Active Learning via Membership Query Synthesis for Semi-supervised Sentence Classification

Raphael Schumann
Institute for Computational Linguistics
Heidelberg University, Germany
rschuman@cl.uni-heidelberg.de

Ines Rehbein
Leibniz ScienceCampus
Heidelberg/Mannheim
rehbein@ids-mannheim.de

Abstract

Active learning (AL) is a technique for reducing manual annotation effort during the annotation of training data for machine learning classifiers. For NLP tasks, pool-based and stream-based sampling techniques have been used to select new instances for AL while generating new, artificial instances via Membership Query Synthesis was, up to know, considered to be infeasible for NLP problems. We present the first successful attempt to use Membership Query Synthesis for generating AL queries for natural language processing, using Variational Autoencoders for query generation. We evaluate our approach in a text classification task and demonstrate that query synthesis shows competitive performance to pool-based AL strategies while substantially reducing annotation time.

1 Introduction

Active learning (AL) has the potential to substantially reduce the amount of labeled instances needed to reach a certain classifier performance in supervised machine learning. It works by selecting new instances that are highly informative for the classifier, so that comparable classification accuracies can be obtained on a much smaller training set. AL strategies can be categorized into pool-based sampling, stream-based sampling and Membership Query Synthesis (MQS). The first two strategies sample new instances either from a data pool or from a stream of data. The third, MQS, generates artificial AL instances from the region of uncertainty of the classifier. While it is known that MQS can reduce the predictive error rate more quickly than pool-based sampling (Ling and Du, 2008), so far it has not been used for NLP tasks because artificially created textual instances are uninterpretable for human annotators.

We provide proof of concept that generating highly informative artificial training instances for text classification is feasible. We use Variational Autoencoders (VAE) (Kingma and Welling, 2013) to learn representations from unlabeled text in an unsupervised fashion by encoding individual sentences as low-dimensional vectors in latent space. In addition to mapping input sequences into latent space, the VAE can also learn to generate new instances from this space. We utilize these abilities to generate new examples for active learning from a region in latent space where the classifier is most uncertain, and hand them over to the annotator who then provides labels for the newly created instances.

We test our approach in a text classification setup with a real human annotator in the loop. Our experiments show that query synthesis for NLP is not only feasible but can outperform other AL strategies in a sentiment classification task with respect to annotation time.

The paper is structured as follows. We first review related work (§2) and introduce a formal description of the problem (§3). Then we describe our approach (§4), present the experiments (§5) and analyze the results (§6). We discuss limitations and possible further experiments (§7) and finally conclude our findings (§8).

2 Related work

Membership query synthesis was introduced by Angluin (1988) and describes a setting where the model generates new queries instead of selecting existing ones. Early experiments in image processing (Lang and Baum, 1992), however, showed that the generated queries are hard to interpret by human annotators. This holds true even for recent approaches using Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) to create uncertain instances (Zhu and Bento, 2017; Huijser and van Gemert, 2017). In contrast to im-

age processing, discrete domains like natural language do not exhibit a direct mapping from feature to instance space. Strategies that circumvent this problem include the search for nearest (observed) neighbors in feature space (Wang et al., 2015) or crafting queries by switching words (Awasthi and Kanade, 2012).

Sentence representation learning (Kiros et al., 2015; Conneau et al., 2017; Subramanian et al., 2018; Wang et al., 2019) in combination with new methods for semi-supervised learning (Kingma et al., 2014; Hu et al., 2017; Xu et al., 2017; Odena, 2016; Radford et al., 2017) have shown to improve classification tasks by leveraging unlabeled text. Methods based on deep generative models like GANs or VAEs are able to generate sentences from any point in representation space. Mehrjou et al. (2018) use VAEs to learn structural information from unlabeled data and use it as an additional criterion in conventional active learning to make it more robust against outliers and noise.

We use VAEs to generate AL queries from specific regions in latent space. To ensure that the generated instances are not only informative for the ML classifier but also meaningful for the human annotator, we adapt the approach of Wang et al. (2015) (see §3.1). In contrast to their work, however, we do not *sample* existing instances from the pool that are similar to the synthetic ones but directly *generate* the new queries. To our best knowledge, our work is the first to present positive results for Membership Query Synthesis for text classification.

3 Background

3.1 Query Synthesis and Nearest Neighbors

Arbitrary points in feature space are hard to interpret for humans. To evade this problem, Wang et al. (2015) use the nearest neighbor in a pool of unlabeled data as a representative which is then presented to the human annotator. To identify uncertain points along the separating hyperplane of an SVM the following approach is proposed. First the location of the decision boundary is approximated by a binary-search like procedure. An initial *Opposite Pair* (z_+, z_-) is formed by centroid c_+ and centroid c_- of positive and negative labeled instances respectively. The mid point z_s is queried and, depending on the annotated label l, replaces the corresponding z_l. This step is repeated b times, reducing the distance between the initial centroids by a factor of 2^b. Figure 1a depicts this process. Then the mid-perpendicular vector of the *Opposite Pair* is calculated by using the Gram-Schmidt process to orthogonalize a random vector z_r and normalize its magnitude to λ. The new point $z_s = z_r + (z_+ + z_-)/2$ is close to the decision boundary and queried for its class. Depending on the receive label the point z_s replaces z_+ or z_- in the *Opposite Pair*. This process (Figure 1b) is repeated until $n - b$ points along the separating hyperplane are queried.

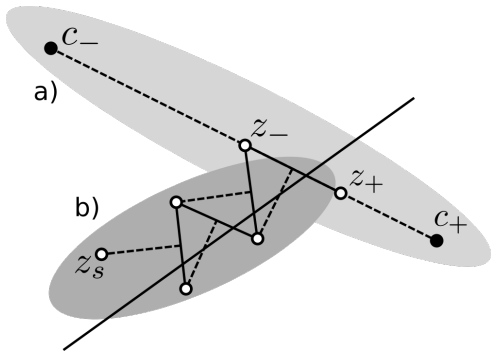

Figure 1: **a)** finds *Opposite Pair* close to the decision boundary. **b)** identify points close to the decision boundary.

3.2 VAE for Sentence Generation

The Variational Autoencoder is a generative model first introduced by Kingma and Welling (2013). Like other autoencoders, VAEs learn a mapping $q_\theta(z|x)$ from high dimensional input x to a low dimensional latent variable z. Instead of doing this in a deterministic way, the encoder learns the parameters of e.g. a normal distribution. The desired effect is that each area in the latent space has a semantic meaning and thus samples from $p(z)$ can be decoded in a meaningful way. The decoder $p_\theta(x|z)$, also referred to as $dec(z)$, is trained to reconstruct the input x based on the latent variable z. In order to approximate θ via gradient descent the reparametrization trick (Kingma and Welling, 2013) was introduced. This trick allows the gradient to flow through non-deterministic z by separating the discrete sampling operation. Let μ and σ be deterministic outputs of the encoder $q_\theta(\mu, \sigma|x)$:

$$z = \mu + \sigma \odot \epsilon \quad \text{where } \epsilon \sim \mathcal{N}(0, I) \qquad (1)$$

and $\odot$ is the element-wise product. To prevent the model from pushing σ close to 0 and thus falling back to a deterministic autoencoder, the objective is extended by the Kullback-Leibler (KL) diver-

gence between prior $p(z)$ and $q(z|x)$:

$$\mathcal{L}(\theta; x) = -KL(q_\theta(z|x)||p(z)) + \mathbb{E}_{q_\theta(z|x)}[logp_\theta(x|z)]. \quad (2)$$

Bowman et al. (2016) apply this idea for sentence generation using an RNN as encoder and decoder. They observe that a strong auto-regressive language modeling ability in the decoder reduces the information stored in the latent variable, right up to a complete collapse of the KL term. They explore different techniques to weaken the decoder, like word dropout or KL term weight annealing, as possible solutions. This guarantees a semantically rich latent variable and good sentence generation ability. Below, we describe how to combine both techniques in order to generate meaningful queries for Membership Query Synthesis.

4 Active Learning Schedule

We train a Variational Autoencoder on an unlabeled corpus of sentences. The text classification task is performed on a binary sentiment dataset split into training, development and test set. As depicted in Figure 2, the sentences in the classification dataset are vectorized using the VAE encoder which generates the latent variable z for each sentence x. This is done deterministically by dropping the σ term in Equation 1, further referred to as $z = enc(x)$.

Next, a *Learner* is trained to fit a linear hyperplane to separate the positive from the negative instances. We use the procedure described in §3.1 to select new query points for AL. But instead of searching for the nearest neighbor in the pool, we decode the point $x = dec(z)$ into a human readable sentence which is then handed over to the human annotator. The annotator assigns a binary label to the instance and the next query point is calculated.

One important parameter for active learning determines how many new instances are to be selected in each AL iteration. Wang et al. (2015) use a predefined number of instances to be selected along the hyperplane. Because we know that a Gaussian prior is imposed on the feature space, we instead stop the selection process when the magnitude of z_s exceeds the expectation. The expected distance of a point sampled from the k-dimensional Gaussian prior to the origin is $\sqrt{\mathbb{E}[\chi^2_k]} = \sqrt{k}$. Then the schedule restarts, learning a new decision boundary, and ultimately terminates when the annotation budget is exhausted. We refer to this method as *gen_wang*. When nearest neighbor search is used instead, we refer to the selection method as *nn_wang*.

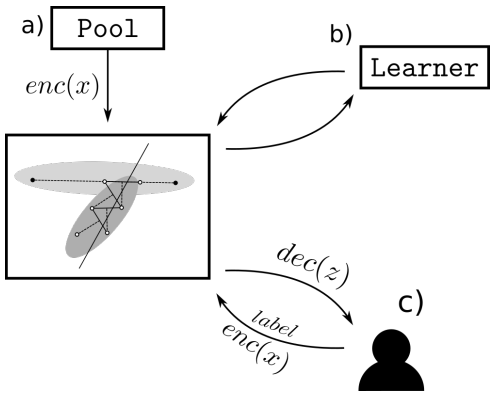

Figure 2: **a)** Instances in corpus are encoded to latent space. **b)** Learner fits a hyperplane to separate points. **c)** Query points selected by method described in Fig.1. Decoder translates point in latent space to human readable sequence. Annotator chooses label for instance.

In addition, we explore a method, *gen_uniform*, where step b) in Figure 1 is reduced to generating only one midperpendicular vector with a magnitude drawn from a uniform distribution. In each iteration this vector will point to a random direction with a different magnitude, selecting diverse points close to the hyperplane. The maximum magnitude is set in a way that the resulting point is not further away than $\sqrt{k}$ from the origin. Similar to above we refer to this method as *nn_uniform* when using nearest neighbor search. The number of possible directions along the hyperplane grows with the size of the latent variable. With this modification we expect to explore more diverse points than following the same direction for several steps.

5 Experiments

In this section we want to explore how the ability to generate human readable sentences from arbitrary points in the feature space affects active learning performance. We compare our approach to a number of baselines (§5.3), where in each experiment we select/generate 500 instances, present them to a human annotator to get a label and evaluate the performance of each setting in a sentiment classification task. We start the active learning process with two utterances in the seed set, namely '*good movie*' and '*bad movie*'. The classifier is trained to separate instances with positive sentiment from negative ones. The human anno-

Parameter	Value
vocabulary size	20.000
RNN cell size	512
embedding size	512
latent variable size	50
dropout	0.3
dropword	0.5
learning rate	0.005
epochs	20

Table 1: Training parameters for the Variational Autoencoder.

tator can skip neutral or uninterpretable instances. These skip actions also count towards the annotation budget.

5.1 Data

The data used in our experiments comes from two sources, (i) the SST2 (Socher et al., 2013) and (ii) SAR14 (Nguyen et al., 2014). We limit sentence length to a maximum of 15 words. This is motivated by lower training times and the tendency of vanilla VAEs not to perform well on longer sentences (Shen et al., 2019).

Sentiment task SST2 (Socher et al., 2013) is a binary sentiment classification dataset compiled from *rottentomatoes.com*. As we only consider sentences with up to 15 words, the sizes of the training, development and test sets are 3103, 380 and 814 instances, respectively.

Sentence pool The active learning pool consists of 1.2M unique sentences from the SAR14 dataset (Nguyen et al., 2014). SAR14 contains 234k movie reviews from IMDB. The data is annotated on review level, which prevents us from removing single neutral sentences. Although the datasets stem from different sources, there is a small overlap. These sentences are removed from the pool.

5.2 Training

Variational Autoencoder Table 1 lists the parameter used for the VAE. For training we limit the vocabulary of the VAE to the top 20k words. Encoder and decoder RNN have layer normalized (Ba et al., 2016) LSTM cells (Hochreiter and Schmidhuber, 1997) with size 512. As additional regularization we set weight dropout to 0.3 (Srivastava et al., 2014). Input embeddings are also of size 512, which allows us to share the embedding weights with the softmax weights of the output layer (Press and Wolf, 2016). To prevent posterior collapse we use logistic annealing of the KL term weight and weaken the decoder by applying word dropout with probability 0.5 (Bowman et al., 2016). The model is trained using the Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of 0.005. Once the KL term weight is close to 1, the learning weight is linearly decreased to 0. The training stops after 20 epochs and the latent variable z has $k = 50$ dimensions. The trained VAE achieves a reconstruction loss of 45.3 and KL divergence of 13.2 on the SST2 training set.

Learner The *Learner* is an SVM[1] with linear kernel. Each instance is represented as the latent variable z learned by the autoencoder. The latent variable is a vector with 50 dimensions and the SVM is trained on this representation. We calculate classification performance on the reduced SST2 test set and report F1-scores.

Generator The generator is the decoder of the VAE described above. Once a point z in feature space is selected, it is used as the input of the decoder $x = dec(z)$ which generates the human readable sentence x in an autoregressive way.

5.3 Baselines

We compare our approach to Membership Query Synthesis for text classification to four baselines. The first baseline selects instances from the pool by *random* choice. The *least confidence* baseline computes the distance of the instances in the pool to the separating hyperplane and chooses the one closest to the hyperplane. The third and fourth baseline follow the procedure described in §4 but search for the nearest neighbor (*nn_uniform*, *nn_wang*) instead of synthesising the exact query point. Nearest neighbor is defined by the minimal euclidean distance between the query point and the latent representation of the pool instance.

5.4 Annotation

The instances selected or generated by any model or baseline are annotated manually by one human coder.[2] Although the pool data has labels on the review level, we do not use these labels in our experiments. Positive reviews can include negative

[1] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[2] The first author of this paper.

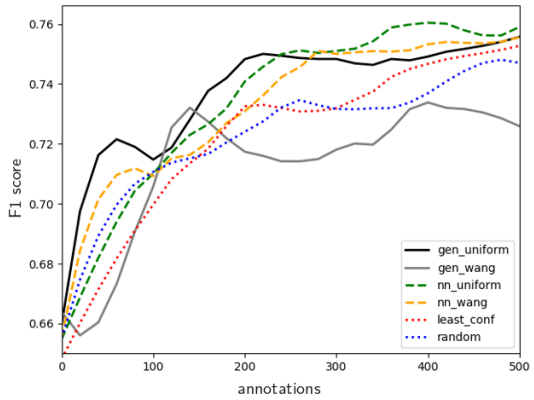

Figure 3: F1-Score as a function of annotation steps (including skipped queries). Averaged over 3 runs.

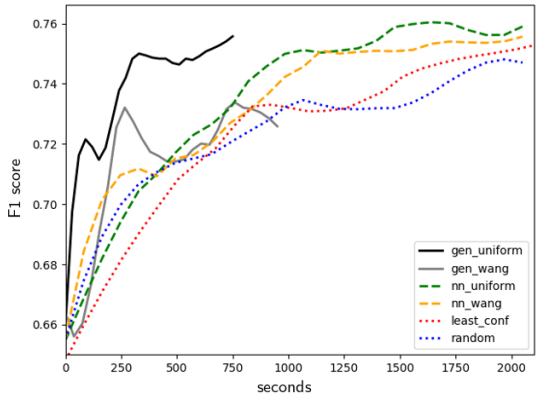

Figure 4: F1-scores as a function of annotation time. Results averaged over 3 runs.

sentences and vice versa. This means that using document-level labels would introduce noise and might impair the baselines. During each of the three experimental runs, all models and baselines are annotated simultaneously by the same person. The annotator is presented with one instance at a time and has no information which of the models has produced each particular instance. Once a label is selected, it is transmitted to the corresponding model and triggers the selection/generation of the next instance. Thus, at any given time there is one unlabeled instance for each model or baseline. From this set of unlabeled instances, one instance is chosen randomly and presented to the annotator. This procedure is repeated until 500 instances are labeled for each model or baseline. Hiding the instance source from the annotator is intended to prevent any bias during the annotation process.

6 Results and Analysis

6.1 Classification Performance

F-scores as a function of annotated instances Figure 3 shows learning curves for the different AL strategies and baselines as a function of the number of annotation instances added to the training data. The *random* and *least_conf* baselines perform reasonably well. *Least_conf* struggles in the beginning, likely attributed to the minimal seed set. Once enough instances are labeled it catches up. *Gen_uniform* has a strong start but, after around 200 instances, is outperformed by the *nearest neighbor* approaches which yield the highest F1-scores. Among the nearest neighbor approaches, the *uniform* schedule ranks better than *wang*. The same behaviour is observed for the *generation* methods, although *gen_wang* produces the worst results overall. Overall, *gen_uniform* is competitive with respect to F1-scores and shows that sentences generated from points in the feature space are informative and useful for training a text classifier.

F-scores as a function of annotation time AL simulations have often been criticized for reporting unrealistic results, based merely on the number of annotated instances (see, e.g., Settles (2009), pp. 37 ff.). It is well known, however, that the number of annotated instances is often not a good predictor for the real annotation costs. AL strategies tend to select the *hard nuts* for human annotators and it is not unreasonable to assume that the annotation of N instances in an AL setup might take longer and thus might be more expensive than annotating the same number of *randomly* selected instances. Therefore, we also show learning curves as a function of annotation time (Figure 4).

The results show a clear advantage for the *generation* models. The reduction in annotation time is due to shorter query length and less neutral or noisy instances, as shown in Table 2. This speeds up the annotation by a significant margin while providing the *Learner* with informative instances, despite their short length.

Figure 5 shows that the length of generated instances increase over time and further exploration also hints that the generated length is correlated with the length of the sentences in the seed set.

As listed in Table 2, the *random* baseline reveals that 36.8 percent of sentences in the pool are neutral/artifacts and positive sentences outweigh negative ones by a factor of 2.6. This means that random sampling results in unbalanced datasets with far more positive examples. Our generation

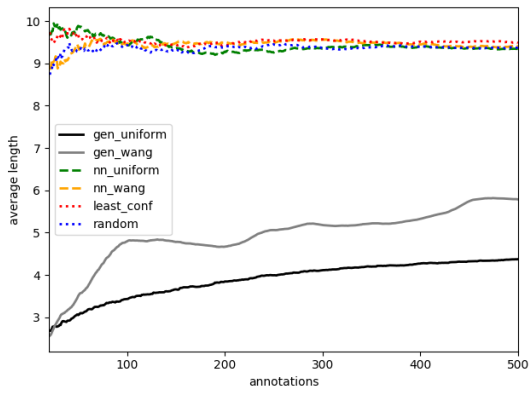

Figure 5: Development of average length of selected/generated instances as more instances are annotated.

method does not show this disadvantage. In contrast, the generated instances maintain a more balanced distribution of class labels and are less likely to be skipped. These are indicators that the selected points are close to the hyperplane and the VAE is able to generate coherent and highly informative sentences from them.

6.2 Computational Complexity

To assure a seamless annotation procedure, the supply of new instances has to be reasonably fast. The generation and selection of the next instance is dependant on the label of the previous instance. Because of this, there is no way to pre-fetch the next instance in the background and the annotator has to wait for the selection/generation process to finish before the next instance is presented for annotation. However, the runtime for pool-based AL methods is increasing with the pool's size. In contrast, the generation method presented in this work does not have this limitation.

The *least confidence* baseline has a complexity of $\mathcal{O}(n)$ where n is the number of instances in the pool. The complexity of *nearest neighbor* search without any approximation techniques like preclustering is also $\mathcal{O}(n)$. Query generation from an exact point with the decoder has a complexity of $\mathcal{O}(m)$ where m is the length of the sentence and $n >> m$. Because sentences have a natural length limit and in this work are capped to 15 words, one could argue that the complexity is $\mathcal{O}(1)$.

6.3 Generated Instances

Table 3 shows examples of generated instances using the *gen_uniform* method. Example 1-6 show

	% skips	M sec	M len	p/n
gen_uniform	28.1	1.4	4	1.7
gen_wang	20.9	1.9	5	1.2
nn_uniform	34.2	4.1	9	1.9
nn_wang	35.8	4.1	10	2.4
least_conf	39.0	4.2	10	2.1
random	36.8	4.1	9	2.6

Table 2: Percentage of skips (neutral or noisy sentences); Median annotation time in seconds; Median number of words in query; Ratio of positive to negative labels.

prototypical positive and negative instances. Example 7 is ambiguous, caused by the decoder generating an unknown (UNK) token at the position where one would normally expect an evaluative adjective. We see this as an indicator that the point is positioned close to the hyperplane and thus the sentiment of the latent variable is ambiguous. We also observe instances with UNK token which still express a sentiment, as seen in Example 8 an 9. This can be interpreted as a placeholder for a named entity or, in other cases, a specifier like movie genre and does not impact the annotation process.

To explore the ability of the model to generate unseen instances we calculate the percentage of instances not seen in the pool. We only look at instances with an annotated sentiment label, because skipped examples often include noise and thus are unlikely to be present in the pool. 41 and 51 percent of labeled instances are newly generated by *gen_uniform* and *gen_wang* respectively. This provides more evidence that the model is capable of generating new and informative instances.

No.	Instance	Label
1.	the acting is excellent	1
2.	powerful and moving	1
3.	this movie is very enjoyable	1
4.	a complete mess	0
5.	nothing spectacular	0
6.	absolutely terrible !	0
7.	the plot is UNK	skip
8.	well done by UNK	1
9.	the UNK is a disappointment	0

Table 3: Example instances generated by *gen_uniform*. Label 1 for positive and 0 for negative class.

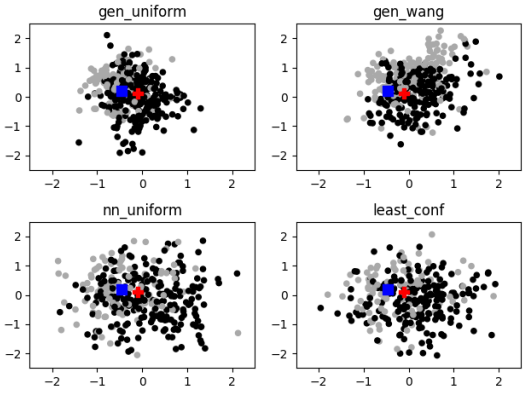

Figure 6: Plot of the 2 *most important* dimensions of selected/generated instances in latent space. Gray points indicate negative, black points positive labels. The blue square denotes '*bad movie*' and the red cross '*good movie*'.

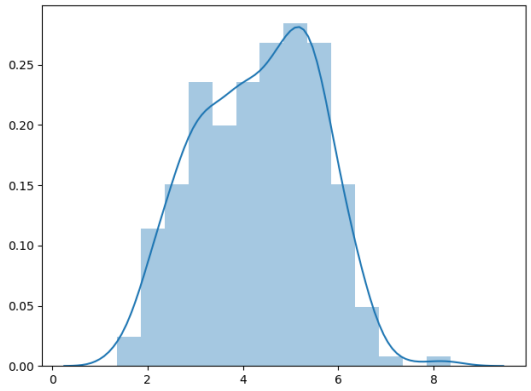

Figure 7: Distribution of euclidean distances between z before and $\hat{z}$ after re-encoding during *gen_uniform*.

6.4 Latent Space

To further analyze the behavior of the different AL strategies, we apply dimensionality reduction and visualize the instances in latent space (Figure 6).

The two largest absolute coefficients of the trained SVM's linear kernel identify the *most important* dimensions. Figure 6 plots the points, represented by theses two dimensions, selected by different active learning schedules. The generated instances lie densely around the seed points, while pool instances are more distributed. In *gen_wang* one can see how the instances are loosely following one direction similar to Figure1.

As indicated in Figure 2 a pool instance is represented as $z = enc(x)$. The same is true for the instances in the development, test and seed set. For the generated instances there are two options. If z is a point selected in feature space and $x = dec(z)$ is the decoded query sequence, the annotated instance can either be represented as z or as $\hat{z} = enc(x)$. In a perfect VAE z and $\hat{z}$ should be nearly identical. In practice however $\hat{z}$ ends up at a different location in feature space. Figure 7 depicts the distribution of distances between z and $\hat{z}$ generated with the *gen_uniform* method. We observe that models trained on $\hat{z}$ perform better than those trained on z, presumably because the test instances are represented the same way. To evaluate if $\hat{z}$ is still an informative point and not just positioned randomly in feature space, we train a model on actual randomly sampled points. The sampled point $z \sim \mathcal{N}(0, I)$ is decoded to query sequence x, labeled and subsequently re-encoded to $\hat{z} = enc(x)$. With the same amount of instances, this model performs much worse than *gen_uniform*, indicating that point $\hat{z}$ still preserves some of the informativeness of z. We thus assume that the closer $\hat{z}$ is to selected point z, the better the generation based active learning schedules will work.

7 Discussion

Related work in the context of semi-supervised learning has focused on developing methods to generate synthetic training instances for different tasks (Sennrich et al., 2016; Hayashi et al., 2018; Alberti et al., 2019; Winata et al., 2019), in order to accelerate the learning process. Sennrich et al. (2016) create artificial training instances for machine translation, using monolingual data paired with automatic back-translations. Their work obtains substantial improvements for several languages and has triggered many follow-up studies that apply the idea of back-translation to different tasks.

For example, Hayashi et al. (2018) augment the training data for attention-based end-to-end automatic speech recognition with synthetic instances, and Winata et al. (2019) generate artificial training examples to improve automatic speech recognition on code-switching material. Alberti et al. (2019) use a large number of synthetic instances to pre-train a Question Answering (QA) model that is then fine-tuned on the target QA dataset. Their approach results in significant improvements over models that are trained without the synthetic datapoints.

While these studies show that huge amounts of synthetic training data can crucially improve the

learning process, our approach uses a different paradigm. Instead of generating millions of synthetic data points, our method is data-lean and only needs a few hundred instances to improve the classifier. Another difference is that we do not rely on automatically generated labels but use human annotations instead. Due to the practical constraints of the active learning process, we need to keep the training time short enough so that the human annotator does not have to wait for the next set of instances to annotate. This rules out the use of computation-intensive models and large training sets. Given that we use an SVM for classification, we do not expect a strong effect for adding large numbers of additional training instances, given that the majority of those data points will not be positioned close to the decision boundary.

One of the main drawbacks of our work is its limitation to binary sentence classification. However, multi-class classification in an one-vs-rest schema is compatible with our method and worth further exploration. Another interesting direction for future work is the synthesis of data for more complex tasks like Natural Language Inference (NLI) or QA. This, however, requires modifications to the structure of the autoencoder and exceeds the scope of this work.

Membership Query Synthesis might also be an interesting approach for tasks where the automatic extraction of large amounts of unlabelled data is not straight-forward. One example that comes to mind is the detection of offensive language or 'hate speech', where we have to deal with highly unbalanced training sets with only a small number of positive instances, and attempts to increase this number have been shown to result in systematically biased datasets (Davidson et al., 2019; Wiegand et al., 2019). Table 2 suggests that the generator produces instances with a more balanced class ratio (1.7 and 1.2) than the pool data (2.6) it was trained on. It might be worthwhile to explore whether the generation of synthetic training instances can help to mitigate the problem to select instances from both classes in an highly imbalanced data pool.

8 Conclusion

This work is the first to show that Membership Query Synthesis in an NLP setting is feasible. Our approach uses a Variational Autoencoder as a representation learner and generates informative active learning queries from latent space. The classification performance for the generated instances is competitive with pool-based active learning strategies and outperforms other AL strategies with regard to annotation cost (time) and computational complexity.

The main advantage of Membership Query Synthesis for active learning is that it allows us to target specific points along the separating hyperplane and thus to provide the classifier with information on specific areas of uncertainty in the data space. While pool-based active learning has the same objective, Membership Query Synthesis gives us a more precise tool to explore the data space and to generate exactly those instances that we need, making MQS a promising approach for future work in active learning.

Acknowledgments

Part of this research has been conducted within the Leibniz Science Campus "Empirical Linguistics and Computational Modeling", funded by the Leibniz Association under grant no. SAS-2015-IDS-LWC and by the Ministry of Science, Research, and Art (MWK) of the state of Baden-Württemberg.

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In *The 57th Conference of the Association for Computational Linguistics*, ACL 2019, pages 6168–6173.

Dana Angluin. 1988. Queries and concept learning. *Machine Learning*, 2(4):319–342.

Pranjal Awasthi and Varun Kanade. 2012. Learning using local membership queries under smooth distributions. *CoRR*, abs/1211.0996.

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364.

Thomas Davidson, Debasmita Bhattacharya, and Ingmar Weber. 2019. Racial bias in hate speech and abusive language detection datasets. In *The Third Workshop on Abusive Language Online*, pages 25–35.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Tomoki Hayashi, Shinji Watanabe, Yu Zhang, Tomoki Toda, Takaaki Hori, Ramón Fernández Astudillo, and Kazuya Takeda. 2018. Back-translation-style data augmentation for end-to-end ASR. In *2018 IEEE Spoken Language Technology Workshop*, SLT 2018, pages 426–433.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Controllable text generation. *CoRR*, abs/1703.00955.

Miriam W. Huijser and Jan C. van Gemert. 2017. Active decision boundary annotation with deep generative models. *CoRR*, abs/1703.06971.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Diederik P. Kingma, Danilo Jimenez Rezende, Shakir Mohamed, and Max Welling. 2014. Semi-supervised learning with deep generative models. *CoRR*, abs/1406.5298.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *CoRR*, abs/1506.06726.

Kevin Lang and Eric Baum. 1992. Query learning can work poorly when a human oracle is used. *IEEE Intl. JointConference on Neural Networks*.

Charles X. Ling and Jun Du. 2008. Active learning with direct query construction. In *The 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 480–487.

Arash Mehrjou, Mehran Khodabandeh, and Greg Mori. 2018. Distributionaware active learning. *arXiv preprint*, abs/1805.08916.

Dai Quoc Nguyen, Dat Quoc Nguyen, Thanh Vu, and Son Bao Pham. 2014. Sentiment classification on polarity reviews: An empirical study using rating-based features. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 128–135.

Augustus Odena. 2016. Semi-supervised learning with generative adversarial networks. *CoRR*, abs/1606.01583.

Ofir Press and Lior Wolf. 2016. Using the output embedding to improve language models. *CoRR*, abs/1608.05859.

Alec Radford, Rafal Józefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *CoRR*, abs/1704.01444.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *The 54th Annual Meeting of the Association for Computational Linguistics*, ACL 2016.

Burr Settles. 2009. Active learning literature survey. Technical report, Computer Sciences Technical Report 1648, University of Wisconsin-Madison.

Dinghan Shen, Asli Celikyilmaz, Yizhe Zhang, Liqun Chen, Xin Wang, and Lawrence Carin. 2019. Hierarchically-structured variational autoencoders for long text generation.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J. Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. *CoRR*, abs/1804.00079.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Liantao Wang, Xuelei Hu, bo Yuan, and Jianfeng Lu. 2015. Active learning via query synthesis and nearest neighbour search. *Neurocomputing*, 147:426434.

Michael Wiegand, Josef Ruppenhofer, and Thomas Kleinbauer. 2019. Detection of abusive language: the problem of biased datasets. In *The 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT 2019, pages 602–608.

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2019. Code-switched language models using neural based synthetic data from parallel sentences. In *The SIGNLL Conference on Computational Natural Language Learning*, CoNLL 2019.

Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. 2017. Variational autoencoder for semi-supervised text classification. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Jia-Jie Zhu and José Bento. 2017. Generative adversarial active learning. *CoRR*, abs/1702.07956.