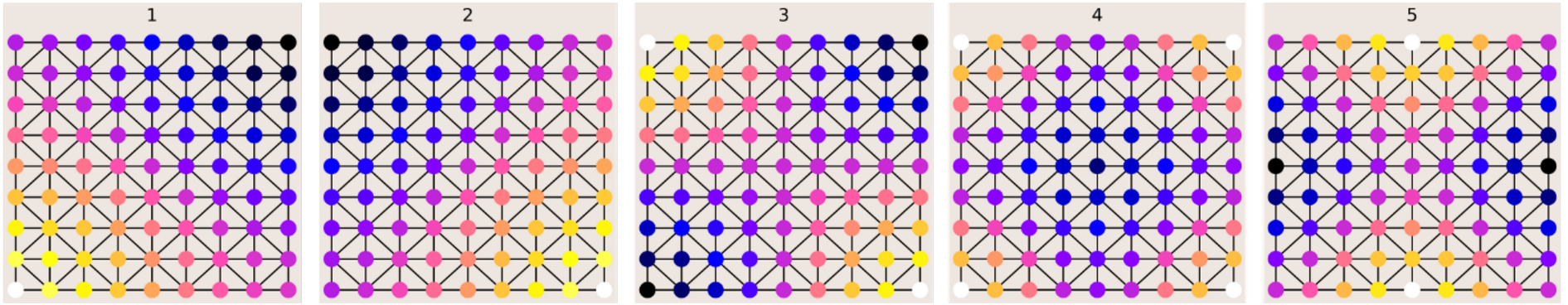
High frequency

$u_0^T L u_0 = \lambda_0 = 0$

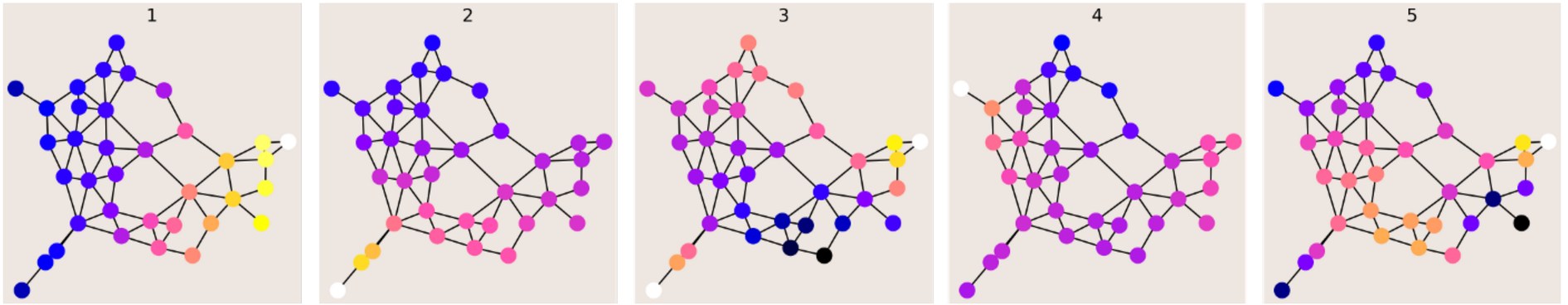
$u_1^T L u_1 = \lambda_1$

$u_7^T L u_7 = \lambda_7$

Visualize Eigen Vectors



Eigenfunction of the Laplacian for a square



Eigenfunction of the Laplacian for a random graph

https://towardsdatascience.com/graph-convolutional-networks-deep-99d7fee5706f

Graph Fourier Transform (GFT)

- With the eigen vectors $(u_0, \cdots, u_{N-1})$, we can define the Fourier transform on the graph. Imitating the definition of the Fourier transform in the continuous domain. **The Fourier transform on the graph is**:

$$F(\lambda_l) = \hat{f}(\lambda_l) = \sum_{i=1}^{N} f(i) u_l^*(i)$$

- Where f is the representation of the nodes on the graph (such as node Embedding), $f(i)$ represents i-th node, and $u_l(i)$ represents the i-th component of the l-th eigen vector. Then the Fourier transform of the graph of node Embedding $f(i)$ under the eigenvalue λ_l is the inner product operation of the eigenvector u_l corresponding to λ_l.
- Note: the above inner product operation is defined in the **complex space**, so we use the conjugation of $u_l(i)$, i.e., $u_l^*(i)$.

Graph Fourier Transform (GFT)

- Extend the Fourier transform on the graph to matrix form:

$$\begin{bmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_N) \end{bmatrix} = \begin{bmatrix} u_1(1) & u_1(2) & \dots & u_1(N) \\ u_2(1) & u_2(2) & \dots & u_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ u_N(1) & u_N(2) & \dots & u_N(N) \end{bmatrix} \begin{bmatrix} f(1) \\ f(2) \\ \vdots \\ f(N) \end{bmatrix}$$

- That is, the matrix form of the Fourier transform of f on the graph is:

$$\hat{f} = U^T f$$

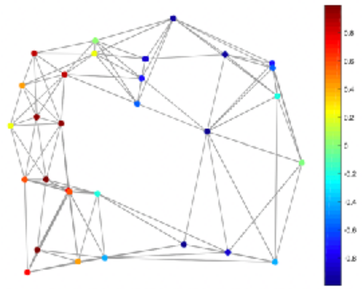
- Briefly speaking, given the input node Embedding f, multiply U^T to the left, then we can get the output Embedding $\hat{f}$ of the Fourier transform of f on the graph

Graph Fourier Transform (GFT)

A signal f can be written as graph Fourier series:

$$f = \sum_{i=0}^{N-1} \underbrace{\hat{f}_i}_{f^T u_i} \cdot u_i$$

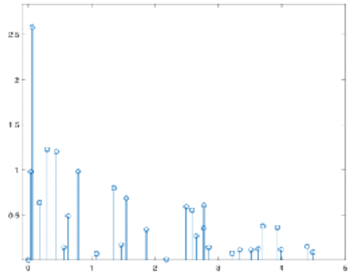
u_i: graph Fourier mode λ_i: frequency $\hat{f}_i$: graph Fourier coefficients



Spatial domain: f

$$\hat{\boldsymbol{f}} = \boldsymbol{U}^T \boldsymbol{f}$$

Decompose signal f



Spectral domain: $\hat{f}$

The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*

Inverse Graph Fourier Transform (IGFT)

- So, after we transform the node from the spatial domain to the frequency domain, how do we transform it back into the spatial domain?
- The traditional inverse Fourier transform is to integrate the frequency ω:

$$\mathcal{F}^{-1}[F(w)] = \frac{1}{2\pi}\int \mathcal{F}(w)e^{-iwt}dw$$

- The analogy to the discrete domain (on the figure) is to sum the eigenvalue λ_l:

$$f(i) = \sum_{l=1}^{N} \hat{f}(\lambda_l)u_l(i)$$

Inverse Graph Fourier Transform (IGFT)

- Extend the inverse Fourier transform on the graph to matrix form by matrix multiplication:

$$\begin{bmatrix} f(1) \\ f(2) \\ \vdots \\ f(N) \end{bmatrix} = \begin{bmatrix} u_1(1) & u_2(1) & \dots & u_N(1) \\ u_1(2) & u_2(2) & \dots & u_N(2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(N) & u_2(N) & \dots & u_N(N) \end{bmatrix} \begin{bmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_N) \end{bmatrix}$$

- That is, the matrix form of the inverse Fourier transform of f on the graph is:

$$f = U\hat{f}$$

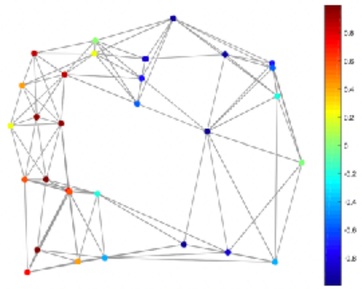
- In short, given the representation of the node Embedding in the frequency domain $\hat{f}$, multiply U to the left to get the representation of the node Embedding in the space domain.

Inverse Graph Fourier Transform (IGFT)

A signal f can be written as graph Fourier series:

$$f = \sum_{i=0}^{N-1} \underbrace{\hat{f}_i}_{f^T u_i} \cdot u_i$$

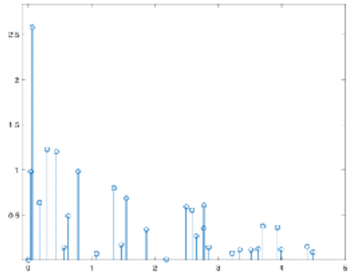
u_i: graph Fourier mode λ_i: frequency $\hat{f}_i$: graph Fourier coefficients



Spatial domain: f

$$\boldsymbol{f} = \boldsymbol{U}\hat{\boldsymbol{f}}$$

Reconstruct signal f



Spectral domain: $\hat{f}$

Todo

- **Suggested Readings**: The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains: https://arxiv.org/pdf/1211.0053.pdf

Next lecture: GNN and Graph-based NLP (II)

References

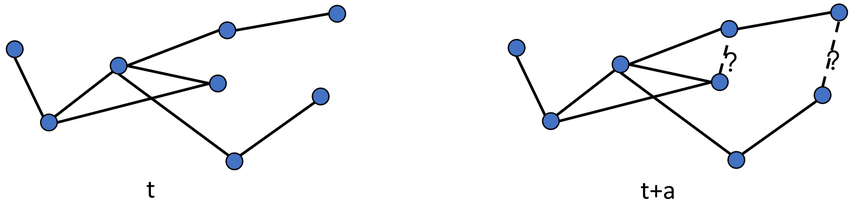
1. http://cse.msu.edu/~mayao4/tutorials/aaai2021/
2. https://luweikxy.gitbook.io/machine-learning-notes/graph-neural-networks/graph-convolutional-networks/gcn-comprehensive-understand
3. https://www.youtube.com/watch?v=spUNpyF58BY&ab_channel=3Blue1Brown
4. https://www.youtube.com/watch?v=r6sGWTCMz2k&t=83s&ab_channel=3Blue1Brown

Graph Neural Networks

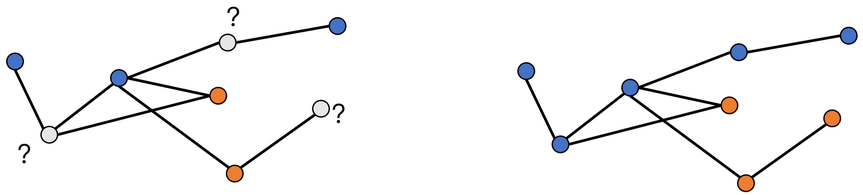
Tasks on Graph-Structured Data

Node-level

Link Prediction

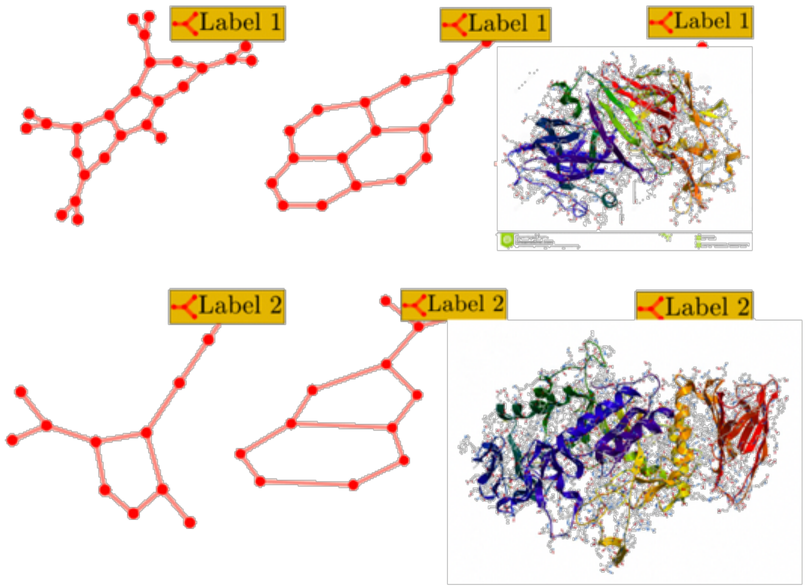


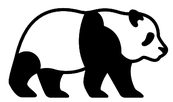
Node Classification



Graph-level

Graph Classification

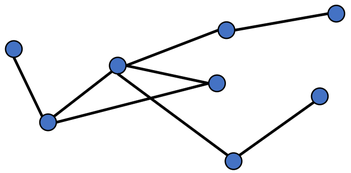




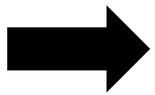
Tasks on Graph-Structured Data

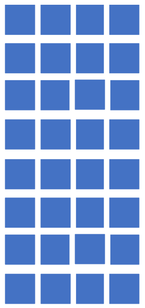
Node-level

Graph-level

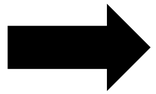


Filtering





Pooling



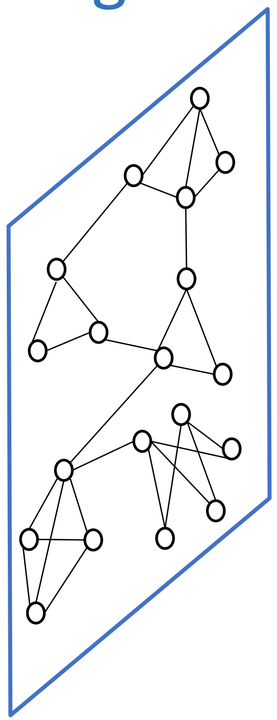
Node Representations

Graph Representations

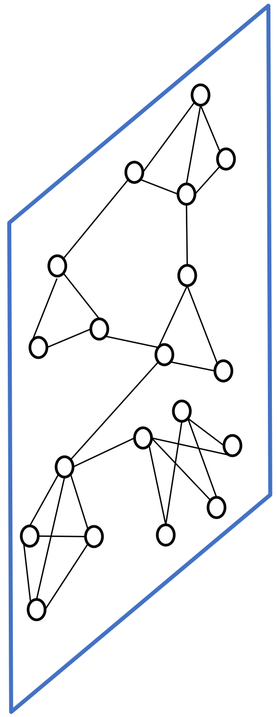
For node level, we want to transform the original graph signals to get better node representations.
For graph level, we want to perform pooling to get whole graph representation

Two Main Operations in GNN

Graph Filtering



Graph Filtering



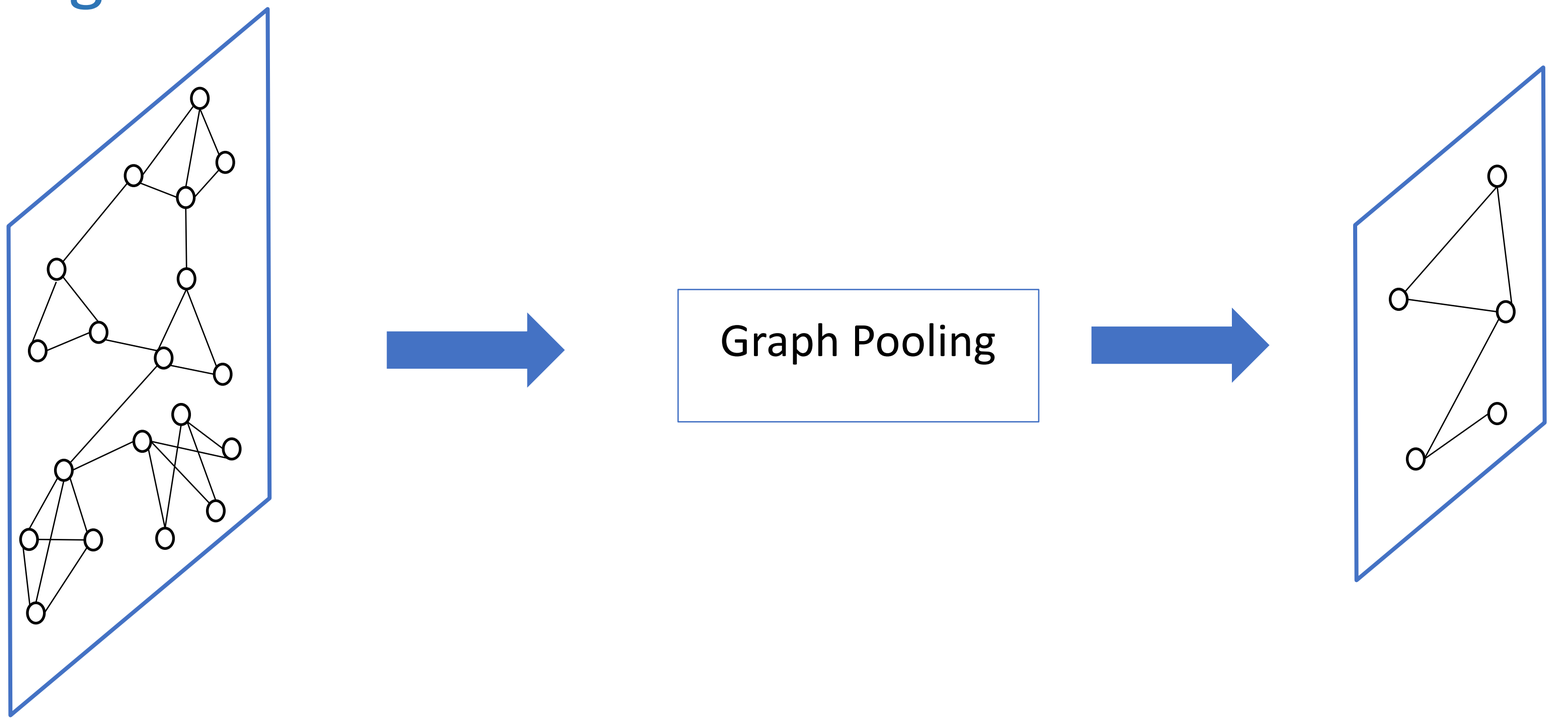
$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$

$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X}_f \in \mathbb{R}^{n \times d_{new}}$

Graph filtering refines the node features

Two Main Operations in GNN

Graph Pooling



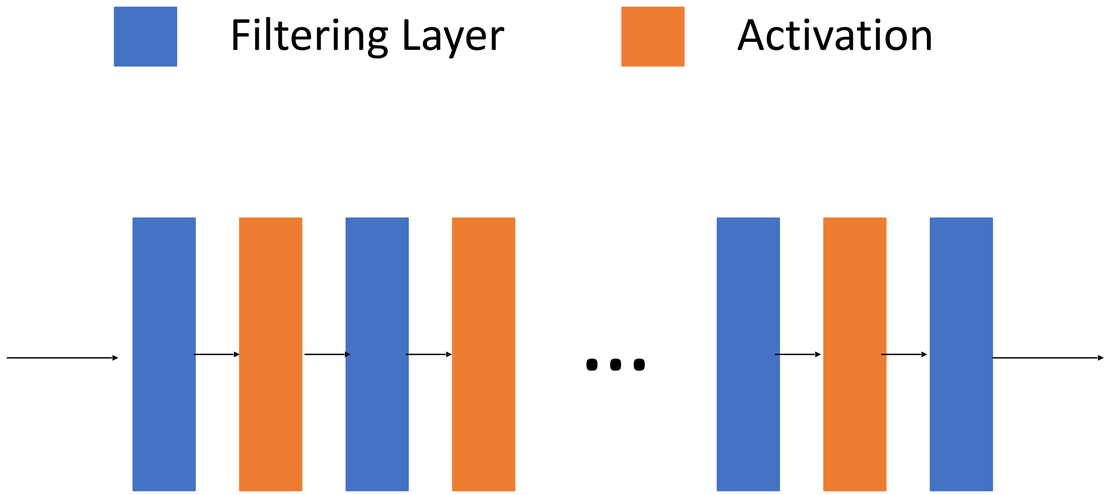
$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{X}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

Graph pooling generates a smaller graph

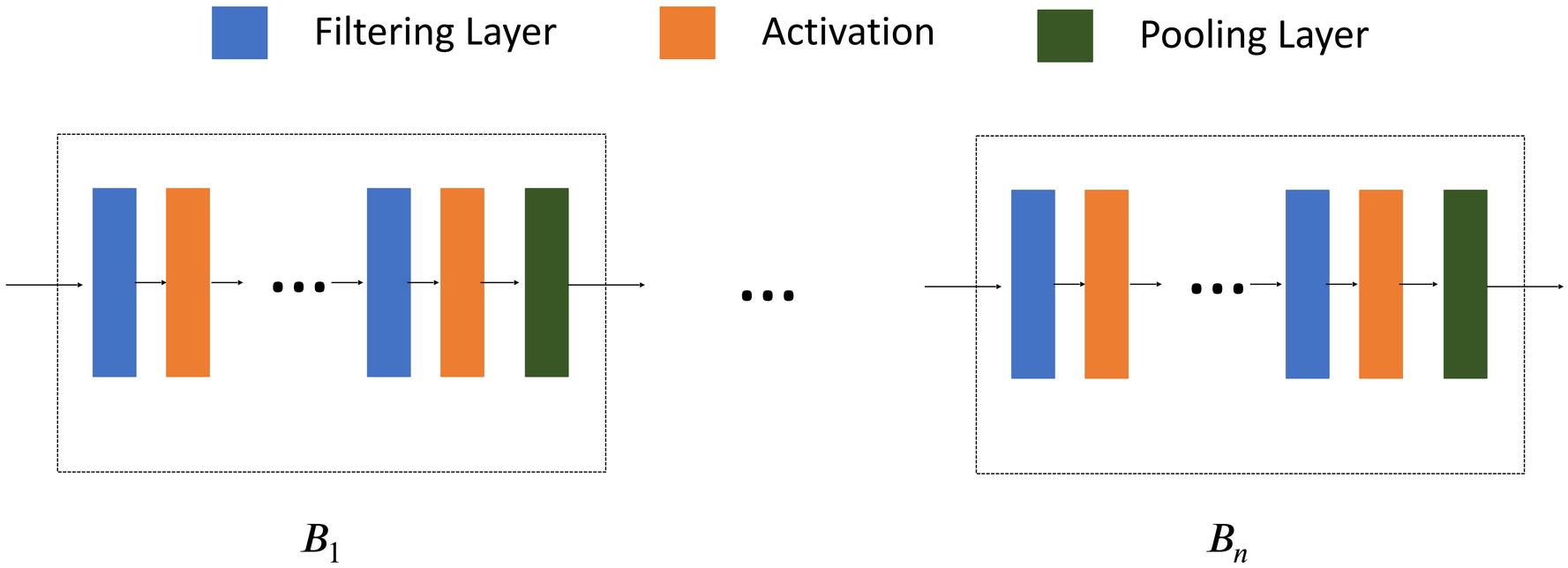
General GNN Framework

For node-level tasks

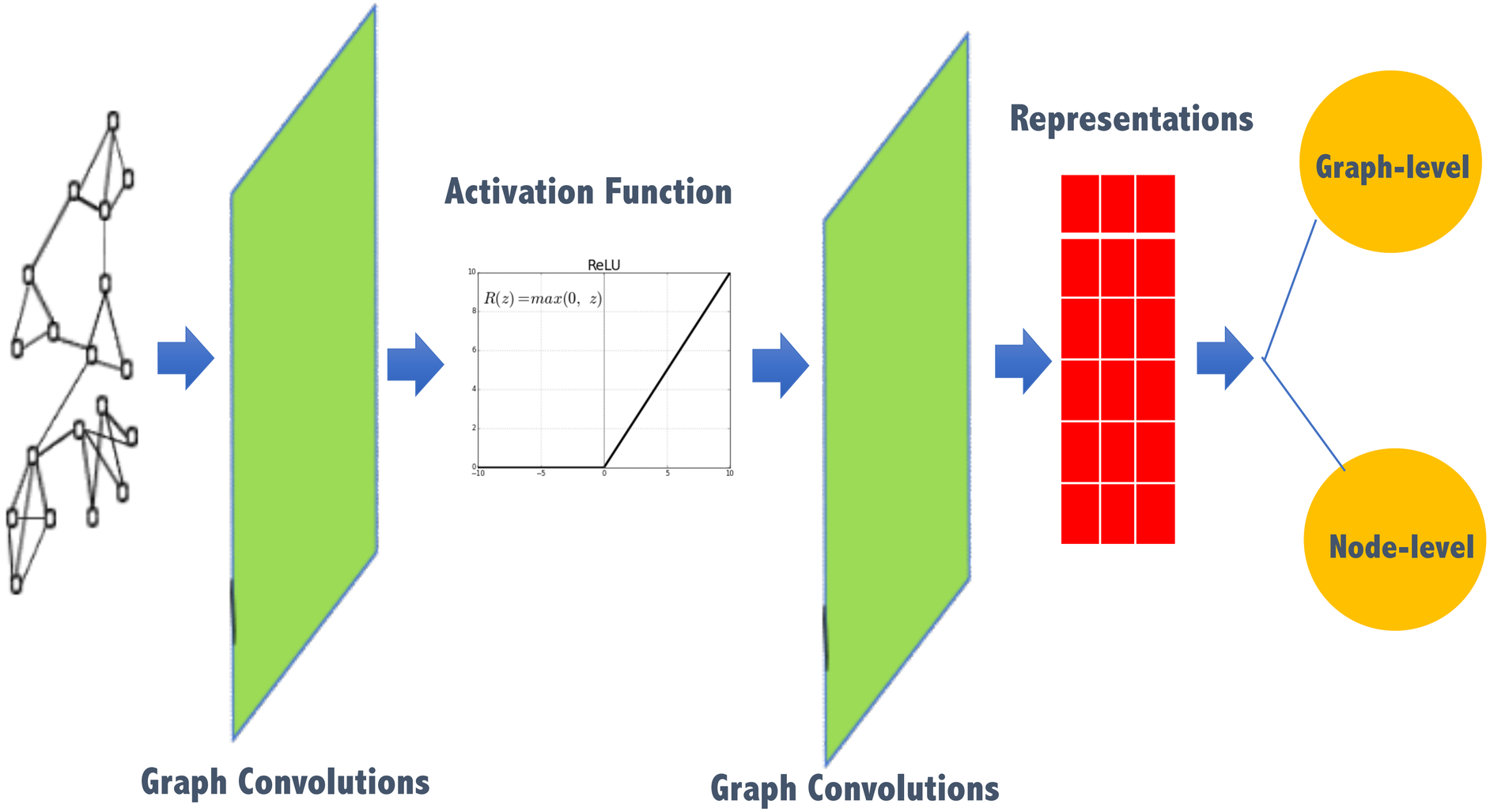


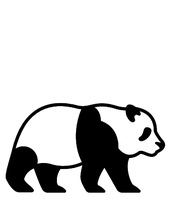
General GNN Framework

For graph-level tasks

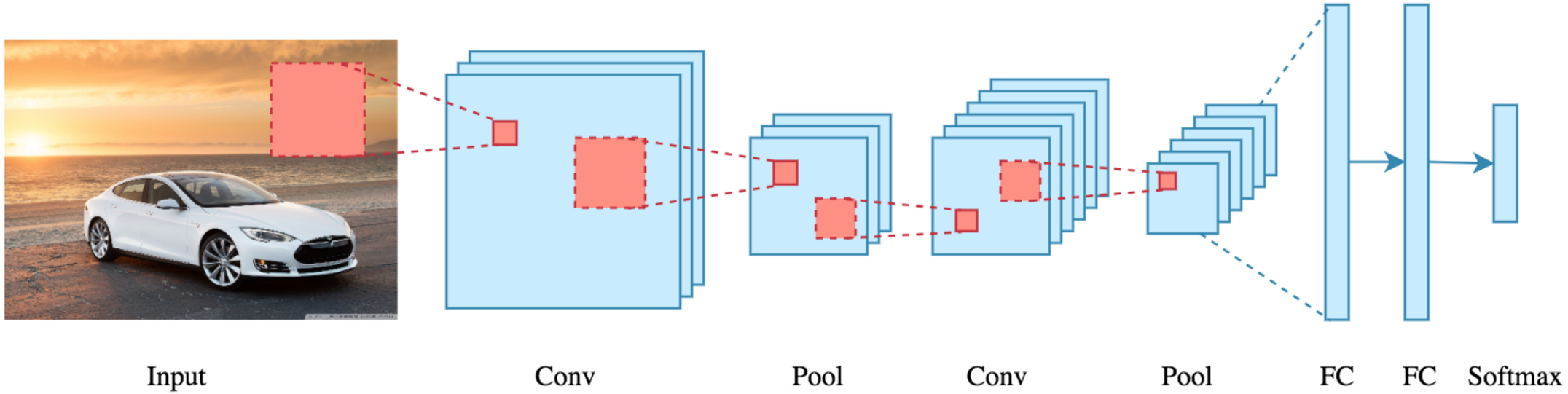


Graph Neural Networks

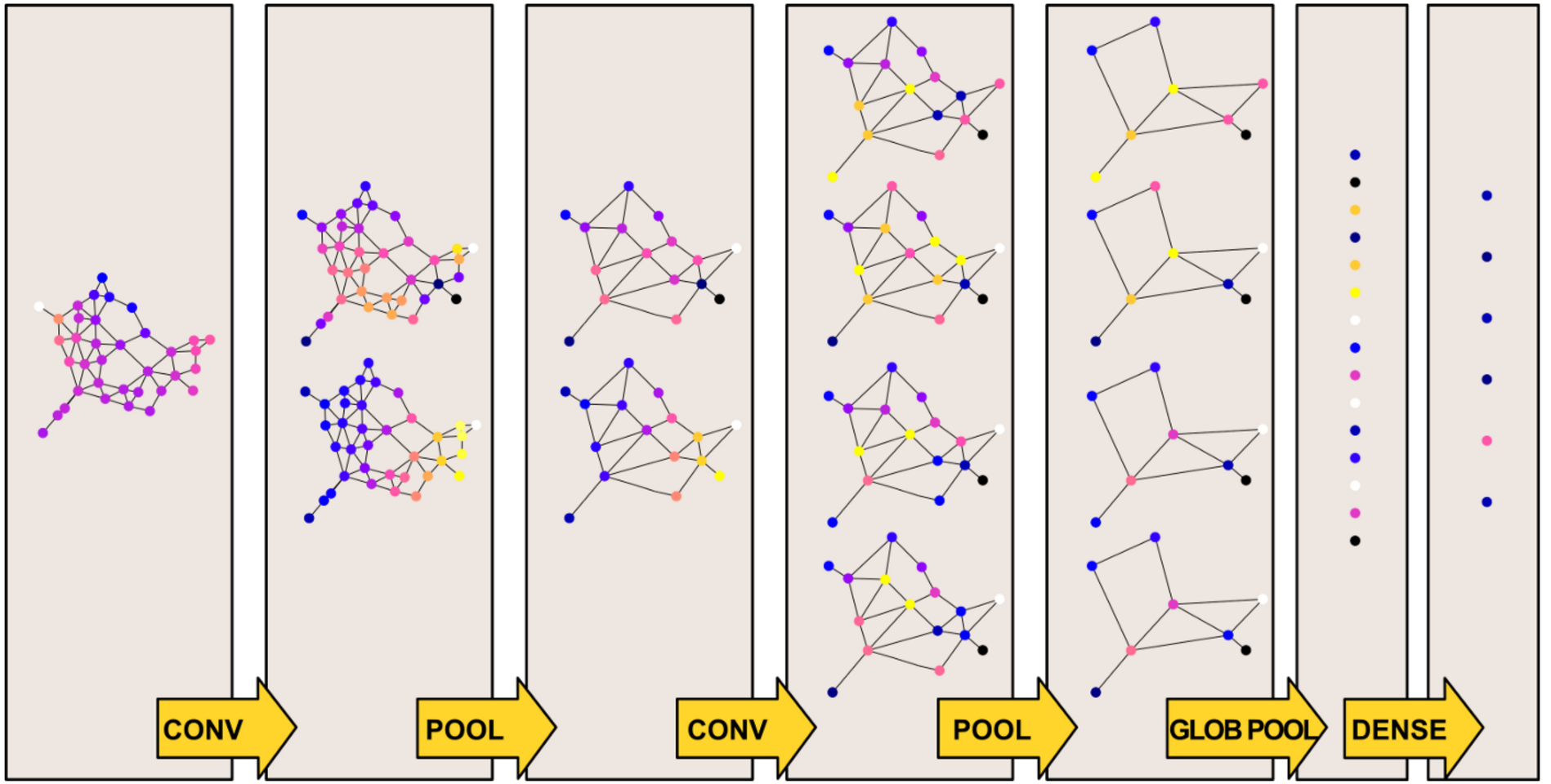




Recap Convolutional Neural Networks



Graph Neural Networks

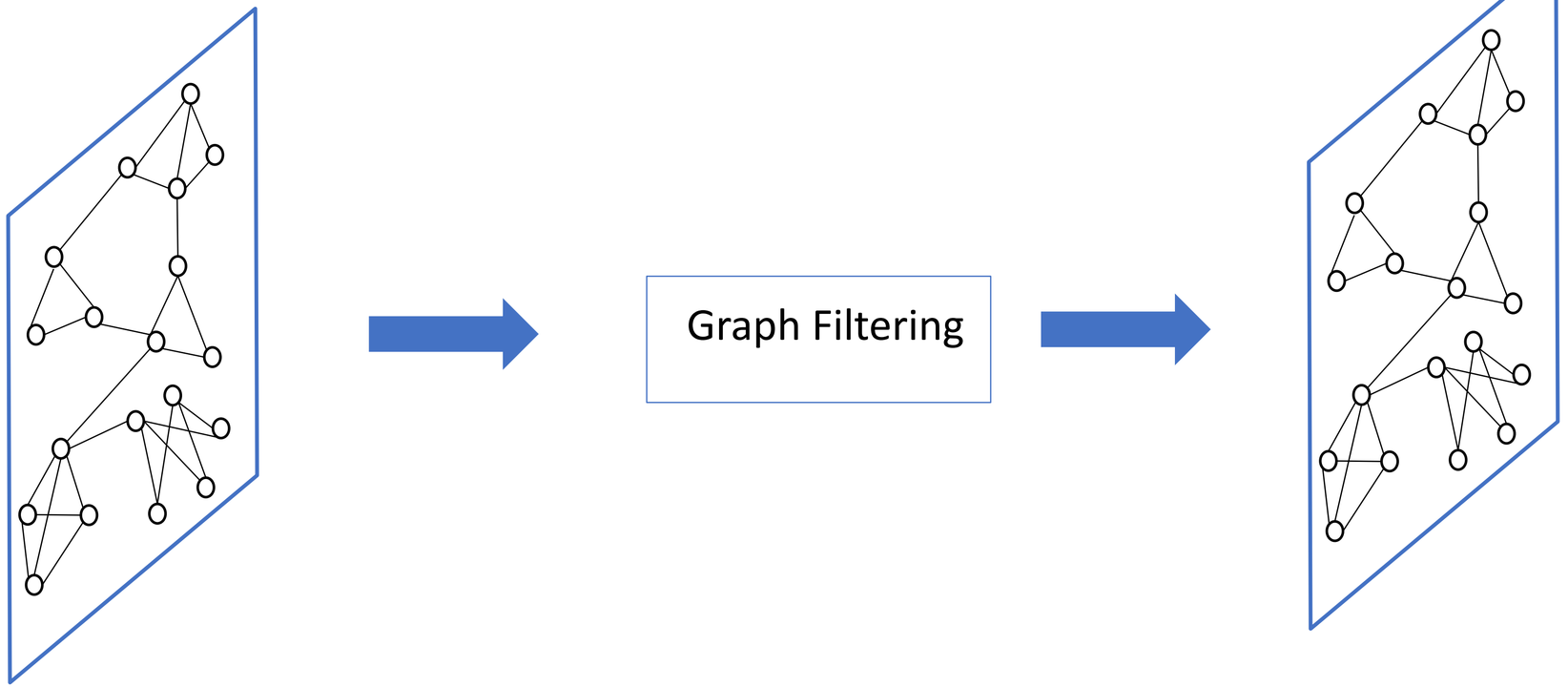


Lecture outline

1. Why graphs for NLP?
2. Modeling text as graphs
3. Graph neural networks
4. Text clustering: Story Forest for fine-grained events detection and organization
5. Text matching: matching article pairs with graphical decomposition and convolutions

Graph Filtering

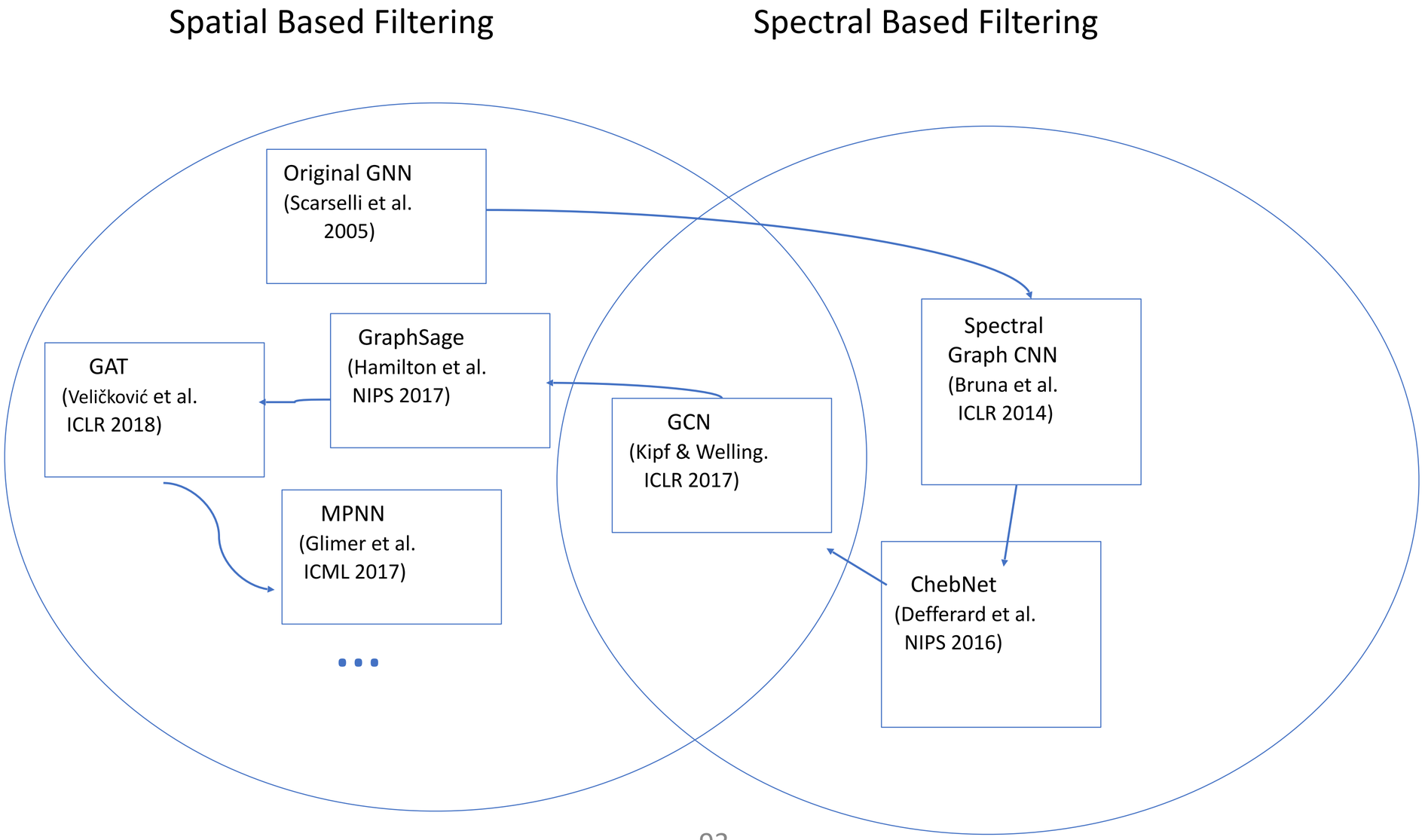
Graph Filtering Operation



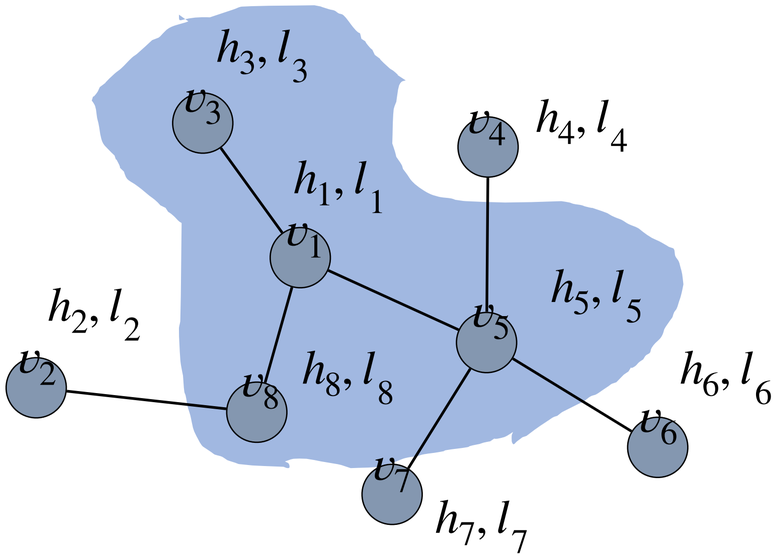
$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$

$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X}_f \in \mathbb{R}^{n \times d_{new}}$

Two Types of Graph Filtering Operation



Graph Filtering in the First GNN Paper

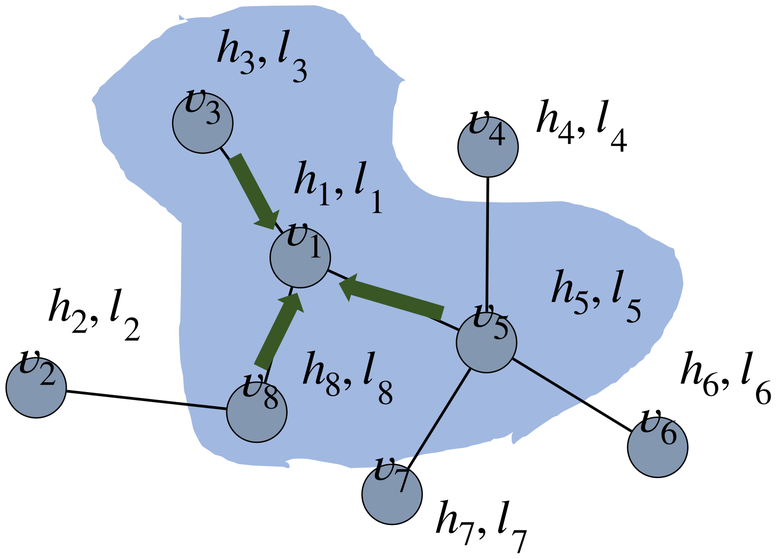


h_i: The hidden features

l_i: The input features

Graph neural networks for ranking web pages. *WI*. IEEE, 2005.

Graph Filtering in the First GNN Paper



h_i: The hidden features

l_i: The input features

$$h_i^{(k+1)} = \sum_{v_j \in N(v_i)} f\left(l_i, h_j^{(k)}, l_j\right), \quad \forall\, v_i \in V.$$

$N(v_i)$: Neighbors of the node v_i.

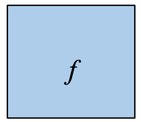
$f(\cdot)$: Feedforward neural network.

Graph Spectral Filtering for Graph Signal

Recall:

$$GFT: \hat{f} = U^T f \qquad\qquad IGFT: f = U\hat{f}$$

Filter a graph signal f:

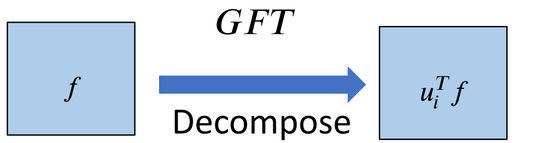


Graph Spectral Filtering for Graph Signal

Recall:

$$GFT: \hat{f} = U^T f \qquad\qquad IGFT: f = U\hat{f}$$

Filter a graph signal f:

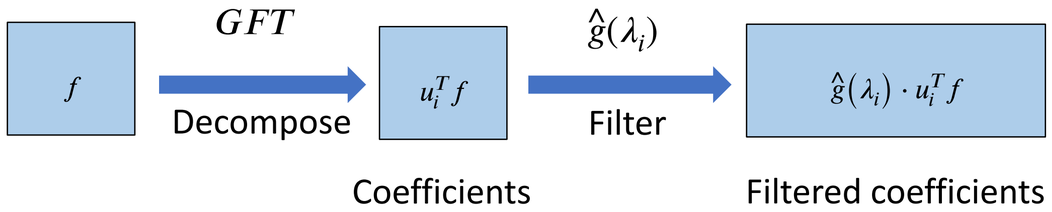


Graph Spectral Filtering for Graph Signal

Recall:

$$GFT: \hat{f} = U^T f \qquad\qquad IGFT: f = U\hat{f}$$

Filter a graph signal f:



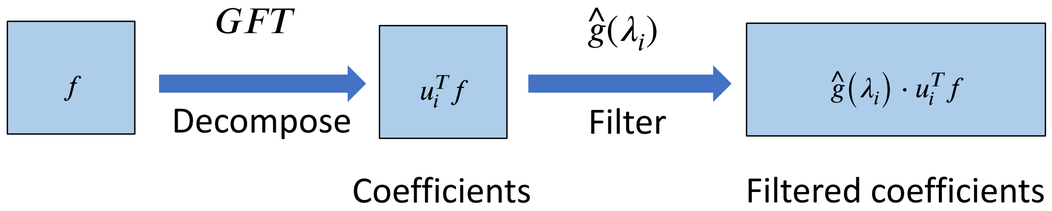
Filter $\hat{g}(\lambda_i)$: Modulating the frequency

Graph Spectral Filtering for Graph Signal

Recall:

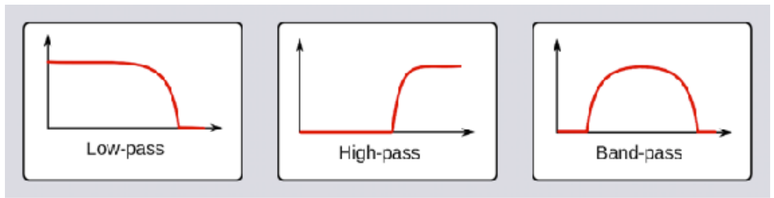
$$GFT: \hat{f} = U^T f \qquad\qquad IGFT: f = U\hat{f}$$

Filter a graph signal f:



Filter $\hat{g}(\lambda_i)$: Modulating the frequency

Example:

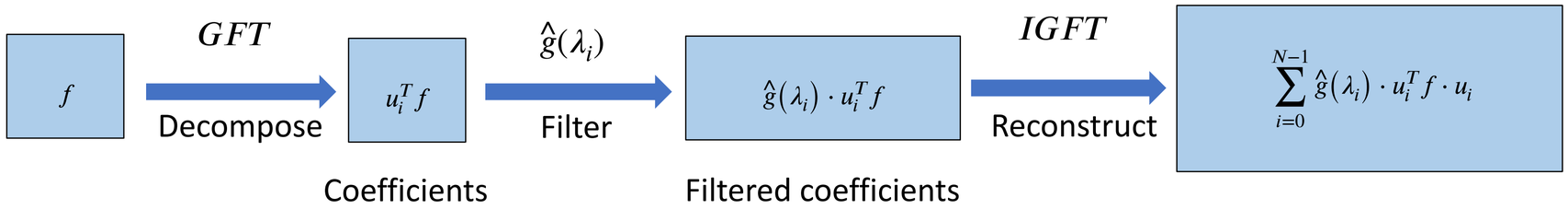


Graph Spectral Filtering for Graph Signal

Recall:

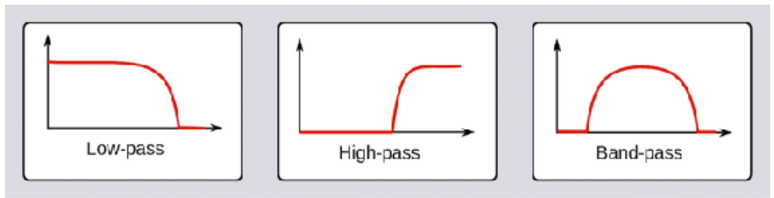
$$GFT: \hat{f} = U^T f \qquad\qquad IGFT: f = U\hat{f}$$

Filter a graph signal f:



Filter $\hat{g}(\lambda_i)$: Modulating the frequency

Example:

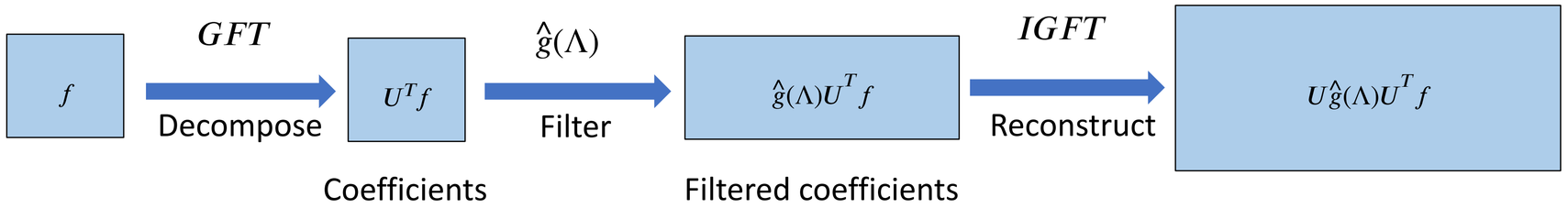


Graph Spectral Filtering for Graph Signal

Recall:

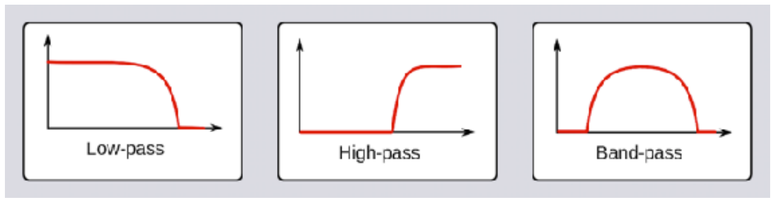
$$GFT: \hat{f} = U^T f \qquad\qquad IGFT: f = U\hat{f}$$

Filter a graph signal f:



$$\hat{g}(\Lambda) = \begin{bmatrix} \hat{g}(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_{N-1}) \end{bmatrix}$$

Example:

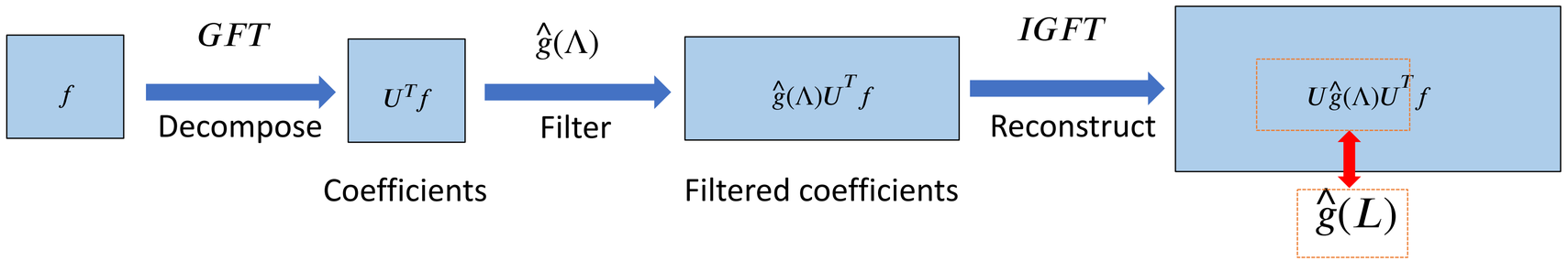


Graph Spectral Filtering for Graph Signal

Recall:

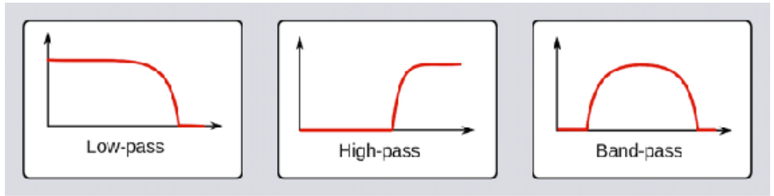
$$GFT: \hat{f} = U^T f \qquad\qquad IGFT: f = U\hat{f}$$

Filter a graph signal f:



$$\hat{g}(\Lambda) = \begin{bmatrix} \hat{g}(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_{N-1}) \end{bmatrix}$$

Example:

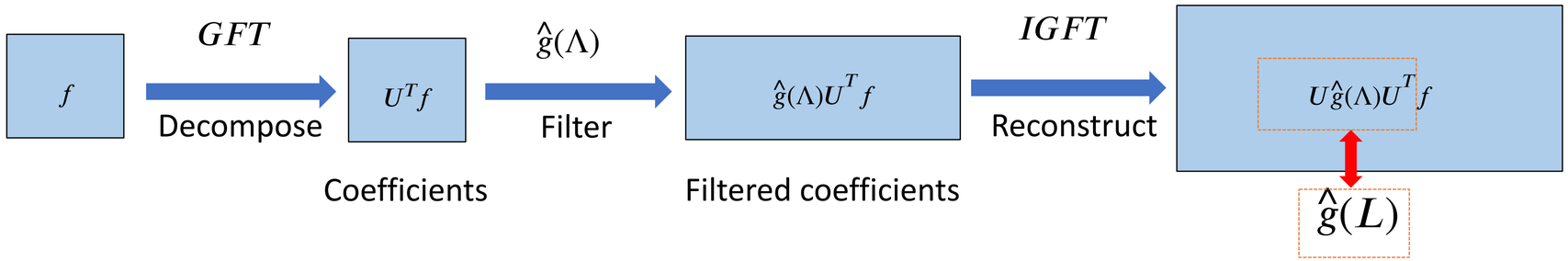


Graph Spectral Filtering for Graph Signal

Recall:

$$GFT: \hat{f} = U^T f \qquad\qquad IGFT: f = U\hat{f}$$

Filter a graph signal f:



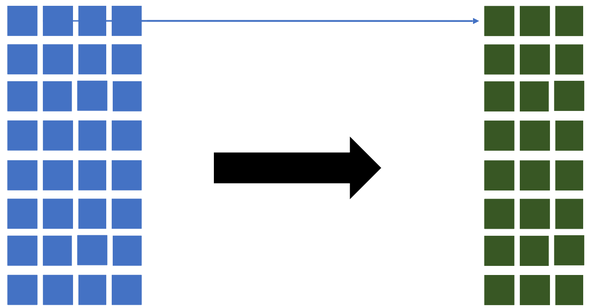
$$\hat{g}(\Lambda) = \begin{bmatrix} \hat{g}(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_{N-1}) \end{bmatrix}$$

Graph Spectral Filtering for GNN

How to design the filter?

Data-driven! Learn $\hat{g}(\Lambda)$ from data!

How to deal with multi-channel signals?



$$\boldsymbol{F}_{in} \in \mathbb{R}^{N \times d_1} \rightarrow \boldsymbol{F}_{out} \in \mathbb{R}^{N \times d_2}.$$

Each input channel contributes to each output channel

$$\boldsymbol{F}_{out}[:, i] = \sum_{j=1}^{d_1} \hat{g}_{ij}(\mathbf{L}) \boldsymbol{F}_{in}[:, j] \qquad i = 1, \ldots d_2$$

Filter each input channel

Learn $d_2 \times d_1$ filters

$\hat{g}(\Lambda)$: Non-parametric

$$\hat{g}(\Lambda) = \begin{bmatrix} \hat{g}(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_{N-1}) \end{bmatrix}$$

Let's first introduce a non-parametric method to implement g hat (filter over different frequencies lambda)

$\hat{g}(\Lambda)$: Non-parametric

$$\hat{g}(\Lambda) = \begin{bmatrix} \theta_1 & & & \\ & \theta_2 & & \\ & & \dots & \\ & & & \theta_N \end{bmatrix}$$

A intuitive idea is: learn it directly. which is the idea in this paper.

Spectral Networks and Locally Connected Networks on Graphs. ICLR 2014

$\hat{g}(\Lambda)$: Non-parametric

$$\hat{g}(\Lambda) = \begin{bmatrix} \theta_1 & & & \\ & \theta_2 & & \\ & & \cdots & \\ & & & \theta_N \end{bmatrix}$$

$d_2 \times d_1 \times N$ parameters

Too many, not scalable

$$U\hat{g}(\Lambda)U^T f$$

Expensive eigen-decomposition

$\hat{g}(\Lambda)$: Polynomial Parametrized

$$\hat{g}(\Lambda) = \begin{bmatrix} \sum_{k=0}^{K} \theta_k \lambda_1^k & & & \\ & \sum_{k=0}^{K} \theta_k \lambda_2^k & & \\ & & \cdots & \\ & & & \sum_{k=0}^{K} \theta_k \lambda_N^k \end{bmatrix}$$

New form: polynomial parametrized. We learn a k-order polynomial expression. The idea is: we assume the transformation $\hat{g}$ is a polynomial function, and all the λ (different frequencies) share the same function

Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. NIPS 2016.

$\hat{g}(\Lambda)$: Polynomial Parametrized

$$\hat{g}(\Lambda) = \begin{bmatrix} \sum_{k=0}^{K} \theta_k \lambda_1^k & & & \\ & \sum_{k=0}^{K} \theta_k \lambda_2^k & & \\ & & \cdots & \\ & & & \sum_{k=0}^{K} \theta_k \lambda_N^k \end{bmatrix}$$

$d_2 \times d_1 \times K$ parameters

Less parameters.

$$U\hat{g}(\Lambda)U^T f = \sum_{k=0}^{K} \theta_k L^k f$$

No eigen-decomposition needed

Polynomial Parametrized Filter: a Spatial View

$$U\hat{g}(\Lambda)U^T f(i) = \sum_{j=0}^{N}\sum_{k=0}^{K}\theta_k L_{i,j}^k f(j)$$

If the node v_j is more than K-hops away from node v_i, then,

$$\sum_{k=0}^{K}\theta_k L_{i,j}^k = 0$$

The filter is localized within K-hops neighbors in the spatial domain

Chebyshev Polynomials

The polynomials adopted have non-orthogonal basis $1, x, x^2, x^3, \ldots$

$$g(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots$$

Unstable under perturbation of coefficients

Chebyshev polynomials:

Recursive definition:

- $T_0(x) = 1; T_1(x) = x$
- $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$

The Chebyshev polynomials $\{T_k\}$ form an orthogonal basis for the Hilbert space $L^2([-1,1], \frac{dy}{\sqrt{1-y^2}})$.

https://en.wikipedia.org/wiki/Chebyshev_polynomials

$$g(x) = \theta_0 T_0(x) + \theta_1 T_1(x) + \theta_2 T_2(x) + \cdots$$

ChebNet

Parametrize $\hat{g}(\Lambda)$ with Chebyshev polynomials

$$\hat{g}(\Lambda) = \sum_{k=0}^{K} \theta_k T_k(\tilde{\Lambda}), with\ \tilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - 1$$

($\tilde{\Lambda}$ will be between -1 and 1)

$d_2 \times d_1 \times K$ parameters

$U\hat{g}(\Lambda)U^T f = \sum_{k=0}^{K} \theta_k T_k(\tilde{L}) f$, with $\tilde{L} = \frac{2L}{\lambda_{max}} - I$

No eigen-decomposition needed

Stable under perturbation of coefficients

GCN: Simplified ChebNet

Use Chebyshev polynomials with $K = 1$ and assume $\lambda_{max} = 2$

$$\hat{g}(\Lambda) = \theta_0 + \theta_1(\Lambda - I)$$

Further constrain $\theta = \theta_0 = -\theta_1$

$$\hat{g}(\Lambda) = \theta(2I - \Lambda)$$

$$U\hat{g}(\Lambda)U^T f = \theta(2I - L)f = \theta\left(I + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}\right)f$$

Apply a renormalization trick

$$U\hat{g}(\Lambda)U^T f = \theta\left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}\right)f, with\ \hat{A} = A + I$$

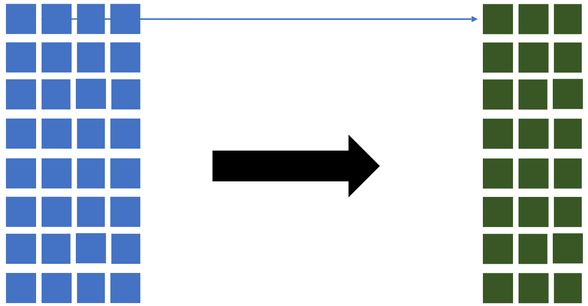
Kipf and Welling ICLR 2017, Semi-Supervised Classification with Graph Convolutional Networks

GCN for Multi-channel Signal

Recall:

$$\boldsymbol{F}_{out}[:,i] = \sum_{j=1}^{d_1} \hat{g}_{ij}(\mathbf{L})\boldsymbol{F}_{in}[:,j] \qquad i = 1,\ldots d_2$$

Filter each input channel



For GCN:

$$\boldsymbol{F}_{out}[:,i] = \sum_{j=1}^{d_1} \theta_{ji}(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}})\boldsymbol{F}_{in}[:,j] \qquad i = 1,\ldots d_2$$

GCN filter

In matrix form:

$$\boldsymbol{F}_{out} = (\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}})\boldsymbol{F}_{in}\Theta \text{ with } \Theta \in \mathbb{R}^{d_1 \times d_2} \text{ and } \Theta[j,i] = \theta_{ji}$$

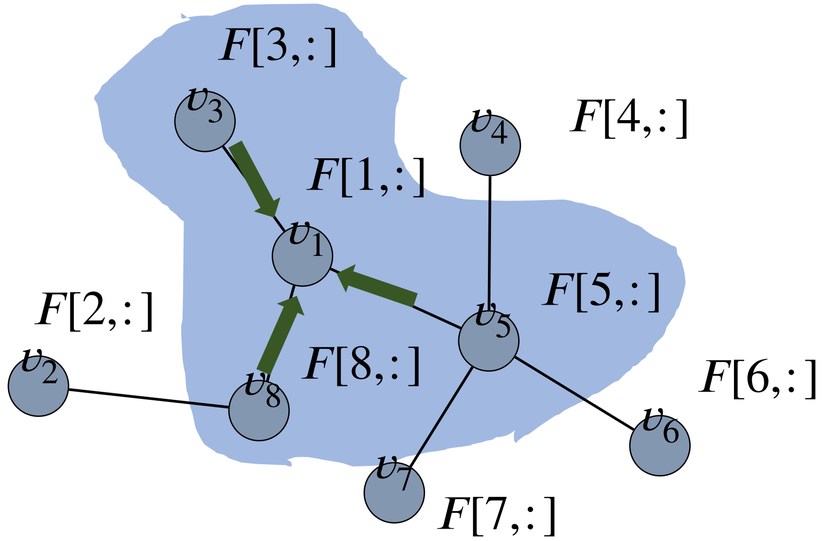
A Spatial View of GCN Filter

Denote $C = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$

- Then $F_{out} = CF_{in}\Theta$
- For node v_i, $F_{out}[i,:] = \sum_j C[i,j]F_{in}[j,:]\Theta$

Observe that:

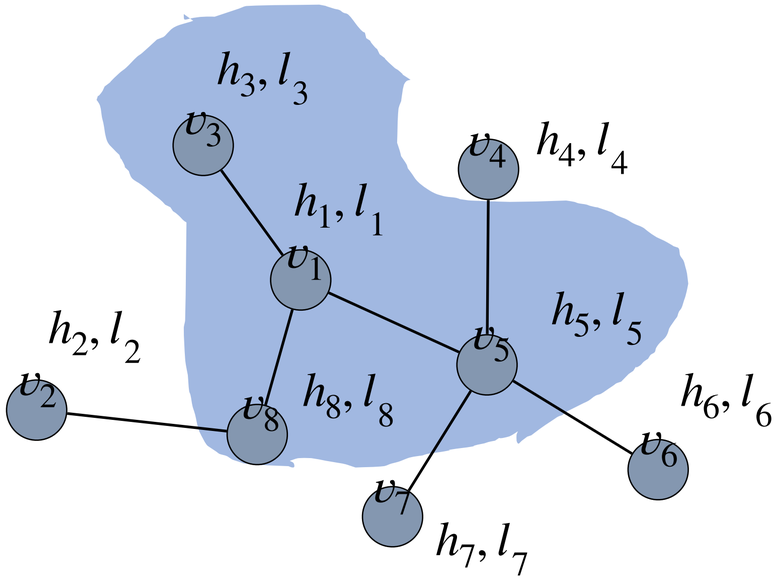
- $C[i,j] = 0$ for $v_j \notin N(v_i) \cup \{v_i\}$



Hence,

$$\mathbf{F}_{out}[i,:] = \underbrace{\sum_{v_j \in \mathcal{N}(v_i)\cup\{v_i\}}}_{\text{Aggregation}} \underbrace{C[i,j]\mathbf{F}_{in}[j,:]\Theta}_{\text{Feature transformation}}$$

Filter in GCN v.s. Filter in the First GNN



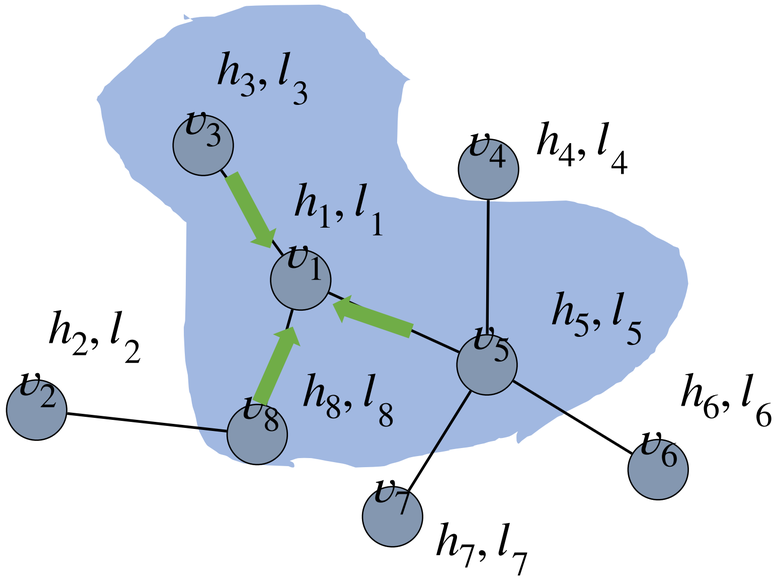
GCN: k-th layer

$$\mathbf{h}_i^{(k+1)} = \sum_{v_j \in \mathcal{N}(v_i) \cup \{v_i\}} C[i,j]\mathbf{h}_j^{(k)}\Theta, \forall v_i \in \mathcal{V}$$

The first GNN: k-th layer

$$\mathbf{h}_i^{(k+1)} = \sum_{v_j \in \mathcal{N}(v_i)} f(l_i, \mathbf{h}_j^{(k)}, l_j), \forall v_i \in \mathcal{V}$$

Filter in GCN VS Filter in the First GNN



GCN: k-th layer

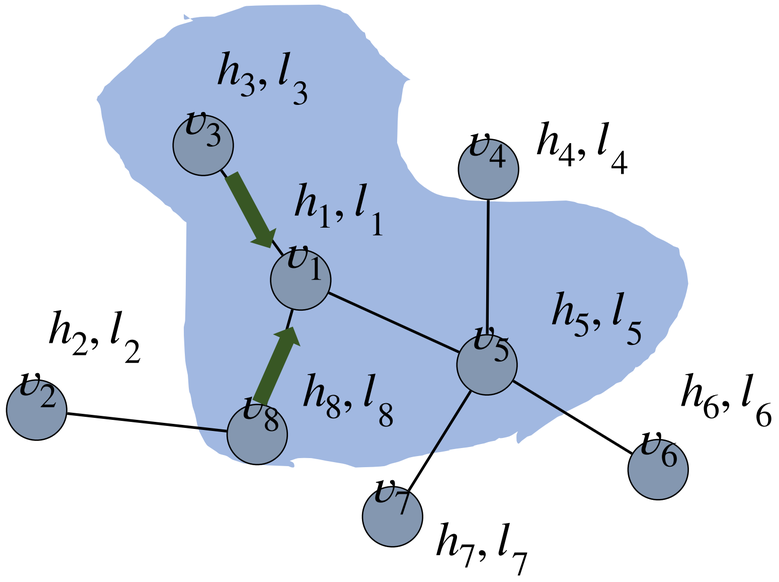
$$\mathbf{h}_i^{(k+1)} = \sum_{v_j \in \mathcal{N}(v_i) \cup \{v_i\}} C[i,j]\mathbf{h}_j^{(k)}\Theta, \forall v_i \in \mathcal{V}$$

The first GNN: k-th layer

$$\mathbf{h}_i^{(k+1)} = \sum_{v_j \in \mathcal{N}(v_i)} f(l_i, \mathbf{h}_j^{(k)}, l_j), \forall v_i \in \mathcal{V}$$

Both do feature transformation and feature aggregation. They share similar form.
Key difference: different transformation, different ways to use graph structure.

Filter in GraphSage



Neighbor Sampling

$$\mathcal{N}(v_i) \rightarrow \mathcal{N}_s(v_i)$$

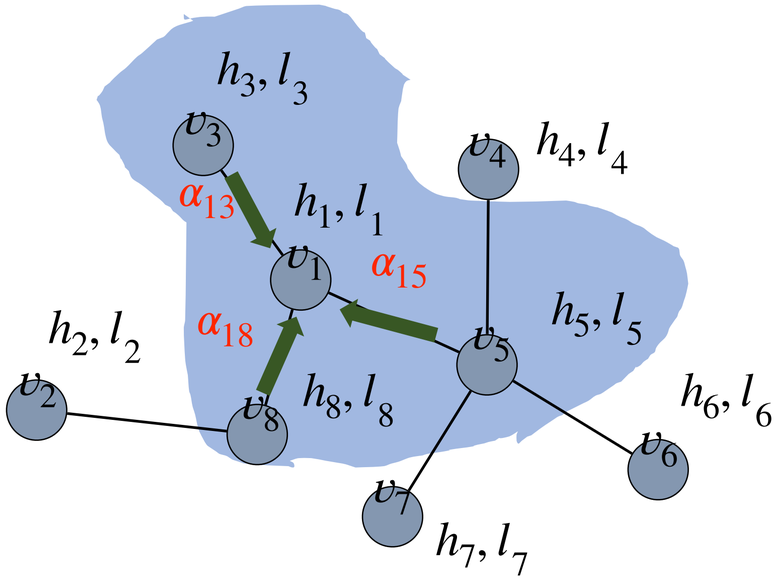
Aggregation

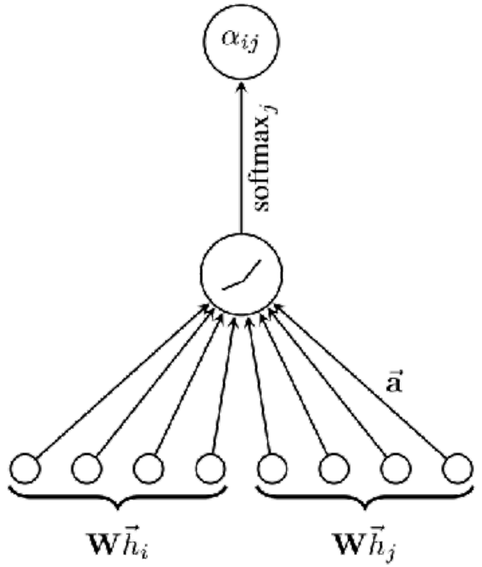
$$\mathbf{h}_{\mathcal{N}_s(v_i)}^{(k+1)} = \underline{AGG}(\{\mathbf{h}_i^{(k)}, v_j \in \mathcal{N}_s(v_i)\})$$

$$\mathbf{h}_i^{(k+1)} = \sigma(\Theta[\mathbf{h}_i^{(k)}, \mathbf{h}_{\mathcal{N}_s(v_i)}^{(k+1)}])$$

Inductive Representation Learning on Large Graphs. NIPS 2017.

Filter in GAT



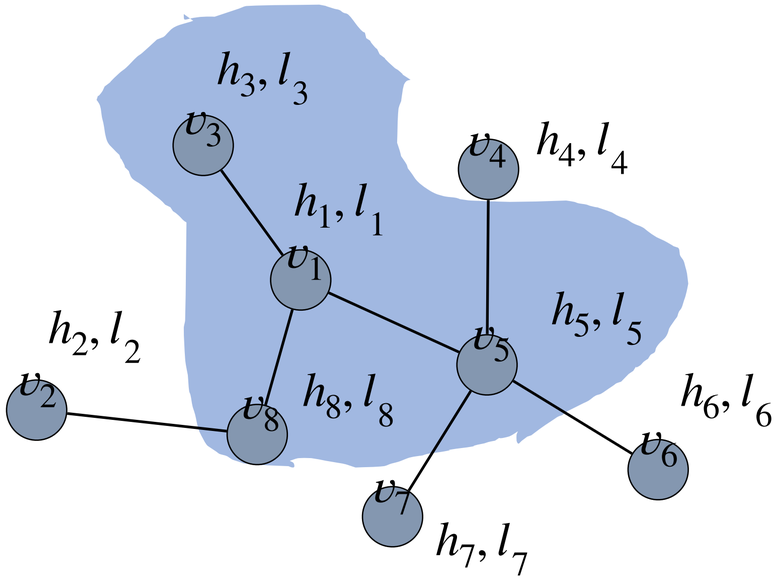


$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

Previously we treat each neighbour equally.
But not good. Different node can be different important.

GAT: weight!

Filter in MPNN



Message Passing

$$m_i^{(k+1)} = \sum_{v_j \in N(v_i)} M_k \left(h_i^{(k)}, h_j^{(k)}, e_{ij} \right)$$

Feature Updating

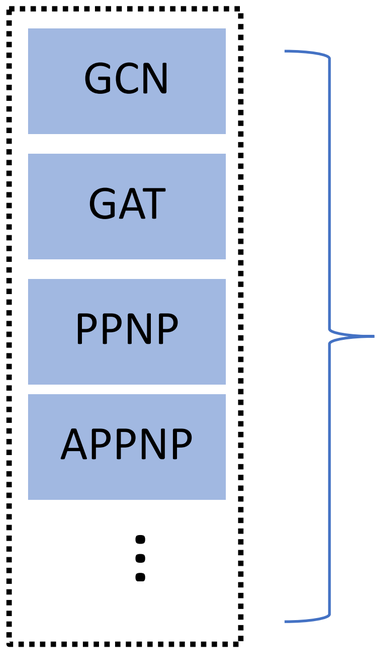
$$h_i^{(k+1)} = U_k \left(h_i^{(k)}, m_i^{(k+1)} \right)$$

$M_k()$ and $U_k()$ are functions to be designed

Neural Message Passing for Quantum Chemistry. ICML 2017.

UGNN: A Unified Framework

Filtering Operations



Graph Signal Denoising

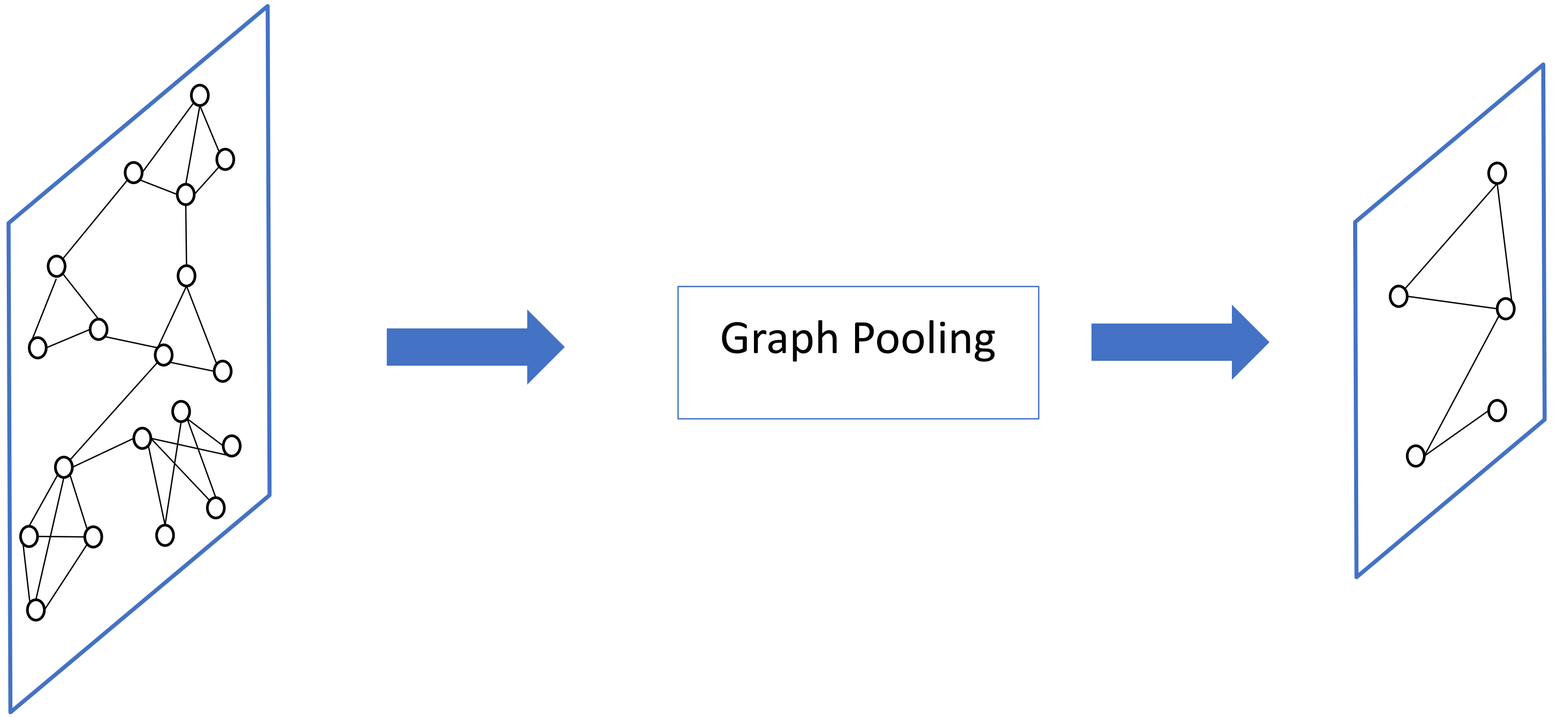
$$\arg\min_{\mathbf{X}_f} \mathcal{L}\left(\mathbf{X}_f\right) = \|\mathbf{X}_f - \mathbf{X}'\|_F^2 + c \cdot \frac{1}{2} \sum_{i \in \mathcal{V}} \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \|\mathbf{X}_f[i,:] - \mathbf{X}_f[j,:]\|_2^2$$

More recently, a work says that many filtering operations are all solving this graph signal denoting problem. For details, see the paper.

A Unified View on Graph Neural Networks as Graph Signal Denosing. arXiv, 2020.

Graph Pooling

Graph Pooling Operation



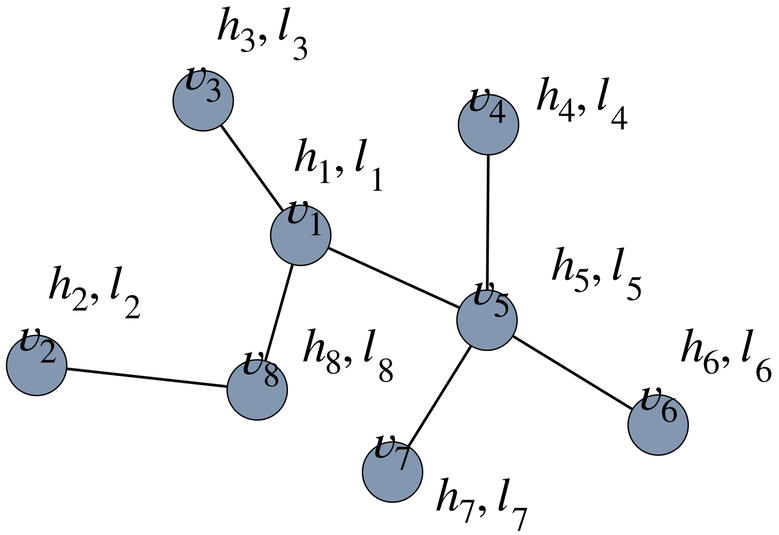
$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$

$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{X}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$

Graph pooling:
1. number of nodes change
2. feature dimension may also change.

gPool

Downsample by selecting the most importance nodes



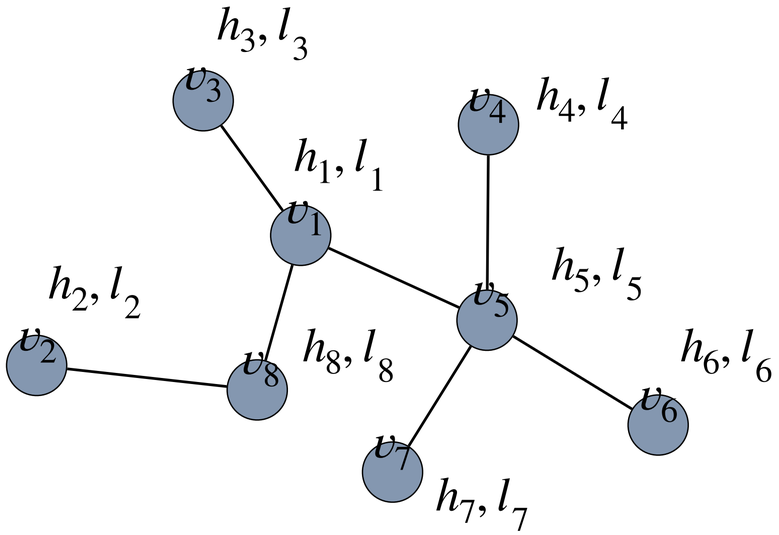
$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

Graph U-Nets. ICML 2019.

gPool

Downsample by selecting the most importance nodes



h: node feature
p: learn a project
y: the importance of vertex v.

Importance Measure

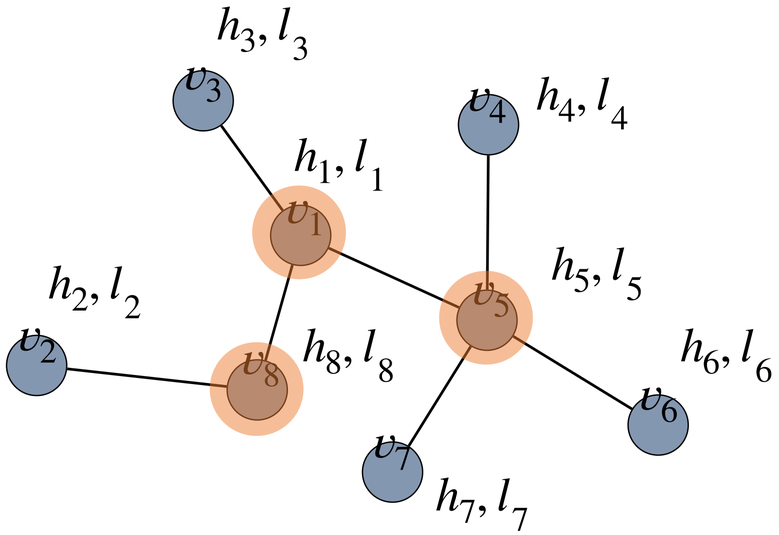
$$v_i \rightarrow y_i \qquad y_i = \frac{h_i^T p}{||p||}$$

$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

gPool

Downsample by selecting the most importance nodes



$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

rank nodes by importance
select top np nodes

Importance Measure

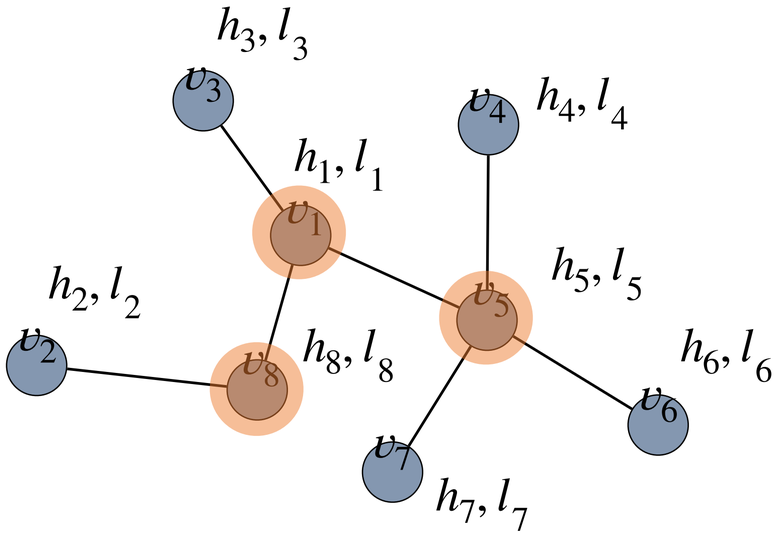
$$v_i \rightarrow y_i \qquad y_i = \frac{h_i^T p}{||p||}$$

Select top the n_p nodes

$$idx = rank(\mathbf{y}, n_p)$$

gPool

Downsample by selecting the most importance nodes



$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

Importance Measure

$$v_i \rightarrow y_i \qquad y_i = \frac{h_i^T p}{||p||}$$

Select top the n_p nodes

$$idx = rank(\mathbf{y}, n_p)$$

Generate A_p and intermediate H_{inter}

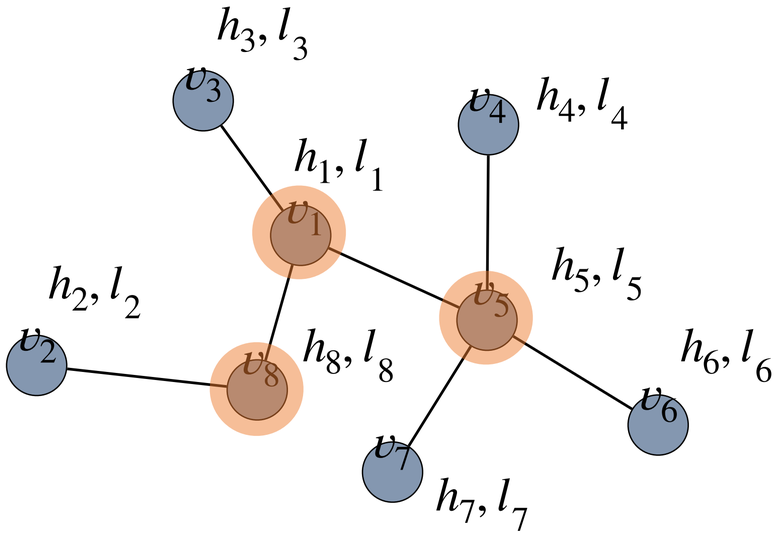
select adjacent matrix's and features' subset

$$A_p = A[idx, idx]$$

$$H_{inter} = H[idx, :]$$

gPool

Downsample by selecting the most importance nodes



$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

get new node features by importance related weight and original features

Importance Measure

$$v_i \rightarrow y_i \qquad y_i = \frac{h_i^T p}{||p||}$$

Select top the n_p nodes

$$idx = rank(\boldsymbol{y}, n_p)$$

Generate A_p and intermediate H_{inter}

$$A_p = A[idx, idx]$$

$$H_{inter} = H[idx, :]$$

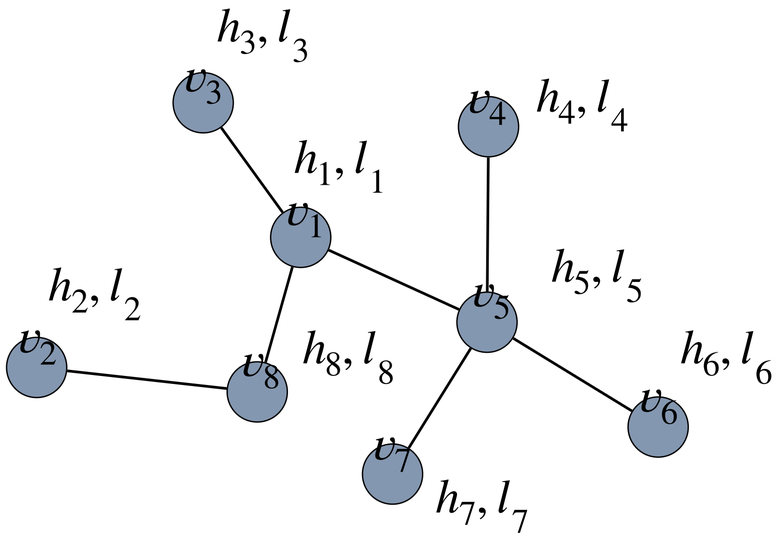
Generate H_p

$$\tilde{y} = sigmoid(y[idx])$$

$$H_p = H_{inter} \odot \tilde{y}$$

DiffPool

Downsample by clustering the nodes using GNN



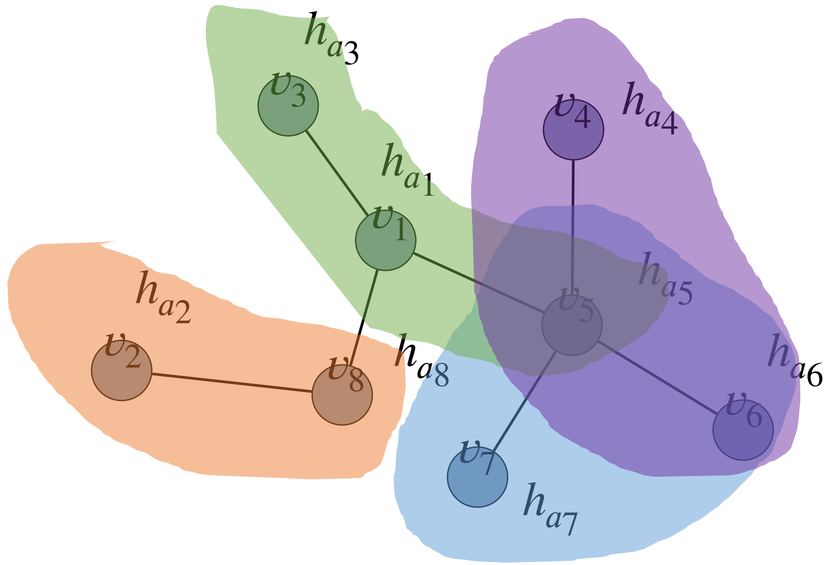
$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

Hierarchical Graph Representation Learning with Differentiable Pooling. NeurIPS 2018.

DiffPool

Downsample by clustering the nodes using GNN



$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

Filter1:

Generate a soft-assign matrix

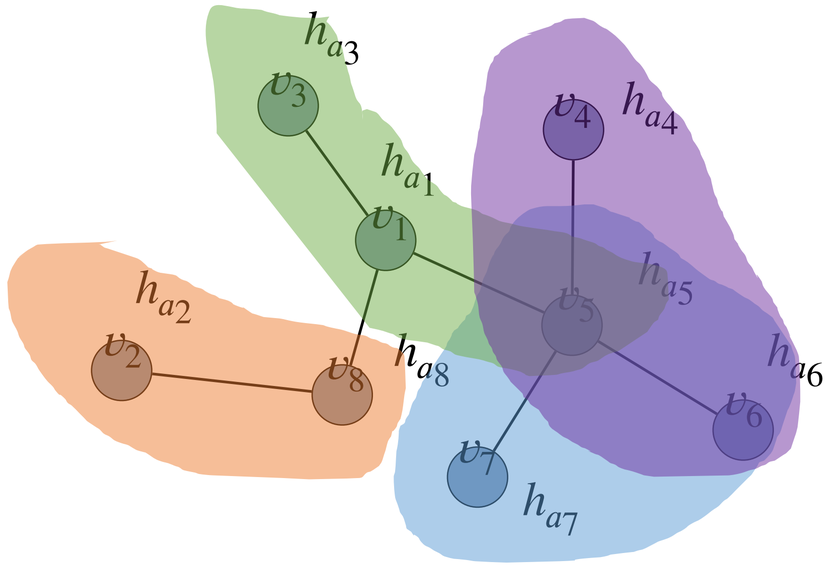
2 filters

$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H}_a \in \mathbb{R}^{n \times n_p}$$

DiffPool

Downsample by clustering the nodes using GNN



$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

Filter1:

Generate a soft-assign matrix

Filter2:

Generate new features

2 filters

$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

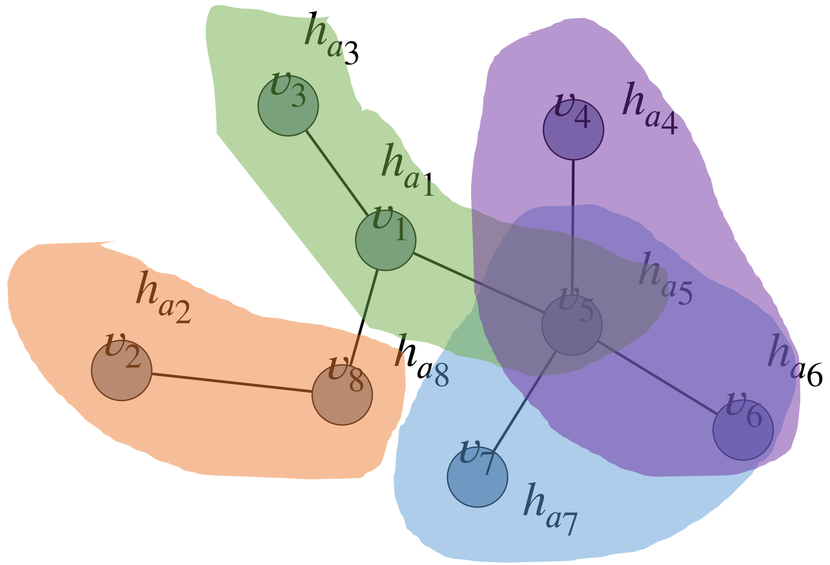
$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H}_a \in \mathbb{R}^{n \times n_p}$$

$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H}_f \in \mathbb{R}^{n \times d_{new}}$$

DiffPool

Downsample by clustering the nodes using GNN



$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

Generated soft-assign matrix $\mathbf{H}_a \in \mathbb{R}^{n \times n_p}$

Generated new features $\mathbf{H}_f \in \mathbb{R}^{n \times d_{new}}$

After we get H_a, H_f, we calculate new adjacent matrix A_p and new features H_p (basically for each community, it is a weighted sum of the nodes' features assigned to it)

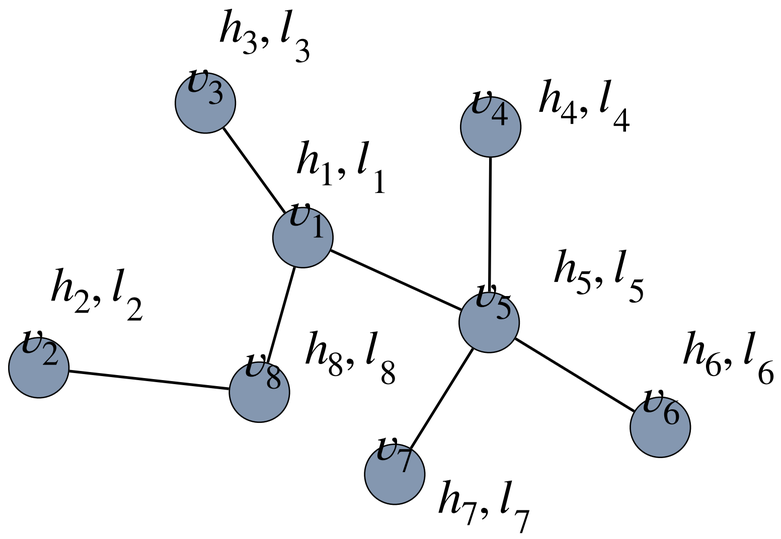
Generate A_p

$$\mathbf{A}_p = \mathbf{H}_a^T \mathbf{A} \mathbf{H}_a$$

Generate H_p

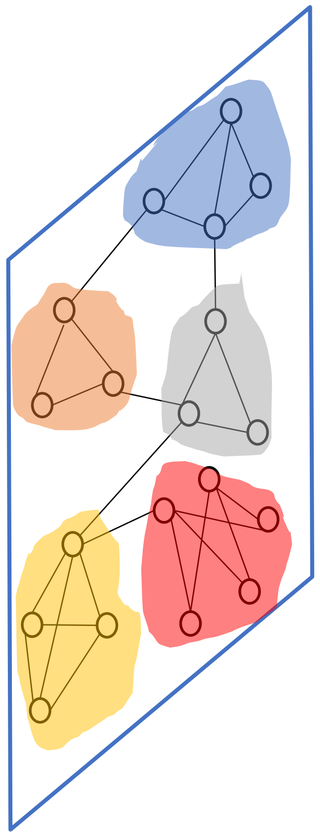
$$\mathbf{H}_p = \mathbf{H}_a^T \mathbf{H}_f$$

Eigenpooling



$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$



Learn A_p using clustering methods

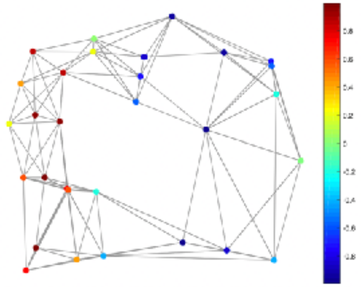
Focus on learning better H_p

Capture both feature and graph structure

Graph Convolutional Networks with EigenPooling. KDD 2019.

Going Back to Graph Spectral Theory

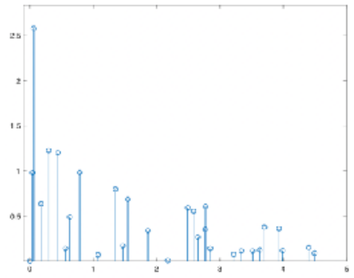
Recall:



Spatial domain: f

$$\hat{f} = U^T f$$

Decompose signal f

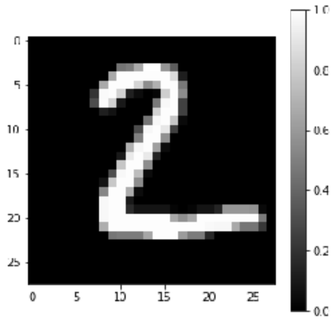


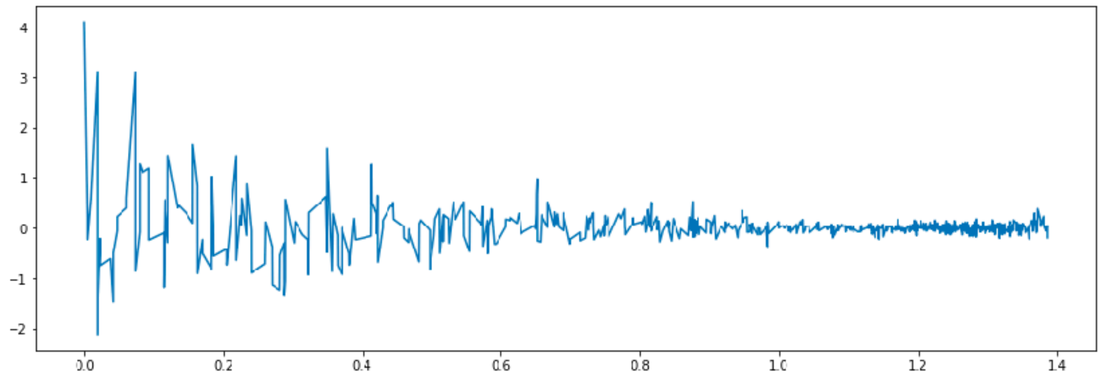
Spectral domain: $\hat{f}$

$$\mathbf{f} = \hat{f}_0 u_0 + \hat{f}_1 u_1 + \ldots \hat{f}_{N-1} u_{N-1}$$

Going Back to Graph Spectral Theory

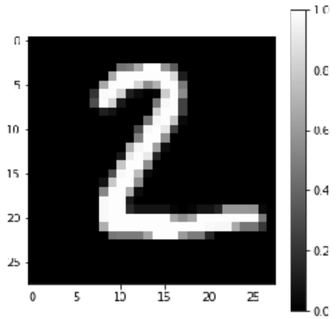
Do we need all the coefficients to reconstruct a “good” signal?

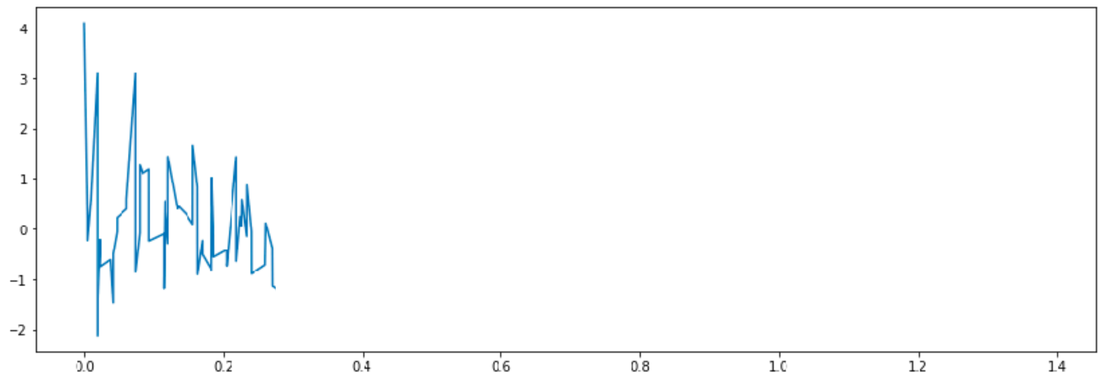




Going Back to Graph Spectral Theory

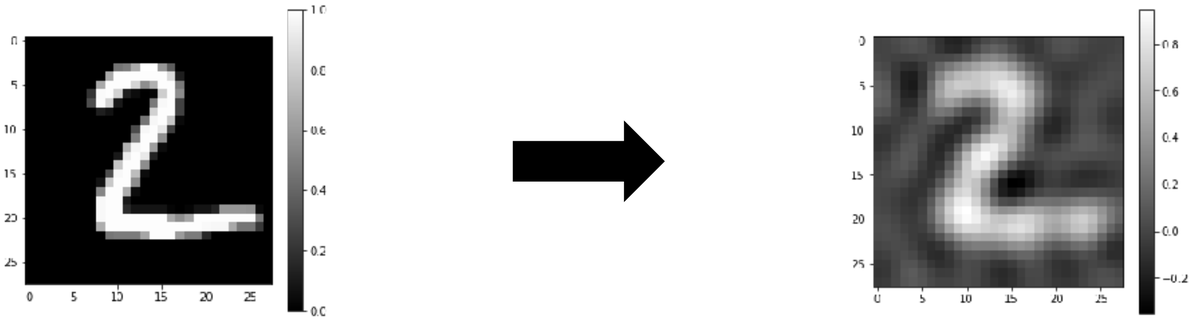
Do we need all the coefficients to reconstruct a “good” signal?

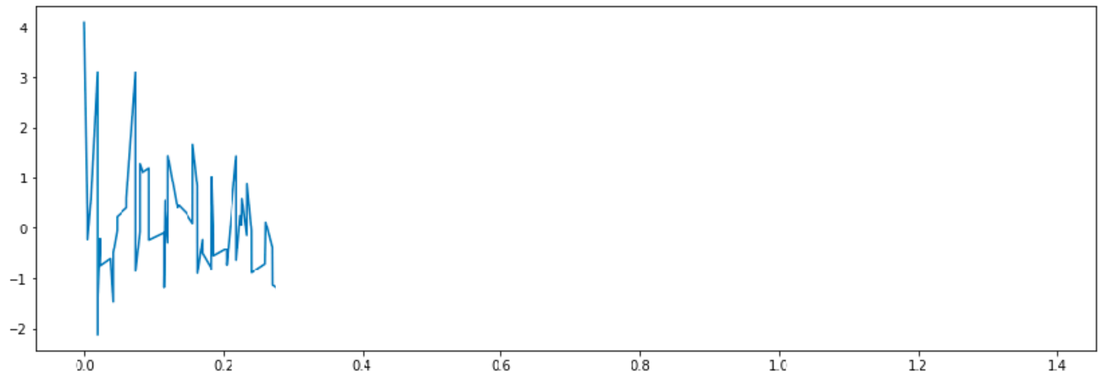




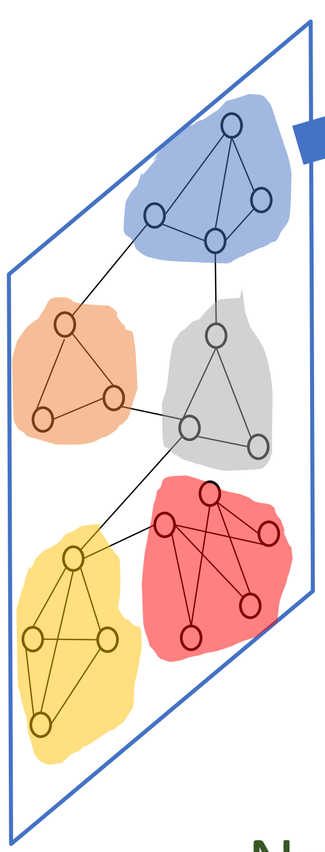
Going Back to Graph Spectral Theory

Do we need all the coefficients to reconstruct a “good” signal?





Eigenpooling: Truncated Fourier Coefficients



Eigenvectors (Fourier Modes) of the subgraph

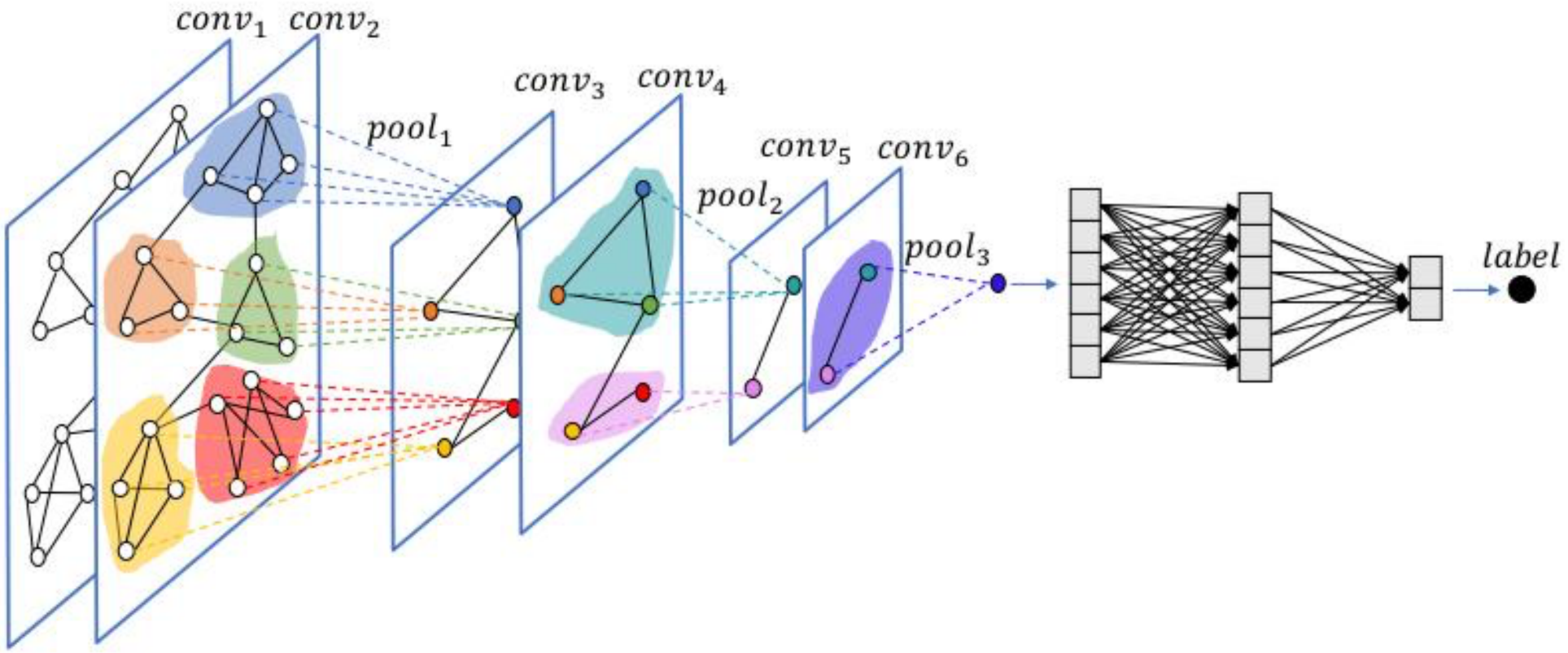
GFT

Fourier coefficients

Truncated Fourier coefficients

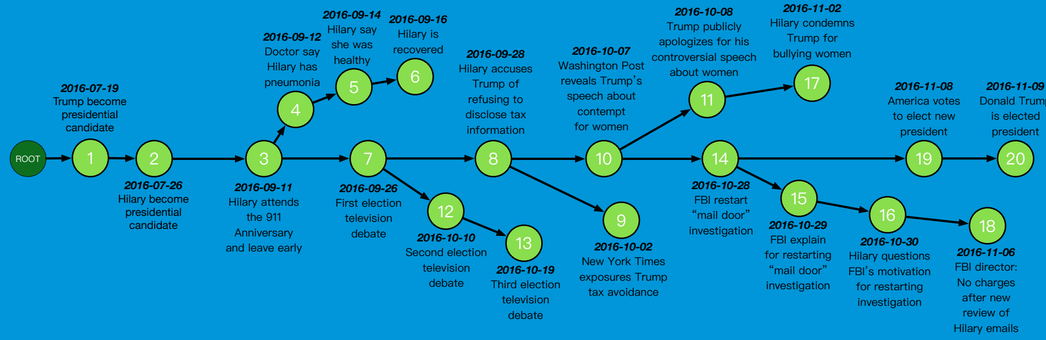
New features for the subgraph (a node in the smaller graph)

Eigenpooling: Truncated Fourier Coefficients



An illustrative example of the general framework

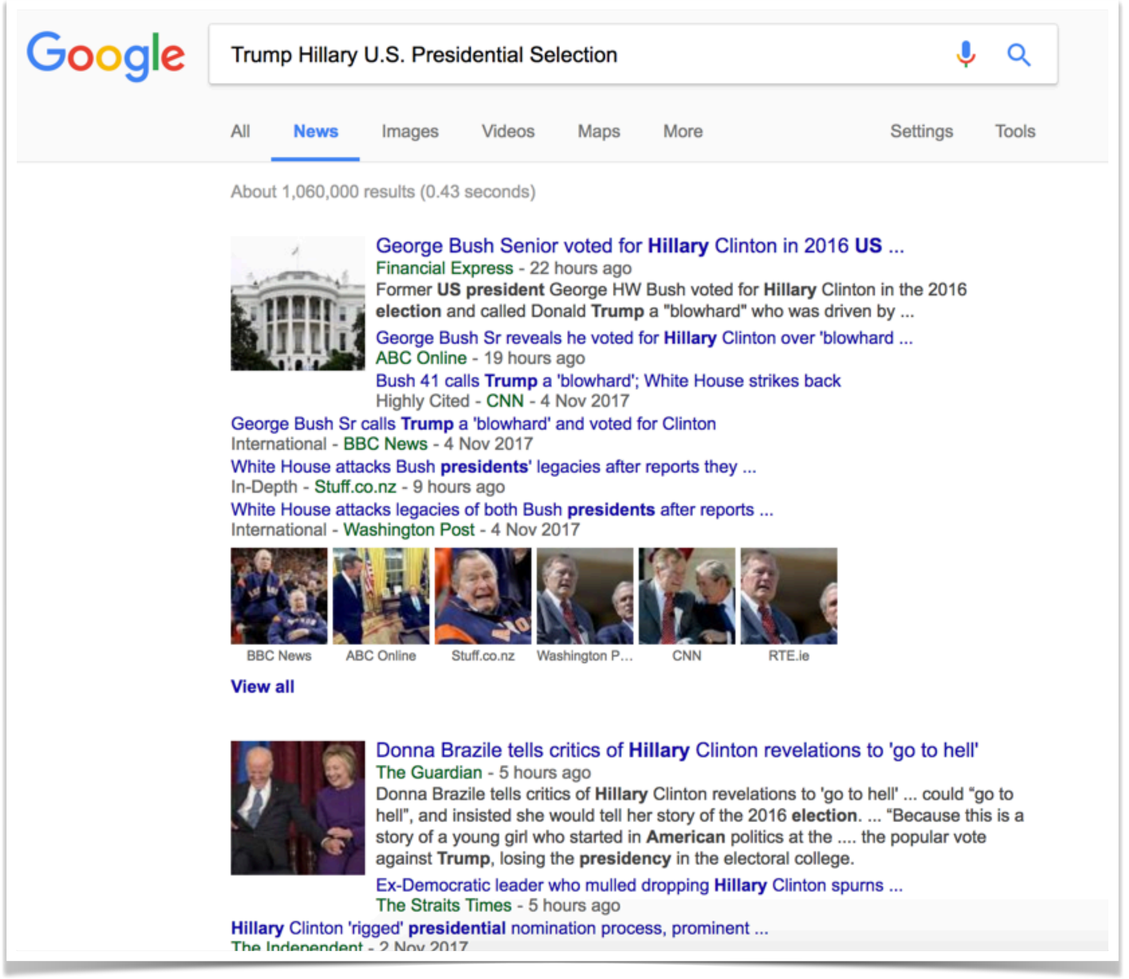
Text Clustering:
Story Forest System



Information Explosion

Reading vs. Browsing

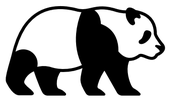
News Reading: Search Engines



News Reading: Feed Stream

Disadvantages of existing systems

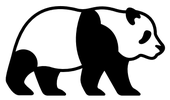
- Messed document lists
- Extremely fine-grained (articles)
- Redundant useless information
- Unstructured information



How We Remember Information

Event: something revolve around one or a group of specific persons (or entities) and happen at certain place during specific time .
Examples: Trump becomes a candidate, The first game between Kejie and AlphaGo

Story: multiple events that interdependent and evolve by time form a story.
Examples: 2016 U.S. Presidential Election, Kejie VS AlphaGo



How We Remember Information

Event: something revolve around one or a group of specific persons (or entities) and happen at certain place during specific time .
Examples: Trump becomes a candidate, The first game between Kejie and AlphaGo

Story: multiple events that interdependent and evolve by time form a story.
Examples: 2016 U.S. Presidential Election, Kejie VS AlphaGo

The smallest granularity of memory: event

Why Event Matters

Tags we have

 # Category tags
 # Automotive Technology

 # Entity tags
 # Tesla

Tags we don't have

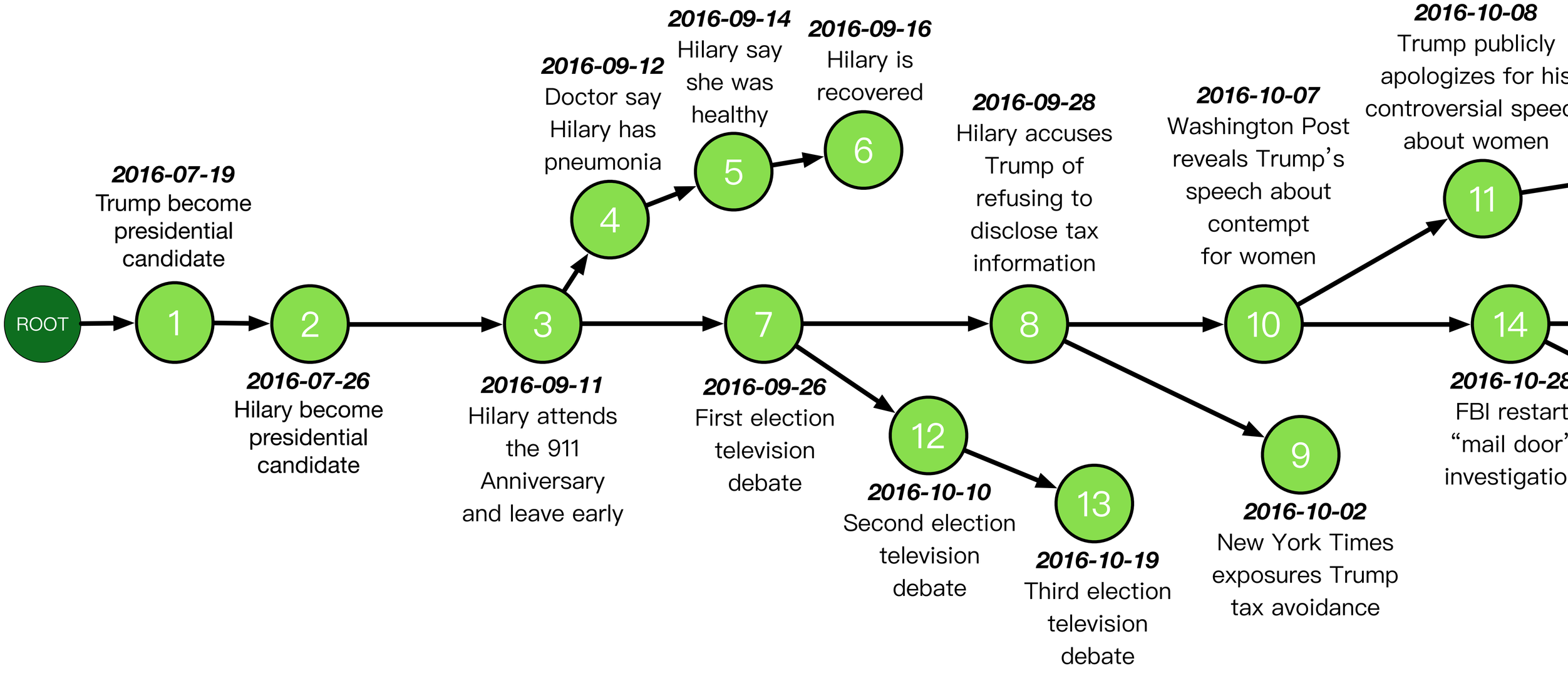
 # Event tags
 # Tesla launches new model X

Title translation

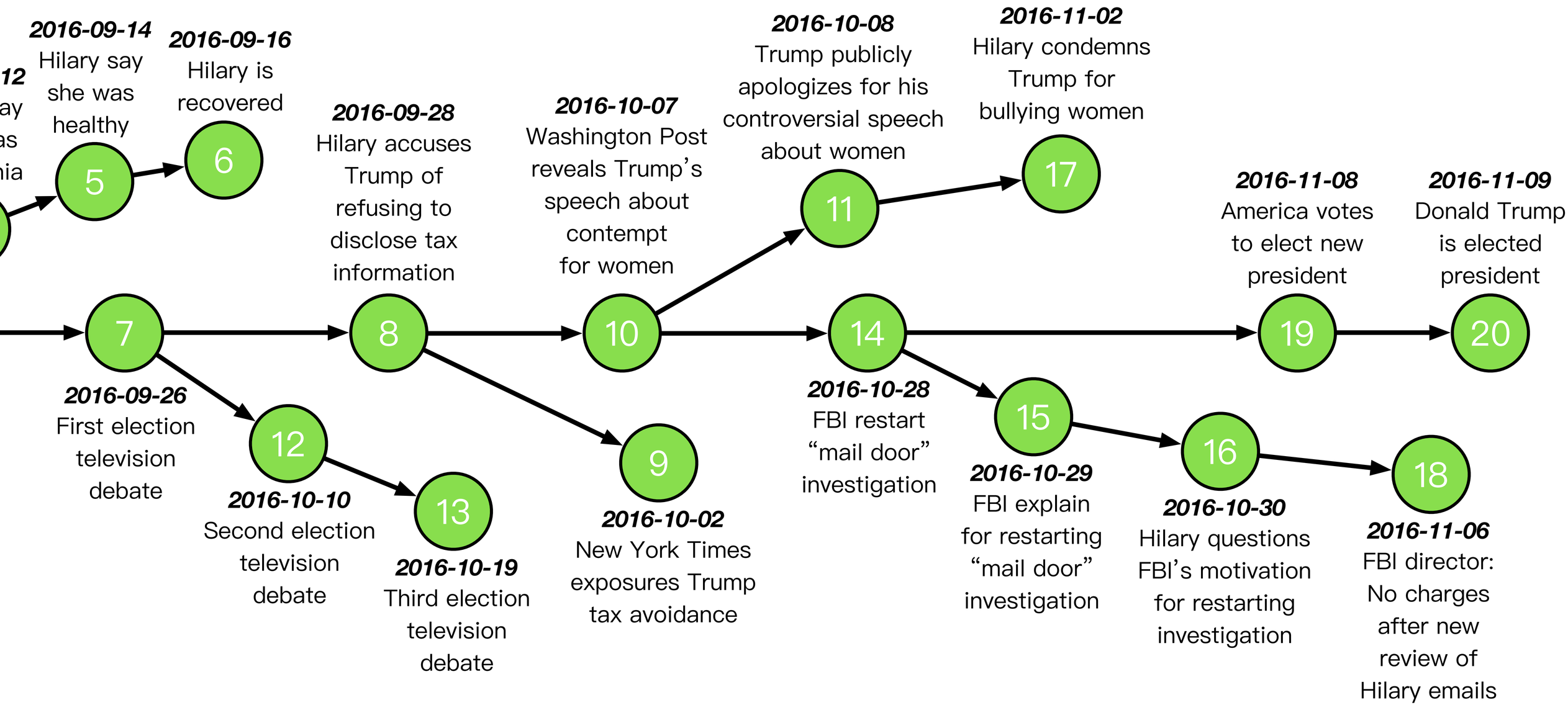
Tesla: The most conscientious pricing of imported brands turned out to be it?

7.5% articles with event tags account for **40%** of the user traffic

A Better Way to Organize Information



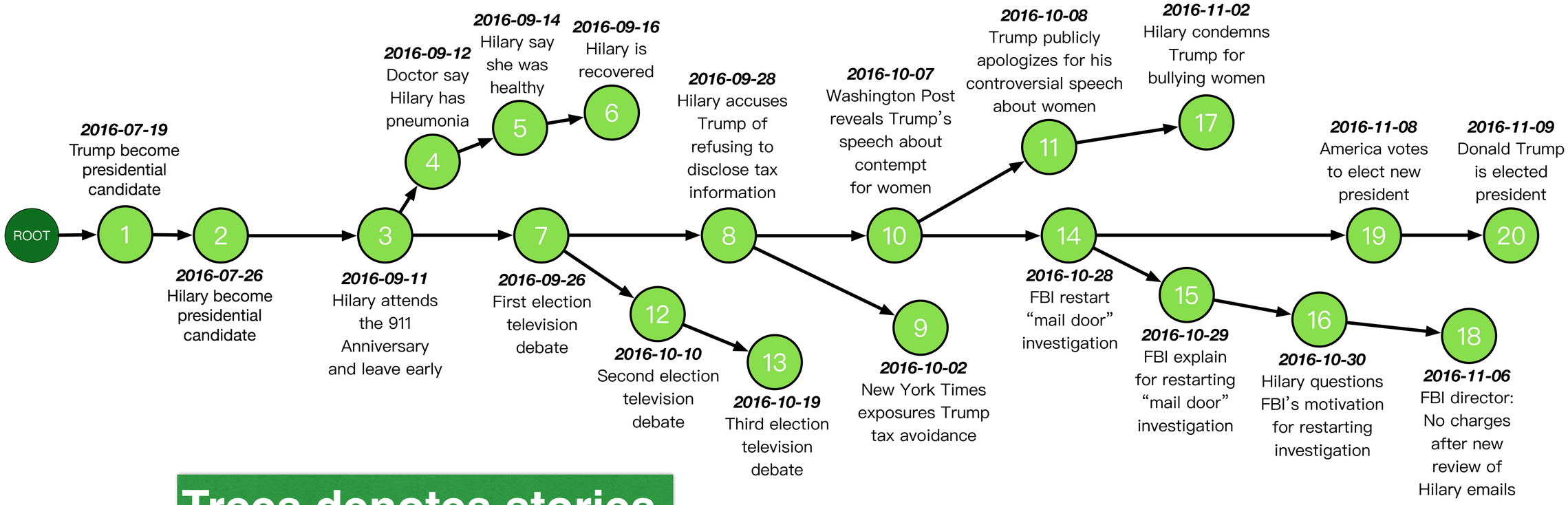
A Better Way to Organize Information



Reinvent
information platform
that matches human habits

Story Forest

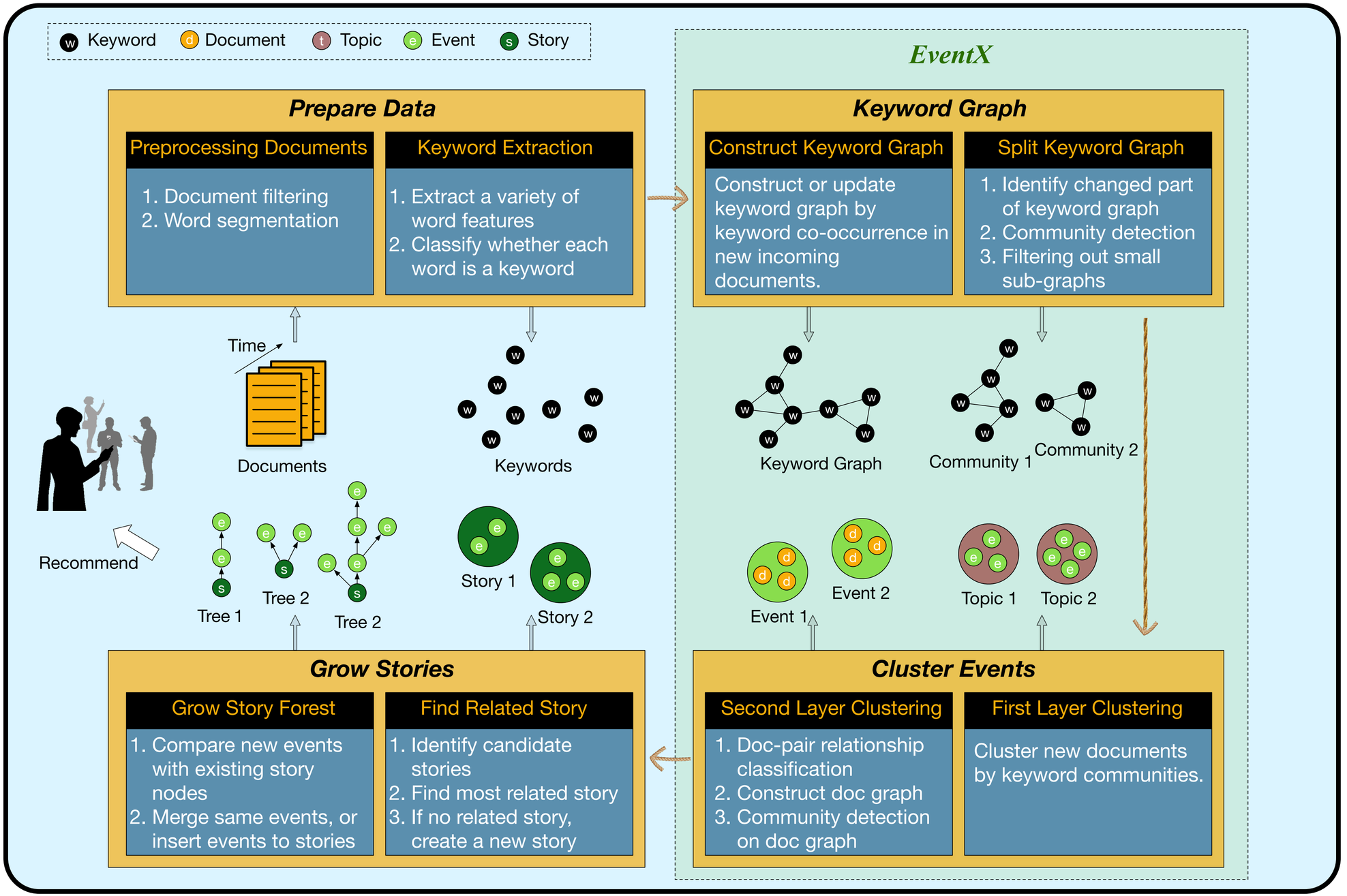
Detect events automatically from massive news articles



Trees denotes stories, nodes denotes events

Edges in the tree denotes events evolving relationship

Story Forest System



Preprocessing

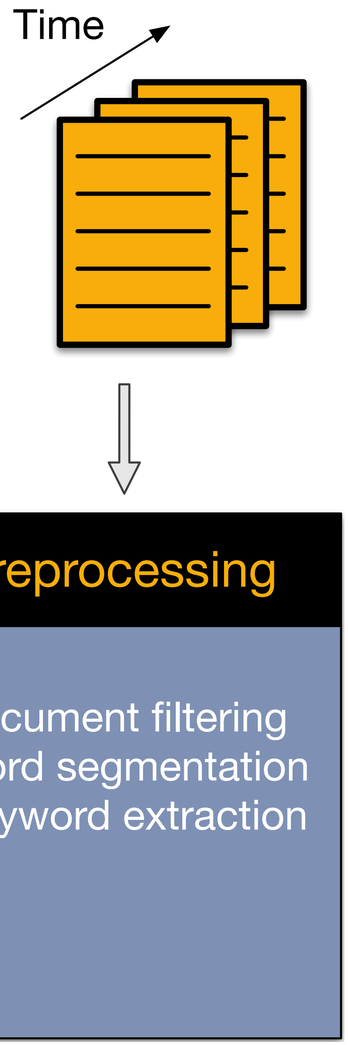
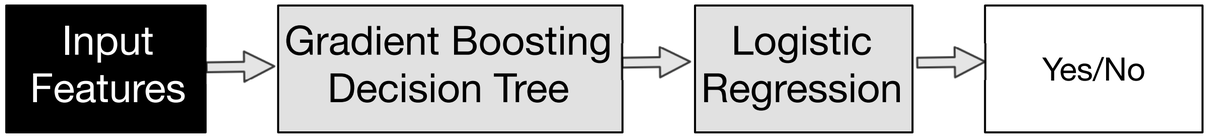
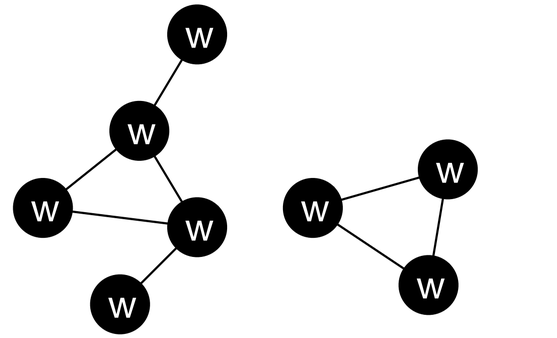


Table 1: Features for the classifier to extract keywords.

Type	Features
Word feature	Named entity or not, location name or not, contains angle brackets or not.
Structural feature	TFIDF, 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

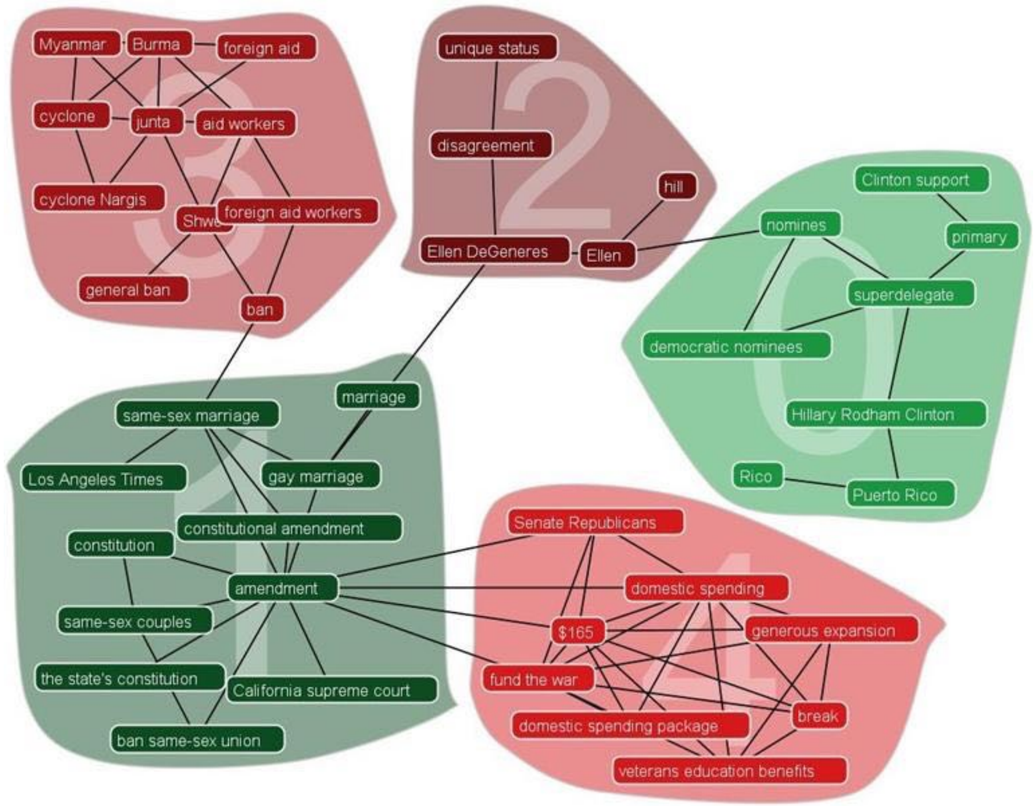


Keyword Graph

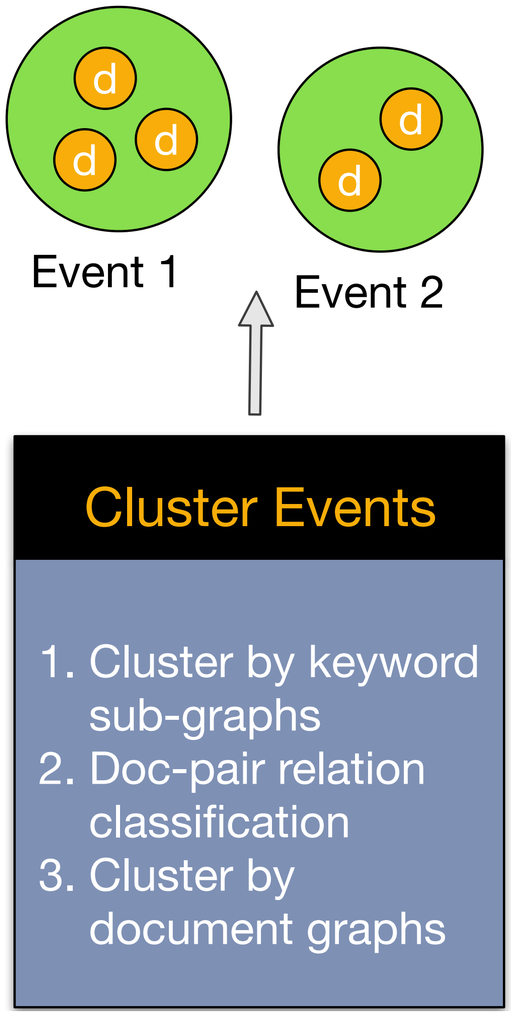


Keyword Graph

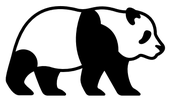
1. Construct keyword graph
2. Community detection
3. Filtering out small sub-graphs



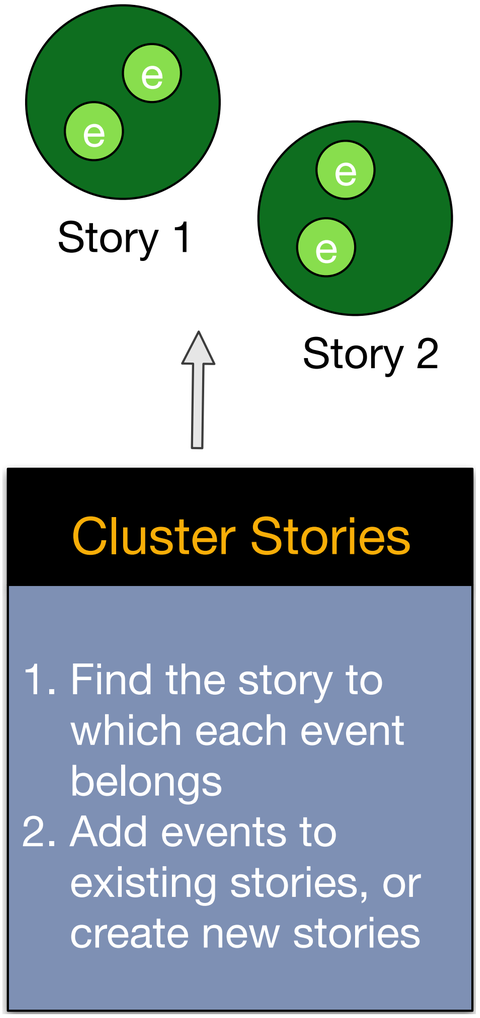
Cluster Events



- Cluster by **Keyword Graph**.
- Extract **doc-pair features**: title similarity measures, content similarity measures, news category.
- Train an **SVM classifier**: input two documents features, output same event or not.
- Community detection on **Document Graph**

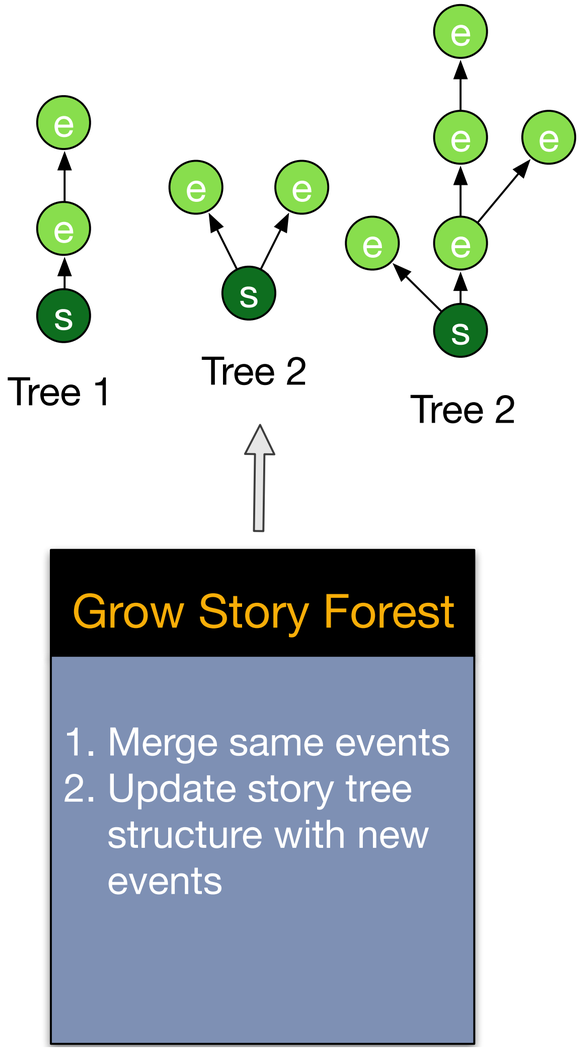


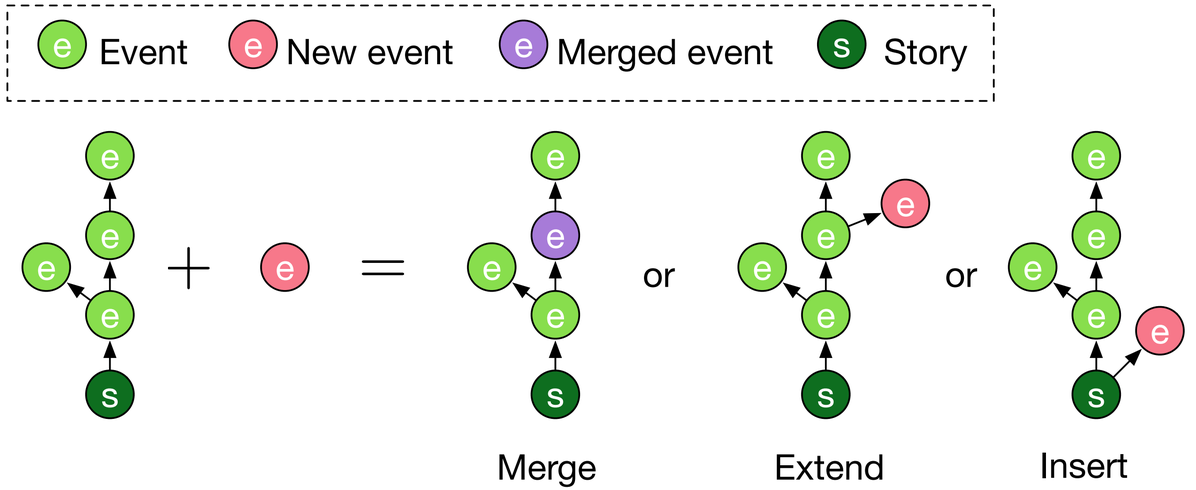
Cluster Stories



Story: multiple events that interdependent and evolve by time form a story.

Story Structure Generation



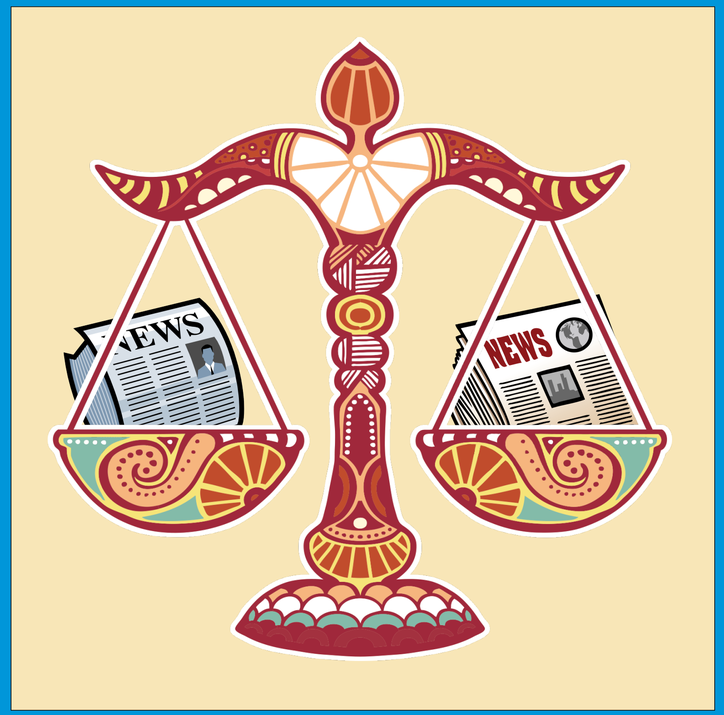


Choose best location to insert.

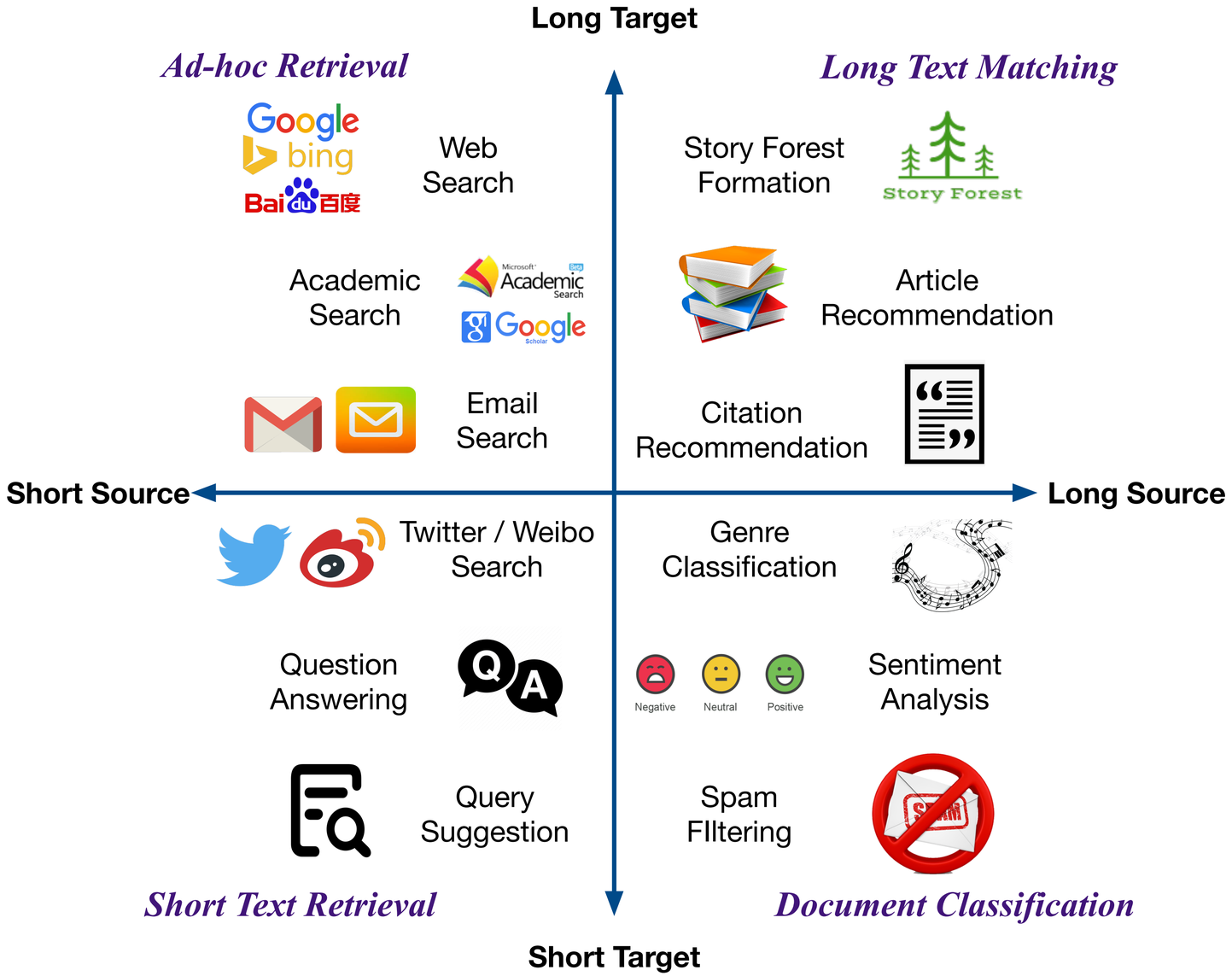
Applied to Tencent QQ browser hot topic list

Hawking public PhD thesis

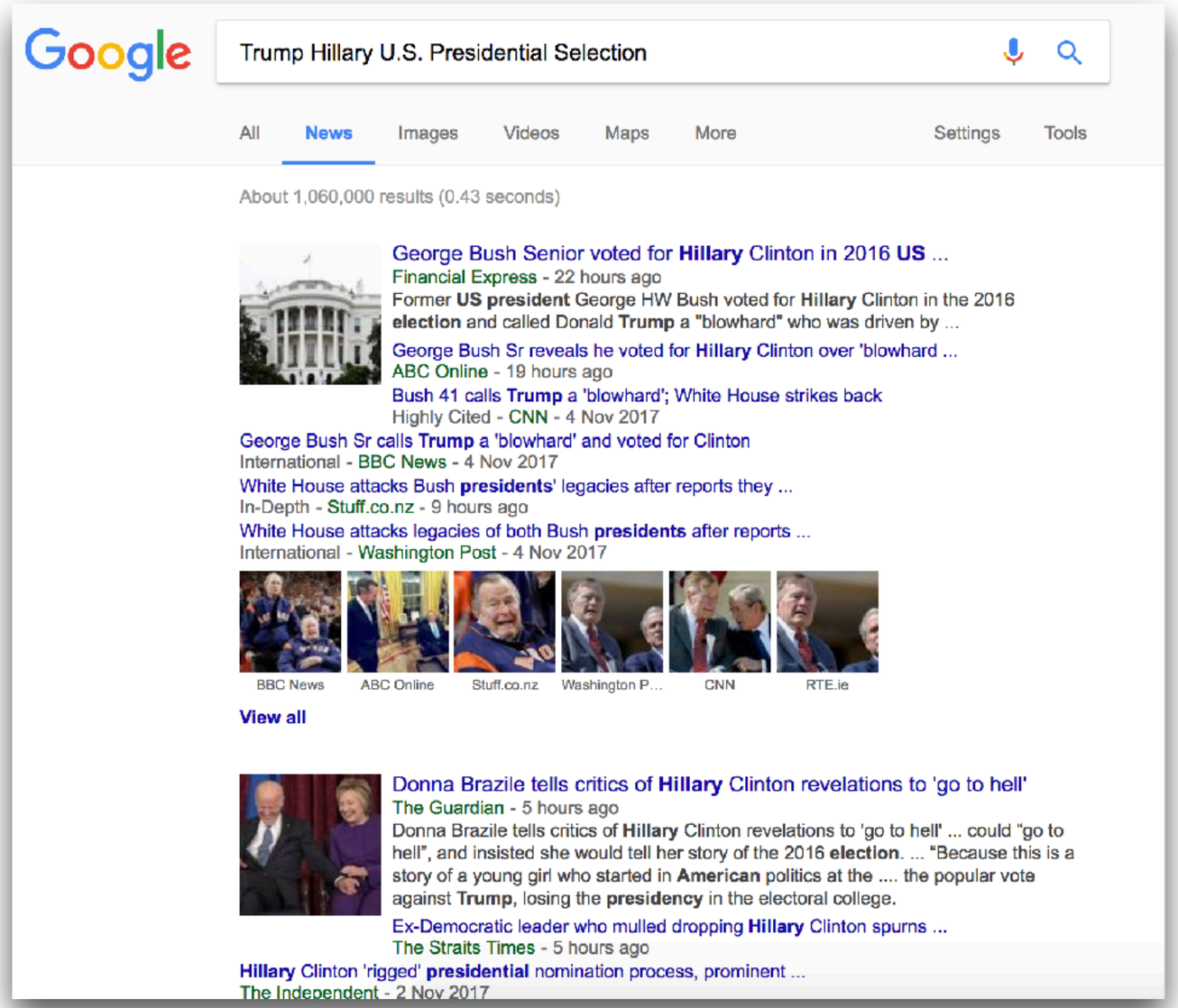
Text Matching: Graph Neural Networks and Long Document Matching



Text Matching Tasks



Why Long Document Matching



Information Explosion

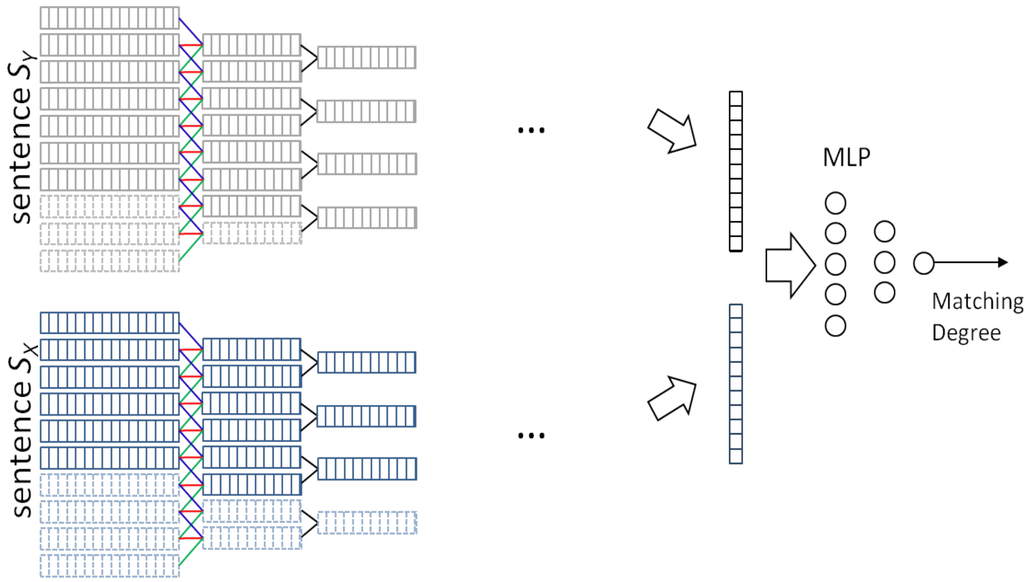
Why Long Document Matching

Identify the relationship between documents

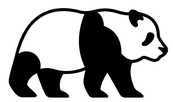
Same event?
Related events?

Existing Approaches for Text Matching

Representation-based (ARC-I)

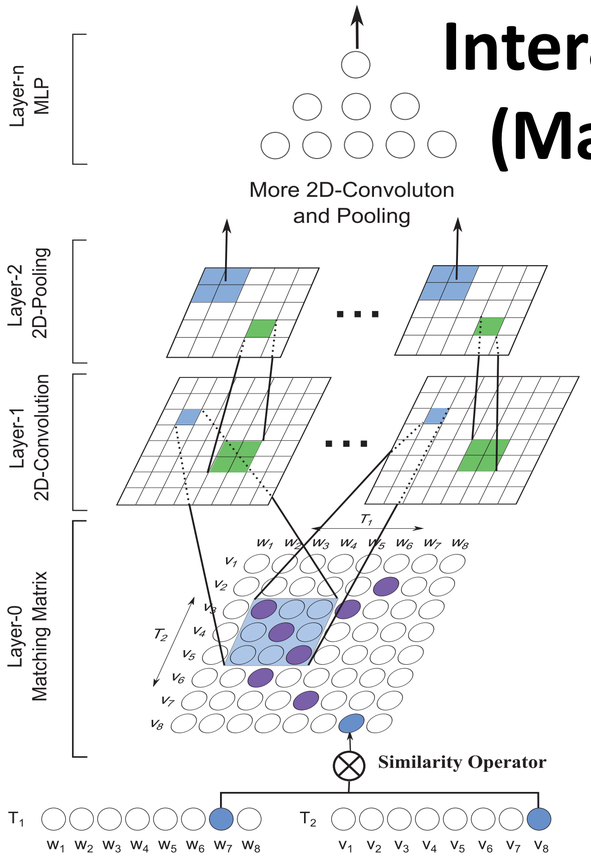


Y. Bengio. Learning deep architectures for ai. Found. Trends Mach. Learn., 2(1):1–127, 2009.



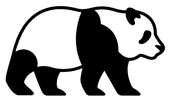
Existing Approaches for Text Matching

Interaction-based (MatchPyramid)



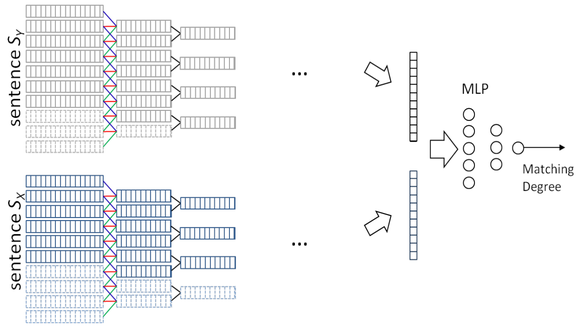
An overview of MatchPyramid on Text Matching.

Pang, Liang, et al. "Text Matching as Image Recognition." *AAAI*. 2016.



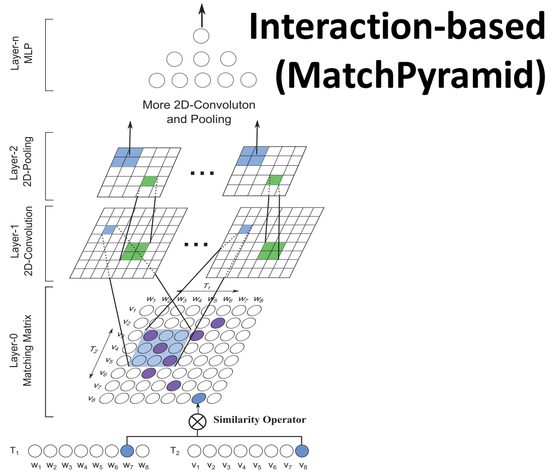
Not Suitable for Long Document Matching

Representation-based (ARC-I)



Y. Bengio. Learning deep architectures for ai. Found. Trends Mach. Learn., 2(1):1–127, 2009.

Interaction-based (MatchPyramid)

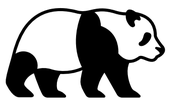


An overview of MatchPyramid on Text Matching.

Pang, Liang, et al. "Text Matching as Image Recognition." *AAAI*. 2016.

Limitations

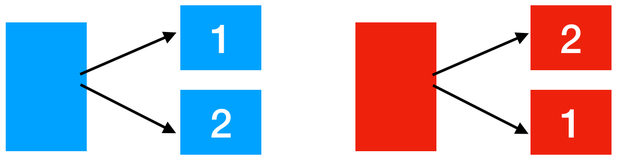
- ✦ Hard to encode
- ✦ Flexible order
- ✦ Time complexity



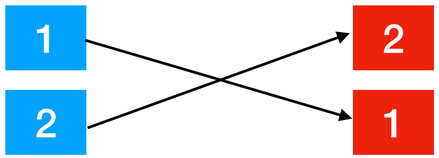
Divide-and-Conquer

Our strategies

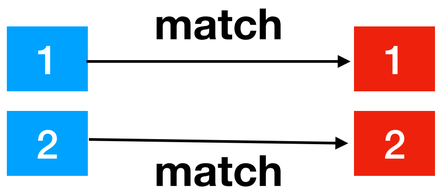
- Divide



- Align

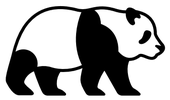


- Distributed match



Limitations

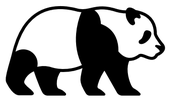
- Hard to encode
- Flexible order
- Time complexity



Decompose a Document

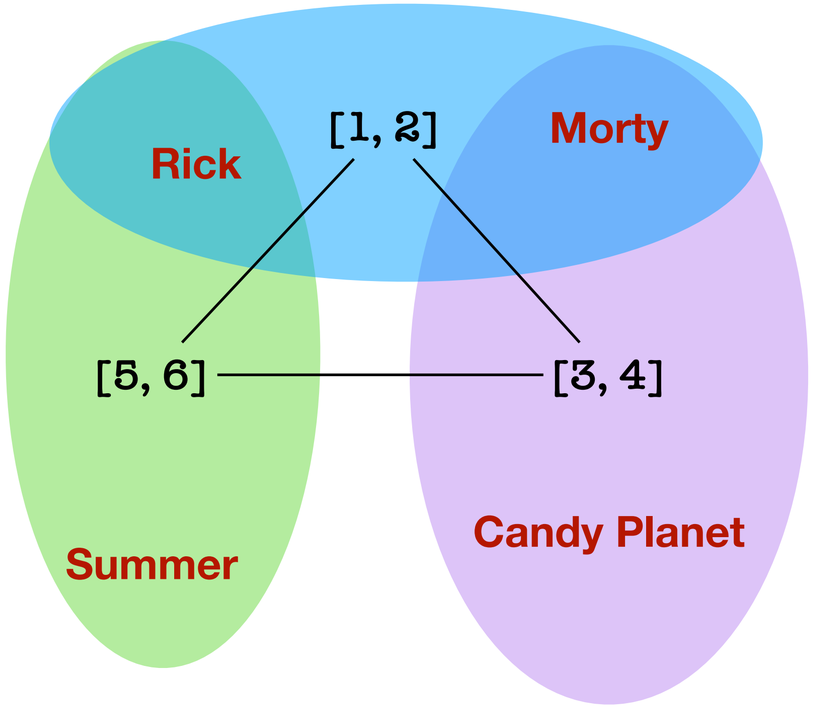
[1] Rick asks Morty to travel with him in the universe.
[2] Morty doesn't want to go as Rick always brings him dangerous experiences.
[3] However, the destination of this journey is the Candy Planet, which is a fascinating place that attracts Morty.
[4] The planet is full of delicious candies.
[5] Summer wishes to travel with Rick.
[6] However, Rick doesn't like to travel with Summer.

.



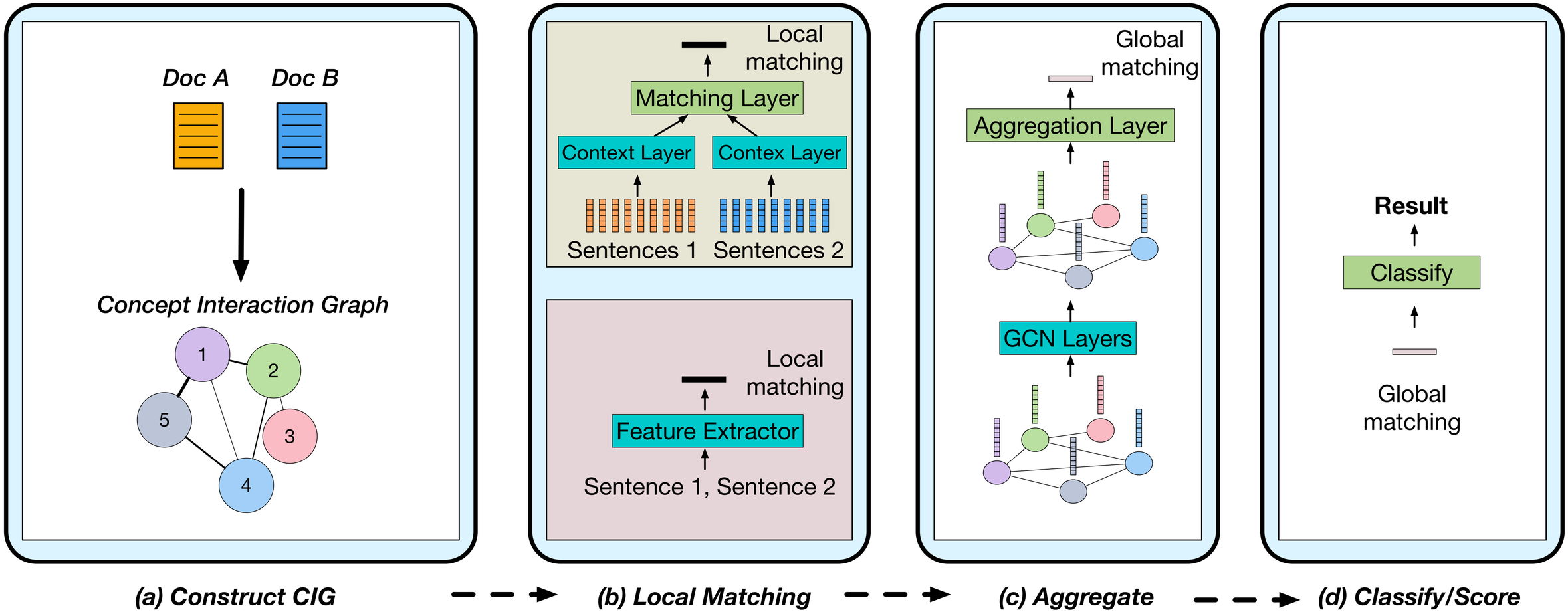
Concept Interaction Graph

[1] Rick asks Morty to travel with him in the universe.
[2] Morty doesn't want to go as Rick always brings him dangerous experiences.
[3] However, the destination of this journey is the Candy Planet, which is a fascinating place that attracts Morty.
[4] The planet is full of delicious candies.
[5] Summer wishes to travel with Rick.
[6] However, Rick doesn't like to travel with Summer.



STEP 1: Extract keywords
STEP 2: Group keywords
STEP 3: Assign sentences
STEP 4: Construct edges

Graph Decomposition for Document Matching



[Liu et al., ACL 2019]

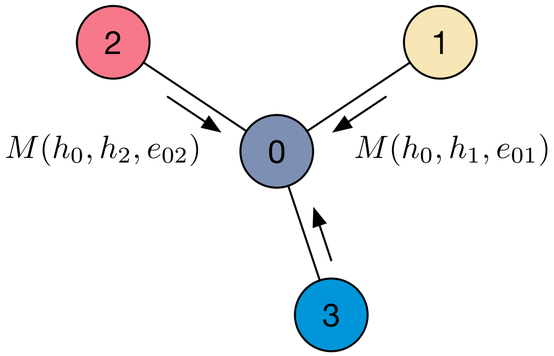
Graph Convolutional Network

Message Passing: a General Framework

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

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

$$\hat{y} = R(\{h_v^T | v \in G\}).$$



M_l : Message function,

U_l : Update function,

e_{vw} : Edge features,

R : Readout function.

Graph Convolutional Network

$$h_v^{l+1} = \sigma\left(\frac{1}{|\mathcal{N}(v)|} \sum_{w \in N(v)} W^l h_w^l + b\right)$$

Normalization

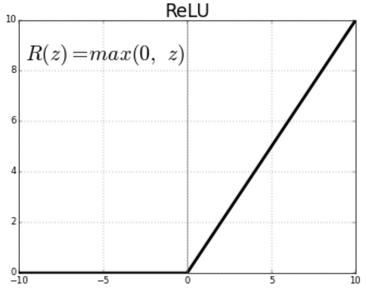
Neighborhood aggregation

Non-Linearity

Filter matrix (model parameter)

bias (model parameter)

node feature

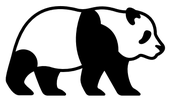


Experiments

Baselines	CNSE Acc	CNSE F1	CNSS Acc	CNSS F1	Our models	CNSE Acc	CNSE F1	CNSS Acc	CNSS 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

✦ **Graph Representation:** greatly improves performance. (IX vs. XI)
(+4% Acc, F1)

✦ **Graph Convolution**: greatly improves performance. (XIV vs. XV)
(+10% Acc, F1)



Other Works on Text Matching

Sentence Pair Matching **Short-Short** **WWW18**

Matching Natural Language Sentences with Hierarchical Sentence Factorization

Bang Liu[1], Ting Zhang[1], Fred X. Han[1], Di Niu[1], Kunfeng Lai[2], Yu Xu[2]
[1]University of Alberta, Edmonton, AB, Canada
[2]Mobile Internet Group, Tencent, Shenzhen, China

- **Hierarchical factorization**
- **Multi-granularity matching**

Query-Doc Matching **Short-Long** **CIKM18**

Multiresolution Graph Attention Networks for Relevance Matching

Ting Zhang[1], Bang Liu[1], Di Niu[1], Kunfeng Lai[2], Yu Xu[2]
[1]University of Alberta, Edmonton, AB, Canada
[2]Mobile Internet Group, Tencent, Shenzhen, China

- **Keyword graph**
- **Match and aggregate**

Todo

- **Reading assignment:**
  Neural Message Passing for Quantum Chemistry
  https://arxiv.org/pdf/1704.01212.pdf (March 25th, 2022 23:59pm EST timezone)
- **Suggested readings**:
  - Story Forest: Extracting Events and Telling Stories from Breaking News: https://dl.acm.org/doi/10.1145/3377939
  - Semi-Supervised Classification with Graph Convolutional Networks: https://arxiv.org/abs/1609.02907
  - Modeling Relational Data with Graph Convolutional Networks: https://arxiv.org/pdf/1703.06103.pdf

Next lecture: Invited talk

References

1. http://cse.msu.edu/~mayao4/tutorials/aaai2021/

Thanks! Q&A

Bang Liu
Email: bang.liu@umontreal.ca
Homepage: http://www-labs.iro.umontreal.ca/~liubang/