The 35th Canadian Conference on Artificial Intelligence
DOI: 0
Canadian AI 2022
30 May–3 June 2022, Toronto

A working model for textual Membership Query Synthesis

Frédéric Piedboeuf[†,*], Philippe Langlais[†]
[†] RALI, Université de Montréal, Montréal

Abstract

Membership Query Synthesis (MQS) is an active learning paradigm in which one labels generated artificial examples instead of genuine ones to extend a dataset. Despite prodigious advances in the power of generative models, an essential component of MQS, the field stays severely under-studied, especially in the textual domain. We found only one other paper, which selects examples in a latent space close to the decision boundary and shows good results on a curated dataset of short sentences. We show that this performs poorly when used on a real dataset. We propose and report better results than random selection of unlabelled genuine data with random generation of artificial data from a variational auto-encoder coupled with a simple set of filtering mechanisms. This provides an improvement of 31.1% over the previous MQS state-of-the-art on the SST-2 dataset, and of 2.7% over random active learning. To the best of our knowledge, this is the first time MQS is reported to work on a textual task with no constraint on the size of the input sentences.

Keywords: Membership Query Synthesis, Active Learning, Variational Auto-Encoders, Generative Models

1. Introduction

With the increasing power of deep neural networks and the consequent demand for large datasets, data acquisition has become a central issue in modern machine learning [1]. However, data can be costly to acquire [2], especially when considering specialized data such as medical ones [3] or rare languages for machine translation [4]. To alleviate this problem, solutions have been proposed to create good performing models with minimal or no annotated training material, such as unsupervised learning [5], semi-supervised learning [6], or active learning [7]. In this paper, we take a close look at active learning, and more specifically at a subset of it called membership query synthesis (MQS).

Active learning (AL) is a learning paradigm in which the data points are manually labelled as the algorithm is training, by selecting at each iteration the most likely relevant data points to annotate. It is generally categorized in three broad families of techniques: pool-based AL in which one has access to a pool of unlabelled data, stream-based AL where one receives data points one at the time and must decide to annotate them or not, and MQS in which one generates the most informative examples to annotate. MQS is of great theoretical interest since it allows a full exploration of the classification space, while active learning is restricted to the unlabelled data distribution, which may be biased or fail to cover adequately the input space [8]. Due to the current limitations of generative models and therefore the lesser quality of sentences when compared to genuine ones, we do not expect MQS to beat pool-based AL. We include it in this paper as an upper bound on the performance of MQS.

Because of the difficulty of implementing a MQS system (see Section 2), as well as the limits imposed by the generative models, there have been few studies on MQS, especially on textual data. In fact, positive results have so far been reported only on a curated dataset where long sentences (longer than 15 words) were removed, and additional data was added from other datasets [9]. We show in this paper that previous state-of-the-art on MQS for sentence classification does not work well when limitations on the maximal length of the sentences are removed and no additional data is added, a situation which rarely fits the annotation scenario in the real world. In this paper, we propose a simple random sampling strategy coupled with some filtering of the generated examples. We show that our MQS systems work as well as a naive random pool-based AL system (and in some

[*] frederic.piedboeuf@umontreal.ca

case surpasses it), thus establishing a new state-of-the-art in MQS for sentence classification. We overall test three generators, namely a variational auto-encoder (VAE), a conditional VAE (CVAE), and a semi-supervised VAE (SSVAE), with various seed sizes of the labelled dataset and multiples strategies for selecting the next point to label. Our contributions are:

(1) Showing that MQS on a realistic textual classification task works,
(2) Showing that MQS on textual data can work better than random pool-based AL,
(3) Providing a new state-of-the-art on textual MQS,
(4) Providing analysis and ideas for the future of the field.

The paper is organized as follows. In section 2, we describe MQS and the related works. We present the VAE model, which is central to our work, in section 3. Then, sections 4 and 5 present the experimental protocol and the results. Finally, we analyze our results in section 6 and conclude in section 7.

2. Membership Query Synthesis and Related Work

In pool-based AL, we have an unlabelled dataset $\mathcal{U}$ and a labelled one $\mathcal{L}$ in a feature space $\mathcal{X}$, as well as a classifier $\mathcal{F} : \mathcal{X} \rightarrow \mathcal{Y}$ trained iteratively, where at each iteration we want to select from $\mathcal{U}$ the most informative instances to label and transfer them to $\mathcal{L}$.[1] The challenge of pool-based AL is to correctly define what "most informative" means, which often is defined based on point distribution in $\mathcal{X}$ [12] or as a lack of confidence of the classifier $\mathcal{F}$ [13]. In simpler terms, the goal is to select from an unlabelled set of data the fewest number of examples to annotate in order to maximize the performance of the classifier. In a real world application, a human labeller (or oracle) $\mathcal{O}$ labels the examples, but in experiments, this is most often bypassed by using an already labelled dataset and "hiding" the labels until the example is selected to be added to $\mathcal{L}$.

MQS is a variation of that idea where, instead of selecting the most informative examples from $\mathcal{U}$, one picks the most informative points in $\mathcal{X}$ to label. This implies a generator $\mathcal{G}$ which transforms points from the feature space $\mathcal{X}$ into readable data, and which is trained on all unlabelled data $\mathcal{U}$. This also implies that we need a labeller or oracle O, since newly generated points do not come with a label. For experimentation, however, the oracle can be replaced by a classifier [14]. Two more important things that need to be taken care of are when to stop, and how many initial labelled data to use, which we denote as the *seed*. The seed is used to help the selection algorithm select the next points, and the stopping point is used to objectively compare several algorithms. In this paper, we use a labelling budget of 500, which includes the initial labelled data. This means that once we have labelled 500 data, we stop and train the model.

Because of the constraint that we need to be able to pick any points from $\mathcal{X}$ to label, most MQS systems introduce a continuous latent space $\mathbf{z}$ from which points are transformed into readable data. Overall, a MQS system totals five components, namely the generator (composed of the encoder and the decoder), the selector, the classifier (the model we want to train), and the labeller, as shown in Figure 1. Some algorithms modify slightly this pipeline. For example, the authors of [15] use a GAN on MNIST and CIFAR-10 for MQS, therefore not relying on an encoder. To find which points to label, they solve an optimization problem, projecting the decision boundary in the latent space and finding points near this decision boundary.

[1] We focus here on classification, but active learning systems have also been developed for other kinds of tasks, such as NER [10] or machine translation [11].

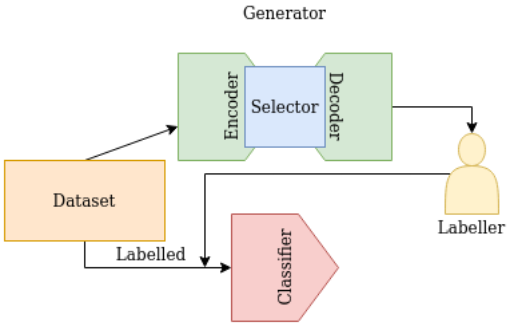

Figure 1. A typical MQS pipeline. The selector picks the next point to label based on the distribution of the dataset in the latent space $\mathbf{z}$. The decoder transforms that point into a readable sentence, which is labelled by the labeller. Because generated examples are often of lower quality than genuine ones, we do not use the generated sentences for bootstrapping the generator and only use them to train the classifier.

As mentioned, MQS for textual data has been less studied due in part to the limits of the generative models for text. The authors of [16] present a MQS algorithm, but they use it as pool-based AL method by labelling the closest point in $\mathcal{U}$ instead of the selected point. We denote this algorithm DB, or decision boundary.

The DB algorithm starts by finding a pair of points (one of each class) that should be close to the boundary decision. To do so, points at mid-distance of the centroids of the two classes are queried for a fix number of iterations, adding the new points to $\mathcal{L}$ and moving the centroids closer to the boundary decision every time. Then, starting with the two last points of opposite classes found (x_+, x_-), it iteratively: 1- generates a vector of magnitude λ midperpendicular to the two points, 2- queries this point to obtain its class y_i, 3- replaces the point of the class y_i in the pair (x_+, x_-) by this new point. This has the effect of moving along the boundary decision, sampling along both sides. The algorithm, illustrated in Figure 2, is stopped when the labelling budget is reached. In [16] they use the active version of the algorithm on several medical datasets, as well as binary versions of MNIST, and show an improvement when compared to other active learning algorithms.

The authors of [9] use the MQS version of the algorithm for directly generating and labelling the selected point, on a curated dataset of sentiment classification composed of short sentences (less than 15 words) from SAR14 and SST-2. They compare to random selection and two pool-based AL techniques (random sampling and least confidence), and report that both MQS techniques outperform AL when comparing the annotation cost. This is however a fairly unrealistic assumption in practice. For example, if we look only at the SST-2 dataset, we find that the average length of sentences when split by white spaces is 19.3, showing that the findings of [9] wouldn't necessarily apply to the full dataset.

The only other work we have found on textual MQS is the one of [14], where sentences are edited to perturb the examples and create new ones. To do so, words are replaced by semantically near substitutes, and these sentences are then labelled to see if the replacement made them change the class. While it is technically MQS, we do not compare ourselves to it in our work because the focus is not on how to best select and create a dataset from a continuous space, but rather on generating new examples when the pool of unlabelled data is very small.

3. Variational auto encoders

VAEs are stochastic auto-encoders composed of two components: an encoder $q_\phi(\mathbf{z}|\mathbf{x})$ and a decoder $p_\theta(\mathbf{x}|\mathbf{z})$. Both are represented using neural networks, with the last layer of the encoder predicting the parameters of the distribution associated with $\mathbf{z}$, often the μ and σ of a diagonal Gaussian. While training, we sample from the distribution $\mathcal{N}(\mu, \sigma)$ using the reparametrization

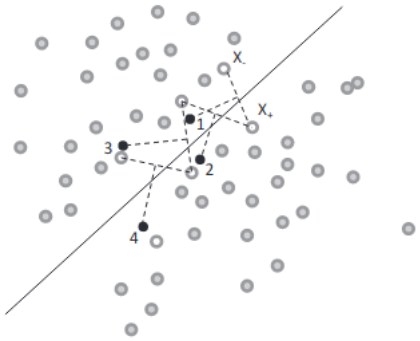

Figure 2. Selection of the next point in the latent space to transform into a sentence. The algorithm goes along the boundary decision. In [9], the point is directly queried. In [16], the nearest neighbour is queried. Image from [16].

trick [17], which samples from $\mathbf{z} = \mu + \sigma\epsilon$, where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$. This allows the gradient to flow back through the sampling step and to the encoder, while keeping the expectation and variance of $\mathbf{z}$ the same as if sampled directly. The training objective is the Evidence Lower Bound (ELBO), which is a combination of the reconstruction loss and of the KL divergence between the posterior of q and the prior.

$$\mathcal{L} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{3.1}$$

The adaptation of the VAE for textual generation consists in the use of a RNN for both the encoder and the decoder, but that comes with its own challenges [18]. The main problem is that a powerful decoder makes it easy for the VAE to collapse the latent distribution to a single point, effectively ignoring the information from the latent space. Two standard solutions to this are annealing the strength of the KL divergence in the ELBO from 0 to 1, as well as using word dropout on the decoder. The first one allows the decoder to learn to rely on the latent space before we start forcing our prior on it, and the second one forces the decoder to use the information from the latent distribution for predicting words that are masked. While these two solutions are the ones used in our paper, many more have been developed [19] and should be explored. A simple modification to this framework is the CVAE [20], where we model instead $p(\mathbf{x}|\mathbf{z}, \mathbf{c})$, conditioning on the class of the sentence (in our case positive/negative). Finally, we present a SSVAE, inspired by [21], which is a variation of the CVAE where the class is not hard, but a distribution computed by a classifier. In [21] and in traditional SSVAEs, the real label is given if available, and the label predicted by a classifier is used otherwise. However, here we want our generator to learn to generate given the confidence of the classifier, so we always feed it the probability distribution instead.

4. Experimental Protocol

In this section, we present the protocol and systems we use for our experiments. As mentioned, the MQS pipeline contains five components, namely the generator (composed of the encoder and of the decoder), the selector, the classifier, and the labeller.

The selector is the algorithm in charge of selecting the next point to generate in the latent space. We test three selection algorithms: RANDOM (select randomly from the prior), DB (decision boundary - algorithm from [9], only for the VAE), and CONFIDENCE (generates examples that have low confidence for the classifier, only for the SSVAE). We report the hyperparameters used for the variational algorithms in Table 1.

We classify using BERT [22], and more specifically *bert-small*, which we finetune for 4 epochs on $\mathcal{L}$ once the dataset is assembled. For the SSVAE, since it uses the classifier, we finetune it anew at

$x0$	15
k	0.0025
Batch Size	64
Latent Size	512
Hidden Size	512
Nb Epoch	30
Dropout	0.3
Word dropout	0.6
Nb layers GRU	1

Table 1. Hyperparameters used for the variational algorithms. We use a sigmoid function to anneal the KL divergence, with $x0$ referring to the number of epochs where the KL strength is 0.5, and k controlling the strength of the slope.

each of the 4 iterations, and once more at the end. That way, each method is evaluated in a uniform way with 4 epochs of finetuning.

Because labelling becomes rapidly expensive when running experiments, we replace what would normally be a human by a classifier, as was done in [14]. We pick BERT due to the excellent performances it has shown in recent years, but in order to get it closer to human performance, we fine-tune it on all labelled data from both the train and test set. In Section 6, we consider the difference this makes vs both human performance and a BERT trained only on the training set. We denote this BERT-FULL to differentiate it from the BERT classifier, which has only access to $\mathcal{L}$.

We get a human/BERT-FULL agreement of 92% when testing on 100 sentences generated by the SSVAE, which means we are here in a case of noisy labelling. Since we want to discard examples that are neither positive nor negative, we filter out examples that have under 70% (threshold found experimentally) of certitude by the labeller, but still count those in the labelling budget. This would correspond to a situation where the human labelling the dataset would look at the sentence and reject it because of its bad quality. We furthermore apply a filter that checks if the generated example is already in the labelled dataset, in which case it is not presented to the labeller and not counted in the labelling budget.

We use the SST-2 dataset [23], which is a sentiment classification dataset of movie reviews. The dataset is composed of 6920 training examples averaging 19.3 words each, with 3610 positive examples, the rest being negative. We conduct three sets of experiments with different seed sizes: 10, 50, and 100.

5. **Results**

In Table 2, we show the results of our experiments with the various generators and selection strategies, averaged over six runs and with different random seeds and initial data points. We run two baselines: BERT fine-tuned with only the seed, without annotating any further data, as well as random sampling with a simple auto-encoder (AE). We also compare ourselves to two active learning baselines, which provide an experimental upper bound on the results. As mentioned, we do not hope to surpass the active learning algorithms, since the quality of the sentences generated by our VAEs is limited when compared to genuine sentences. We use random sampling as well as confidence active learning, even if the last one has been shown to be somewhat ineffective with deep neural networks and batch samplings [7].

Unsurprisingly, all algorithms outperform the baseline. Surprisingly, most perform as well as the active learning strategies, which we established as our upper bound on the performance. We furthermore see a clear difference between the algorithms that perform poorly (AE, VAE-BD, CVAE) and the ones that perform well (VAE-Random, SSVAE-Random, SSVAE-Confidence). However, due to the high standard deviation, it is hard to establish with certainty which performs best. This standard deviation is unfortunately a natural result of several random choices during the training, including the initial sampling of $\mathcal{L}$ and the randomness of the initialization of the classifier layer of BERT. It

Algorithm 1 Membership Query Synthesis algorithm for the VAE. For the CVAE, the generator would be trained only on $\mathcal{L}$ (line 7). For the SSVAE, the line 6 to 14 would be encased in a loop of 4 iterations which would also include the fine-tuning of a classifier, and we would generate a quarter of the examples at each iteration.

1: **procedure** MEMBERSHIP QUERY SYNTHESIS($\mathcal{L}, \mathcal{U}$):
2: *Budget* $\leftarrow 500 - |\mathcal{L}|$ ▷ $\mathcal{L}$ contains [10, 50, 100] randomly selected examples
3: *Classifier* $\leftarrow$ BERT
4: *Labeller* $\leftarrow$ BERT-FULL ▷ Labeller is already fine-tuned on train+test
5: *Generator* $\leftarrow$ VAE
6: *Generator.train*($\mathcal{U} + \mathcal{L}$)
7: **while** *Budget*>0 **do**
8: *example* $\leftarrow$ *Generator.generate()*
9: *label*, *confidence* $\leftarrow$ *Labeller.label(example)*
10: **if** *example* in $\mathcal{L}$ **then**
11: Continue
12: **else if** *confidence*>0.7 **then**
13: $\mathcal{L}$.append(*example, label*)
14: *Budget* $\leftarrow$ *Budget* $- 1$
15: *Classifier.finetune*($\mathcal{L}$)

is also interesting to note that the SSVAE seems to have a lower standard deviation than the VAE, maybe due to the interplay between the classifier and the generator, which may help stabilize the system.

	10	50	100
Baseline	52.1 (2.1)	53.2 (3.0)	61.6 (2.6)
AE- RANDOM	70.1 (7.4)	75.6 (4.4)	73.5 (6.7)
Random AL	82.5 (1.6)	82.6 (2.3)	82.5 (2.6)
Confidence AL	83.8 (2.1)	80.9 (2.2)	84.7 (1.4)
VAE- RANDOM	82.2 (4.7)	**82.5** (4.3)	**84.7** (4.1)
VAE- DB	50.8 (3.6)	58.5 (4.1)	65.1 (5.4)
CVAE- RANDOM	51.2 (0.9)	54.5 (4.5)	60.3 (2.6)
SSVAE- RANDOM	**83.4** (2.3)	81.9 (2.8)	82.1 (1.0)
SSVAE- CONFIDENCE	81.3 (2.8)	82.0 (1.2)	82.1 (3.5)

Table 2. Results for three seed sizes and a total annotation budget of 500. In parentheses are the standard deviations calculated through six runs. Bold figures indicate the best performance among the MQS variants, while underlined scores are the best ones overall.

We show in Table 3 positive and negative sentences as labelled by BERT-FULL, for each algorithm and with a seed size of 100. We can see that both the VAE and SSVAE generates sentences of a similar quality, while the CVAE generates sentences of poor quality. In the next section we analyze results for all three algorithms.

6. Analysis

6.1. Performance and limits of the generators

VAE The VAE performs well when random sampling is used, but poorly when the boundary decision strategy is used. This is because, as shown by Figure 3, the VAE does not consider the class when organizing its latent space, instead focussing on other factors, such as the length of the sentence

Algorithm	Generated sentence	Polarity
VAE	an authentic and deeply felt work of the worst kind of an artist .	Positive
	the movie is a desperate miscalculation .	Negative
CVAE	it ' s , and it ' s , and it ' s , and it ' s , and it ' s , and it ' s , and it ' s , and it ' s , and it ' s , and it ' s , and it ' s , and it ' s , and it ' s , and	Positive
	a waste .	Negative
SSVAE	as the director ' s most refreshing and most likeable , the film is a smart , unforced intimacy .	Positive
	in all the way , it ' s just tediously bad .	Negative

Table 3. Examples of generated sentences for a seed size of 100 as well as the polarity as determined by the labeller.

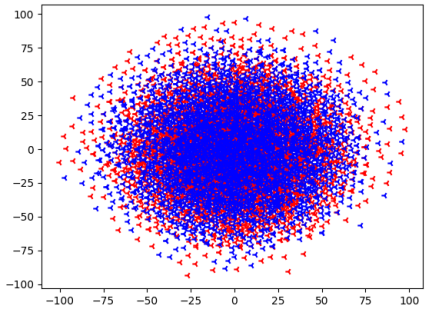

Figure 3. T-SNE visualization of the dataset encoded in the latent space of the VAE. We can see that it doesn't make a distinction between positive examples (red) and negative ones (blue), focussing on other characteristics of the input sentences instead.

or the words themselves[2]. This lack of disentanglement is a known issue in the literature, especially for textual VAE [24]. Nevertheless, we can see from Figure 4 that the VAE is trained correctly, and that interpolating between two random points in the latent space produces an interpolation between the two corresponding sentences. One interesting thing to explore in future work is the use of disentangled VAE, which could help with the class-separation issue in the latent space.

CVAE The CVAE performs poorly due to a lack of data. As a sanity check that it works correctly, we train it on all data, generate 500 examples of each class, and pass them through the labeller to verify that they are of the correct class. Doing so, we observe that with enough data it generates correctly from the given class 83% of the time, but with the limited data we use, it suffers from KL-collapse and cannot efficiently learn to generate from the latent space.

1	enriched about a devastating indictment of unbridled greed and materialism .
2	enriched , a warm and moving film that is part of the fun .
3	made a devastating , sobering film that is not always a fun .
4	a smart , funny and frequently funny film that is not in the first part .
5	a smart , sassy and exceptionally charming that is funny .

Figure 4. Interpolation between two random points (1 and 5) in the latent space for the VAE.

[2]We attempted several combinations of hyperparameters, but no matter what we were unable to make the VAE learn to separate classes in the latent space.

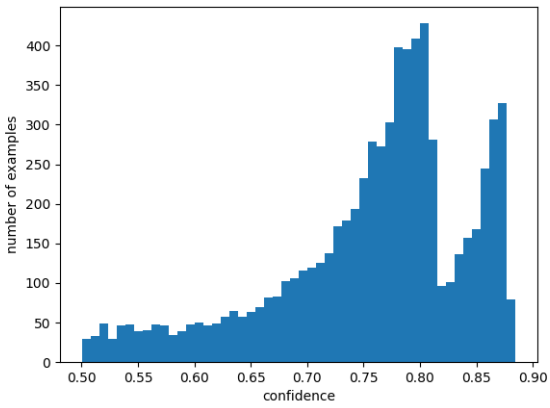

Figure 5. Distribution of the confidence of BERT for the training examples when trained with 100 labelled examples.

SSVAE Similarly to the CVAE, we want to check that our SSVAE learns correctly. When all labelled data is used, we reach an agreement of 76.2%[3]. Thus, we believe that the reason for which SSVAE-confidence does not work better than SSVAE-random is because while it learns to generate sentences according to the confidence of the classifier, our BERT classifier has a tendency to be overconfident in its predictions.

In fact, if we check the confidence of BERT predictions over the whole dataset, as shown in Figure 5, we can see that there are very few examples for which BERT is uncertain about. This has the unfortunate effect that the SSVAE rarely sees examples for which BERT is not very confident, and therefore when we attempt to generate those examples, the SSVAE does not know what to do. We attempted to weaken the classifier by lowering the number of training epochs at each of the four iterations, so it would be less confident in its predictions, but that worsened considerably the performance[4]. Another solution we attempted was to introduce a secondary classifier (a logistic regression based on TF-IDF), to give the probability distribution to the SSVAE instead of BERT. This did not improve the performance much[5], most likely because what is important to the logistic regression may not be what is important for BERT. Finally, there are several techniques that attempt to calibrate the confidence of the network so that it matches the accuracy [25], any of which could help the SSVAE see more uncertain examples.

6.2. **Seed size and labelling budget**

Until now, we have considered a fixed labelling budget of 500 for all our experiments, which allowed us to compare efficiently the various algorithms. We have also experimented with three seed sizes, namely 10, 50, and 100, but with a large labelling budget that drowned out the genuine examples in artificial ones. In this section, however, we consider and analyze various seed sizes as well as various labelling budgets.

Figure 6 shows the results of our experiments, for a random selection of the latent space of the VAE. We observe that it is very difficult to determine which version works consistently better, especially considering the high standard deviation. It is not surprising that the seed size has no impact on the final performance, given that we use random selection which doesn't use the seed to

[3]We classify it this time with the classifier and not the labeller, since the goal of the SSVAE is to learn to generate data according to the confidence of the classifier.

[4]Reducing the number of epoch of the classifier to 2, we obtained 79.4%, 81.4%, and 79.8% on the three seed sizes.

[5]Using a logistic regression as a secondary classifier, we obtain 82.1%, 80.4%, and 82.7% on the three seed sizes.

start the MQS process, but it does show that no matter how big the labelling budget and how many new generated sentences we add, they stay as informative as genuine ones.

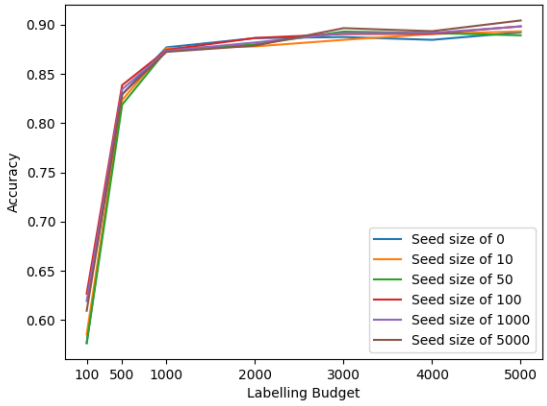

Figure 6. Accuracy vs labelling budget for various seed sizes. When the labelling budget is smaller than the seed, we randomly sample from it, which corresponds in that case to random active learning. Each point is the average over six experiments with the given seed size and labelling budget.

Another interesting case is the use of MQS to extend the training set after its initial collection. It has been shown on text that data augmentation can be inefficient when the size of the dataset is large enough [26], so MQS could be an alternative solution for that. Running experiments with the full dataset as the seed and adding 15000[6] examples, we obtain an accuracy of 91.0%. When we train BERT on the seed only, we obtain 90.5%. While the MQS gain may seem small, especially considering the human labelling cost in a real setting, there are many domains where even a small augmentation is worth the cost, such as medical diagnostics. As a point of comparison, we use EDA [27], a popular data augmentation technique for sentences based on word edition (substitutions, swapping, insertion, and deletions of words) to generate 15000 new data points, and we obtain 89.7%, showing that it is definitively not an ideal solution for large datasets.

Finally, our experiments raise the question of what proportion of the generated examples are identical to examples present in $\mathcal{U}$. Taking 5000 generated examples, we find only 50 that are present in $\mathcal{U}$. They correspond to very short sentences, averaging 5.4 words. Examples include "thumbs down .", "there is greatness here .", and "... bibbidy-bobbidi-bland .".

6.3. **Limits of BERT as a labeller**

While we reached a 92% human / BERT-FULL agreement with the SSVAE, the *decrease* of performance while using the CVAE highlights some shortcomings of replacing the human labeller with a BERT system, namely that it has a tendency to label examples that are unintelligible with high confidence, such as a repetition of ", and", which gets labelled as positive.

As a final sanity check, we run one experiment using 100 initial examples, a total budget of 500, the VAE, and a random sampling, and compare the performance of a BERT-FULL, a human labeller, and a BERT fine-tuned only on the training set. With a human labeller, we get a performance of 86.5%, while using BERT-FULL on the same sentences gives a performance of 82.5%. Finally, using BERT fine-tuned only on the training set as a labeller gives a performance of 80.0%. That implies that the performances we report in Table 2 are probably underestimated, and that the real

[6]Because of the diminishing return obtained by adding new examples as the dataset size grows, we chose to add a lot of new data, more than tripling the size of it.

performance of MQS is better than random pool-based AL. While we only use random sampling of the latent space, this is most likely due to the fact that the generated examples are "more difficult", since they are less grammatically sound than examples produced by a human, which provides a sort of regularization as the dataset is created.

Overall, there are several limitations to the use of BERT-FULL as a labeller. First and foremost, as we have just shown, the use of BERT-FULL to simulate the human labeller is limited, since it is noisy. Furthermore, we cannot expect the classifier to outperform the labeller, so replacing the human component upper bounds the final performance.

Closely related to that is the fact that we count here a uniform budget for all examples, but labelling artificial examples has a higher cost than labelling genuine ones, due to their lower quality.

Finally, there exists some key differences in the way BERT-FULL and a human will label examples. Mostly, the examples that are skipped because of labelling uncertainty differ. For example, we labelled "the movie is a lumbering load of hokum and a storyline, and an old vision." as negative, while BERT skipped it. Similarly, we skipped "the movie's narrative gymnastics and its point view of view, but it's also a nice to watch the target practice of the year.", while BERT labelled it as positive.

These differences limit the analysis that can be done on the generated examples, since labelling is not the same as what a human would do.

7. Future Work and Conclusion

Membership query synthesis is often understudied because of the cost associated with labelling new data points. In this paper, we circumvent this issue by using a labeller trained on both train and test sets as our oracle, and show that using MQS to artificially create new datapoints can reach performances equal to Random Active Learning on a real task. We also show, although not rigorously, that the use of such an oracle provides a lower bound on the performance with a human labeller.

There is much work that remains to be done in MQS. For example, given that our best strategy remains random sampling with a VAE, an interesting approach to evaluate would be to see if using top-k sampling with a large pretrained generative model to generate examples that would be labelled would beat this strategy, similarly to what is done in data augmentation [28]. Recently, a VAE-GPT has also been proposed [29], which would provide an interesting avenue since it would theoretically increase the quality of the sentences while allowing us to keep sampling randomly from the distribution.

Given our hypothesis that MQS brings a regularization factor by creating more difficult sentences, it would also be interesting to see if mixing pool-based AL and MQS would work better than using only one strategy. Finally, further fine-tuning with the SSVAE could be done, for example by introducing reinforcement learning.

References

[1] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. ISSN: 2380-7504. Oct. 2017, pp. 843–852. DOI: 10.1109/ICCV.2017.97.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386 (visited on 01/01/2022).

[3] S. Budd, E. C. Robinson, and B. Kainz. "A survey on active learning and human-in-the-loop deep learning for medical image analysis". en. In: *Medical Image Analysis* 71 (July 2021), p. 102062. ISSN: 1361-8415. DOI: 10.1016/j.media.2021.102062. URL: https://www.sciencedirect.com/science/article/pii/S1361841521001080 (visited on 01/02/2022).

[4] I. Feldman and R. Coto-Solano. "Neural Machine Translation Models with Back-Translation for the Extremely Low-Resource Indigenous Language Bribri". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 3965–3976. DOI: 10.18653/v1/2020.coling-main.351. URL: https://aclanthology.org/2020.coling-main.351 (visited on 01/02/2022).

[5] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. "Semi-supervised learning with deep generative models". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Cambridge, MA, USA: MIT Press, Dec. 2014, pp. 3581–3589. (Visited on 02/04/2022).

[6] J.-C. Su, Z. Cheng, and S. Maji. "A Realistic Evaluation of Semi-Supervised Learning for Fine-Grained Classification". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. ISSN: 2575-7075. June 2021, pp. 12961–12970. DOI: 10.1109/CVPR46437.2021.01277.

[7] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang. "A Survey of Deep Active Learning". In: *ACM Computing Surveys* 54.9 (Oct. 2021), 180:1–180:40. ISSN: 0360-0300. DOI: 10.1145/3472291. URL: https://doi.org/10.1145/3472291 (visited on 02/04/2022).

[8] M. Hopkins, D. Kane, S. Lovett, and G. Mahajan. "Point Location and Active Learning: Learning Halfspaces Almost Optimally". In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. ISSN: 2575-8454. Nov. 2020, pp. 1034–1044. DOI: 10.1109/FOCS46700.2020.00100.

[9] R. Schumann and I. Rehbein. "Active Learning via Membership Query Synthesis for Semi-Supervised Sentence Classification". In: *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 472–481. DOI: 10.18653/v1/K19-1044. URL: https://www.aclweb.org/anthology/K19-1044 (visited on 03/05/2020).

[10] Y. Shen, H. Yun, Z. Lipton, Y. Kronrod, and A. Anandkumar. "Deep Active Learning for Named Entity Recognition". In: *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 252–256. DOI: 10.18653/v1/W17-2630. URL: https://aclanthology.org/W17-2630 (visited on 02/04/2022).

[11] G. Haffari, M. Roy, and A. Sarkar. "Active learning for statistical phrase-based machine translation". en. In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics on - NAACL '09*. Boulder, Colorado: Association for Computational Linguistics, 2009, p. 415. ISBN: 978-1-932432-41-1. DOI: 10.3115/1620754.1620815. URL: http://portal.acm.org/citation.cfm?doid=1620754.1620815 (visited on 01/27/2022).

[12] S. Dasgupta and D. Hsu. "Hierarchical sampling for active learning". en. In: *Proceedings of the 25th international conference on Machine learning - ICML '08*. Helsinki, Finland: ACM Press, 2008, pp. 208–215. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390183. URL: http://portal.acm.org/citation.cfm?doid=1390156.1390183 (visited on 07/03/2019).

[13] M. Ravanbakhsh, T. Klein, K. Batmanghelich, and M. Nabi. *Uncertainty-Driven Semantic Segmentation through Human-Machine Collaborative Learning*. Tech. rep. Publication Title: arXiv e-prints ADS Bibcode: 2019arXiv190900626R Type: article. Sept. 2019. URL: https://ui.adsabs.harvard.edu/abs/2019arXiv190900626R (visited on 02/04/2022).

[14] J. Zarecki and S. Markovitch. "Textual Membership Queries". en. In: vol. 3. ISSN: 1045-0823. July 2020, pp. 2662–2668. DOI: 10.24963/ijcai.2020/369. URL: https://www.ijcai.org/proceedings/2020/369 (visited on 02/04/2022).

[15] J.-J. Zhu and J. Bento. "Generative Adversarial Active Learning". In: (Feb. 2017).

[16] L. Wang, X. Hu, B. Yuan, and J. Lu. "Active learning via query synthesis and nearest neighbour search". en. In: *Neurocomputing*. Advances in Self-Organizing Maps Subtitle of the special issue: Selected Papers from the Workshop on Self-Organizing Maps 2012 (WSOM 2012) 147 (Jan. 2015), pp. 426–434. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2014.06.042. URL: http://www.sciencedirect.com/science/article/pii/S0925231214008145 (visited on 02/29/2020).

[17] D. P. Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *arXiv:1312.6114 [cs, stat]* (May 2014). arXiv: 1312.6114. URL: http://arxiv.org/abs/1312.6114 (visited on 04/10/2020).

[18] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio. "Generating Sentences from a Continuous Space". In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 10–

21. DOI: 10.18653/v1/K16-1002. URL: https://www.aclweb.org/anthology/K16-1002 (visited on 03/19/2020).

[19] A. B. Dieng, Y. Kim, A. M. Rush, and D. M. Blei. "Avoiding Latent Variable Collapse with Generative Skip Models". en. In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. ISSN: 2640-3498. PMLR, Apr. 2019, pp. 2397–2405. URL: https://proceedings.mlr.press/v89/dieng19a.html (visited on 02/12/2022).

[20] X. Yan, J. Yang, K. Sohn, and H. Lee. "Attribute2Image: Conditional Image Generation from Visual Attributes". en. In: *Computer Vision – ECCV 2016*. Ed. by B. Leibe, J. Matas, N. Sebe, and M. Welling. Vol. 9908. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 776–791. ISBN: 978-3-319-46492-3 978-3-319-46493-0. DOI: 10.1007/978-3-319-46493-0_47. URL: http://link.springer.com/10.1007/978-3-319-46493-0_47 (visited on 02/12/2022).

[21] W. Xu, H. Sun, C. Deng, and Y. Tan. "Variational Autoencoders for Semi-supervised Text Classification". en. In: *arXiv:1603.02514 [cs]* (Nov. 2016). arXiv: 1603.02514. URL: http://arxiv.org/abs/1603.02514 (visited on 10/10/2020).

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423 (visited on 11/10/2021).

[23] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. URL: https://aclanthology.org/D13-1170 (visited on 11/18/2021).

[24] V. Balasubramanian, I. Kobyzev, H. Bahuleyan, I. Shapiro, and O. Vechtomova. "Polarized-VAE: Proximity Based Disentangled Representation Learning for Text Generation". en. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, 2021, pp. 416–423. DOI: 10.18653/v1/2021.eacl-main.32. URL: https://aclanthology.org/2021.eacl-main.32 (visited on 02/12/2022).

[25] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. "On Calibration of Modern Neural Networks". en. In: *Proceedings of the 34th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, July 2017, pp. 1321–1330. URL: https://proceedings.mlr.press/v70/guo17a.html (visited on 03/02/2022).

[26] X. Dai and H. Adel. "An Analysis of Simple Data Augmentation for Named Entity Recognition". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 3861–3867. DOI: 10.18653/v1/2020.coling-main.343. URL: https://aclanthology.org/2020.coling-main.343 (visited on 11/10/2021).

[27] J. Wei and K. Zou. "EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6382–6388. DOI: 10.18653/v1/D19-1670. URL: https://aclanthology.org/D19-1670 (visited on 11/08/2021).

[28] V. Kumar, A. Choudhary, and E. Cho. "Data Augmentation using Pre-trained Transformer Models". In: *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*. Suzhou, China: Association for Computational Linguistics, Dec. 2020, pp. 18–26. URL: https://aclanthology.org/2020.lifelongnlp-1.3 (visited on 11/09/2021).

[29] K. Zhao, H. Ding, K. Ye, and X. Cui. "A Transformer-Based Hierarchical Variational AutoEncoder Combined Hidden Markov Model for Long Text Generation". en. In: *Entropy* 23.10 (Sept. 2021), p. 1277. ISSN: 1099-4300. DOI: 10.3390/e23101277. URL: https://www.mdpi.com/1099-4300/23/10/1277 (visited on 01/03/2022).