Textual Membership Queries

Jonathan Zarecki* and **Shaul Markovitch**
Department of Computer Science, Technion - Israel Institute of Technology
szarecki@cs.technion.ac.il, shaulm@cs.technion.ac.il

Abstract

Human labeling of data can be very time-consuming and expensive, yet, in many cases it is critical for the success of the learning process. In order to minimize human labeling efforts, we propose a novel active learning solution that does not rely on existing sources of unlabeled data. It uses a small amount of labeled data as the core set for the synthesis of useful *membership queries* (MQs) — unlabeled instances generated by an algorithm for human labeling. Our solution uses *modification operators*, functions that modify instances to some extent. We apply the operators on a small set of instances (core set), creating a set of new membership queries. Using this framework, we look at the instance space as a search space and apply search algorithms in order to generate new examples highly relevant to the learner. We implement this framework in the textual domain and test it on several text classification tasks and show improved classifier performance as more MQs are labeled and incorporated into the training set. To the best of our knowledge, this is the first work on membership queries in the textual domain.

1 Introduction

Machine learning algorithms require sufficient labeled data to produce a high-quality model. However, collecting labeled data can pose a significant challenge in some problem domains and may require human labor. To reduce the labeling cost, instead of passively accepting and using a labeled training set, the learning algorithm may *actively* request the labeling of instances that are estimated to be useful. This approach is called *active learning* (AL).

One of the first theoretical models proposed for active learning was the *membership queries* (MQs) model [Angluin, 1988]. In this setting, the learner may request a label for any unlabeled instance in the instance space, including queries (instances) that the learner generates from scratch. This approach holds strong theoretical promise as its learning model is more robust than the standard PAC [Valiant, 1984] learning model in many cases [Bshouty, 1995; Angluin, 1987], since it does not depend on the assumption that the training and testing instances are drawn from the same distribution.

There is one major obstacle, however, when using the MQ model for real-world learning tasks. Traditionally, training instances, such as images or natural-language sentences, are mapped into feature vectors. MQ-based learning algorithms may generate a query in the form of a feature vector, but a human labeler may not be able to classify it as she can only process an instance (e.g. an image or a sentence). Mapping a feature vector to a recognizable instance may also prove to be very difficult. Consider for example mapping a vector of word frequencies into a text fragment, or mapping a vector of image features into a recognizable image.

Additionally, even if a mapping from feature vectors to instances exists, it may map into instances that are outside of the domain of the classifier. To illustrate this problem, let us look at a flower classification task, where the input is an $N \times N$ black-and-white image of a flower and our goal is to classify it to the correct flower species. Assume that we use the pixels as binary features, thus having $2^{N \times N}$ possible feature vectors. In this case, there is an easy mapping from a feature vector to an image, but the vast majority of the resulted images are not of a flower and are therefore not classifiable. Lang and Baum [1992] encountered a similar problem in their early attempt to use MQs for handwritten digit recognition. Given two digits, they combined them into a new image and queried a human oracle for classification. This process, however, often resulted in an unrecognizable digit which the human labeler could not process. Furthermore, the feature-to-instance mapping is potentially not a valid function, since the mapping induced by the features is not necessarily 1-to-1. In the textual domain, for example, the bag of words (BOW) feature representation [Harris, 1954], is used to represent textual instances. Under this mapping, two different instances may be represented by the same feature vector. Thus, the feature vector associated with the bag {bites, dog, man } may be mapped back to the instances "Man bites dog" and "Dog bites man" with obviously different meanings.

To address these issues, the pool-based [Lewis and Gale, 1994] and stream-based [Atlas *et al.*, 1990] approaches for active learning were suggested. The pool-based approach assumes that a collection (pool) of unlabeled instances is

*Contact Author

available. The instances in the pool are evaluated by the learning algorithm using their feature-vector representation, and the instance estimated to be most useful is presented to the labeler. The stream-based approach assumes the data is presented as a stream of instances, and the learner chooses whether to request the label for the presented instance in a similar fashion.

There are, however, real-world scenarios where using pool-based or stream-based AL can be problematic:

1. A set of unlabeled instances is not available.
2. The learned concept is a very small portion of the instance space, leading to very limited availability of positive examples. For example, assume that our task is to find tweets posted by terrorist organizations. As these organizations do not post regularly and not all tweets are found, positive examples can be difficult to find even in large pools of data. Another type of such problem domains are tasks where we want to classify a behavior within a specific subgroup, where the total number of examples is small. A concrete example of this is offensive language classification in Reddit and its subreddit communities. This behavior is different within every community and we will need specific examples to correctly classify toxic users within each one, it may not be possible to collect enough data in this case.
3. There is a set of unlabeled instances, but the set is too large. One such example is identifying tweets on a specific subject, as the entire pool of tweets is incredibly large, using pool-based approaches to find relevant examples from the entire Twitter data is impractical, especially when using sophisticated AL methods that are computationally expensive [Houlsby *et al.*, 2011; Lindenbaum *et al.*, 1999].

These problems, associated with pool-based active learning and stream-based active learning, do not affect the MQ approach. In all of these scenarios, membership queries will be able to generate useful examples while avoiding the issues.

In this work, we present a new general and practical methodology for generating membership queries. Our work tackles the problems in membership query synthesis and presents a novel algorithm for synthesizing new instances by defining a search space over the set of instances, actively searching for the most informative instances. Our method is capable of generating near-misses — negative examples that are only slightly different from the positive ones. Such examples have been shown to be helpful for learning [Gurevich *et al.*, 2006].

Our main contributions are as follows:

1. We present a new, practical way of synthesizing high-quality membership queries by defining a search space over the instance space.
2. We present such a search space over the textual domain, defining its operators and implementing algorithms that utilize them for the generation of new textual membership queries.
3. We present an experimental methodology for testing MQ-related algorithms on real data.

2 A Framework for Generating Membership Queries

In this section, we present our framework for generating membership queries. We first formalize the learning setup.

2.1 Learning Setup

Let Φ be a set of instances, called the *instance space*. Let o: $\Phi \rightarrow \{0, 1\}$ be an oracle that can label instances from Φ as positive (1) or negative (0)[1]. Let $F = f_1, \cdots, f_N$ be a set of feature functions, where $f_i : \Phi \rightarrow D_i$ and D_i is the range of f_i. We denote the space spanned by F, $\Psi = D_1 \times \cdots \times D_N$, as the *feature space*. Let $x \in \Phi$ be an instance. We denote $f_\Psi(x) = \langle f_1(x), \cdots, f_N(x) \rangle$ as the *feature vector* for x.

It is important to note that we define o, the labeling function, to be independent of F. We can replace F by an alternative set of feature functions without affecting the instance labels. Labeling takes place in the instance space while learning takes place in the feature space. In text classification, for example, a human labeler (o) is only able to classify well-structured sentences and will find it impossible to label feature-based representations such as Word2Vec [Mikolov *et al.*, 2013] embeddings.

2.2 Membership Query Synthesis in the Instance Space

Our query synthesis framework assumes that for a given learning task, we are given the following components:

1. C: a core set of labeled instances.
2. K: the desired number of MQs.
3. O: a set of modification operators on instances, where for each $op \in O$, $op : \Phi \rightarrow \Phi$.

The quality of the modification operators is crucial to the performance of our algorithm. For the algorithm to perform well, the modification operators must be able to create a diverse set of new instances given an input, while keeping their output within the instance space Φ. Given a set of modification operators O, we can define the closure of a given core set C as: $cl(C) = \{s \mid \exists c \in C, op_1, \cdots, op_n \in O[op_n \circ \cdots \circ op_1(c) = s]\}$. We would like to define the operators such that $cl(C)$ is as large and diverse as possible, such that even with a small core set, $cl(C)$ will be useful as a source for high-utility MQs. We would also like a single operator to minimally change the instance, as this will enable the generation of near-miss examples by consecutively applying operators which will slowly move the instance over the classification boundary. We will later discuss the implementation of the modification operators for the textual domain.

Stochastic Query Synthesis

A simple way of utilizing the modification operators is to apply them in random to the core set of instances. The algorithm maintains a set of instances Ω. We initialize Ω with the core set C. At each step, we randomly choose an instance from Ω

[1]For simplicity we describe only binary classifications, but our framework can generalize to multi-class problems.

Algorithm 1: Stochastic query synthesis

Input : C - a core set of initial instances
O - s set of modification operators
K - the required number of new MQs
Output: A set of K membership queries

```
Ω = C;
while |Ω\C| < K do
    new_inst = Apply(RandSelect(O),RandSelect(Ω)) ;
    Ω = Ω ∪ {new_inst} ;
return Ω\C ;
```

and apply a random operator to it. The resulting new instance is added to Ω. When enough instances have been generated, we return $\Omega \backslash C$ as the MQs. The pseudo-code for stochastic query synthesis is listed in Algorithm 1.

Query Synthesis using Search Algorithms

The stochastic synthesis algorithm can be improved by treating the instance space as a search space and actively seeking more informative instances to generate. We define the instance search space as follows: The state space is the instance space, the actions are the modification operators and the heuristic value is given by an evaluation function.

Possible candidates for such heuristics are existing active-learning functions such as Uncertainty Sampling [Lewis and Gale, 1994]. These functions are designed to assign higher values to feature vectors that are estimated to be more informative. Given such a function, $U_{al} : \Psi \rightarrow \mathbb{R}$, we can compose it with the feature mapping function $f_\Psi(x) : \Phi \rightarrow \Psi$ to get instance evaluation function: $U(x) = U_{al}(f_\Psi(x))$.

Now we can use any resource-bounded (RB) heuristic search algorithm, such as greedy best first and beam search, to look for informative queries, while enforcing resource-bounds in terms of time and memory. The search-based MQ-generation algorithm is listed in Algorithm 2.

Algorithm 2: Search-based query synthesis

Input : C - a core set of initial instances
O - s set of modification operators
K - the required number of new MQs
U - instance evaluation
Search - RB heuristic search algorithm
Output: A set of K membership queries

```
Ω = C;
while |Ω\C| < K do
    i = RandSelect(Ω)) ;
    new_inst = Search(i,O,U) ;
    Ω = Ω ∪ {new_inst} ;
return Ω\C ;
```

3 Generating Textual Membership Queries

In this work, we focus on using our approach in the textual domain. At this stage we limit our study to the classification of sentences, but we plan to extend it to larger textual objects. As discussed in the introduction, generating instances in the textual domain is especially problematic. Synthesized sentences can easily become unreadable when not treated carefully. Furthermore, common feature representations, such as BOW [Harris, 1954], are not surjective: there are feature vectors that do not map to any instance. For example, the feature vector {"man", "dog", "cat"} does not map into any legal sentence. These representations are also not injective: there is not a one-to-one correspondence between the instance space and the feature space. As we saw in the introduction, the sentences "man bites dog" and "dog bites man" will receive the same BOW vector. These limitations prevent us from generating examples in the feature space and mapping them back to the instance space to be tagged.

To apply our methodology to the textual domain, we need to define the instance space and the modification operators (our methodology is independent of the feature space). We define the instance space to be the set of all syntactically and semantically legal sentences in English. To define modification operators for textual objects, we first define a *semantic distance function* between terms and then define the *semantic neighborhood* of a term. We will then define modification operators that replace words by other words within their semantic neighborhood.

3.1 Computing the Semantic Distance

For a word to serve as a possible replacement within a specific context, it has to be *functionally similar* [Turney and Pantel, 2010] to the original, meaning that the two words *behave* similarly in their context. In order for our operators to be able to generate near-misses they should strive to make minimal changes to the instance, thus we will to enforce our operators to suggest words that are also *semantically related*, reducing the change to the sentence from a single operator.

This requirement makes common methods for computing semantic relatedness (such as ESA [Gabrilovich and Markovitch, 2007] and Word2Vec) unsuitable, as they do not consider functionality. Thus, "wine" may be considered semantically close to "drinking", despite them having different functionality. On the other end, we also considered masked language-models (LM) like BERT [Devlin *et al.*, 2019] but in many cases found it focused mainly on preserving functional similarity and not on semantic relatedness. Instead, we chose to use Dependency Word2vec [Levy and Goldberg, 2014], a representation with an associated distance function that was specifically trained to exhibit both of these properties.

Modification Operators in the Textual Domain. We define a semantic neighborhood of a word w as the set of words that hold related meaning and can be used in similar contexts. Our modification operators use the semantic neighborhood in order to substitute words in a sentence with other words in their semantic neighborhood, generating new legal sentences. Given a semantic distance function, we define the *k-semantic neighborhood* for a particular word w, $N(w, k)$, as the k closest words to w. Using the semantic neighborhood, we can now define the modification operators for a given sentence s. First, all verbs, nouns and adjectives in s are marked as *replaceable words*. Then, the k-semantic neighborhood of each replaceable word is calculated. A candidate operator replaces

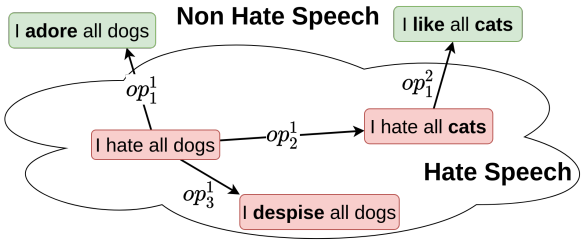

Figure 1: Modification operators for a sentence in the hate-speech detection task

a *replaceable word* with a member of its k-semantic neighborhood. The candidate operators are then filtered by Spacy's latest part-of-speech parser to keep only those that retain the syntactic structure of the original sentence. Note that by consecutively applying the modification operators we are able to significantly change the sentence meaning while each operator minimally changes the sentence's overall theme, building a focused set of new sentences more likely to contain near-misses.

4 Empirical Evaluation

We analyzed the performance of our framework on 5 publicly available sentence classification datasets. The code for all experiments is available here[2].

4.1 Experimental Methodology

We report results on 5 binary sentence classification datasets: three sentiment analysis datasets, one sentence subjectivity dataset, and one hate-speech detection dataset.

CMR: Cornell sentiment polarity dataset [Pang and Lee, 2005]. **SST:** Stanford sentiment treebank, a sentence sentiment analysis dataset [Socher *et al.*, 2013]. **KS:** A Kaggle short sentence sentiment analysis dataset. [3] **HS:** Hate speech and offensive language classification dataset [Davidson *et al.*, 2017]. **SUBJ:** Cornell sentence subjective / objective dataset [Pang and Lee, 2004].

Simulating the Human Oracle

As in most works on active learning, we require a human expert to label our queries. However, because we generate different MQs in every run and need to label these instances every time, a very significant labeling effort is required. To address this problem, we chose to *simulate* a human labeler [Druck *et al.*, 2009]. To make the artificial setting as close as possible to a real-world setting, the artificial expert is a classification model trained on the entire labeled dataset. For each dataset, we chose a model close to the state-of-the-art for the task[4]. This setup is sound as label information is available only when requesting example labels, as in all AL setups. The cross-validation accuracy of each artificial expert on its dataset is 86.8% in CMR, 94.4% in SUBJ, 86.2% in SST, 91.0% in HS, and 94.5% in KS. The expert achieves close to state-of-the-art performance for each dataset. One potential problematic aspect of this methodology is that the performance of the artificial expert was tested only on real examples from the datasets, and not on artificially generated instances. To account for this, we performed the following experiment. We used our algorithm to generate a test set of 100 instances for each of the 5 datasets and asked humans to label these instances. We then applied the artificial expert on these test sets. The accuracy achieved was very close to the one achieved for the real examples (with a maximum of 2% difference). This confirmed that the artificial expert can operate on our artificial examples.

Compared Methods

In the following experiments, we have used 3 variations of our methods: a) **Uncertainty sampling hill-climbing MQ synthesis (US-HC-MQ):** The proposed search-based synthesis method, using hill-climbing search and uncertainty sampling-based heuristic function. b) **Uncertainty sampling beam-search MQ synthesis (US-BS-MQ):** Same as (a) but using the beam search algorithm. **Stochastic Synthesis (S-MQ):** A degenerated version of our method, described in detail in Section 2.2.

As no other work attempted to build membership queries in the textual domain prior to this work, we chose 3 somewhat similar approaches for generation/augmentation of textual instances to compare with: a) **Random Original Examples (IDEAL-RAND):** Randomly select a set of unlabeled examples, label them with original labels and insert them into the model. This model has an unfair advantage of using a pool of unlabeled instances not available to the other methods. It is presented to assume an "lower upper-limit" role, enabling us to see what would happen if we had access to unlabeled examples; b) **RNN Generator (RNN-LM):** A method for generating sentences with an LSTM [Hochreiter and Schmidhuber, 1997] language model. We fine-tuned the models with only the core set of instances. The RNN-LM model was pre-trained on English Wikipedia, and performed conditional generation from the middle of the sentence; c) **WordNet-based data augmentation (WNA):** A possible approach to text augmentation, where words are replaced with their respective synonyms from WordNet [Zhang and LeCun, 2015]. All methods used an environment size of 10, and a linear classifier with an average 300-dim GloVe [Pennington *et al.*, 2014] word-vectors as the learner.

4.2 Experiment 1 - Batch Active Learning with Membership Queries

We measured the performance of our MQ synthesis framework in a batch active learning setup, where at each step, a pool of unlabeled instances is generated, and then m (batch size) samples are chosen to be labeled and incorporated into the core set. In each step, we use a generation algorithm to return a pool of unlabeled instances with size P. Then a heuristic function U is used to extract the m most informative instances

[2] www.github.com/jonzarecki/textual-mqs

[3] www.kaggle.com/c/si650winter11

[4] For SST, CMR, SUBJ & KS we used the open-source implementation of Generating Reviews and Discovering Sentiment [Radford *et al.*, 2017], which achieved state-of-the-art results for CMR and SUBJ. For KS it achieved 94% accuracy. For the hate-speech (HS) dataset we used a BOW-based classifier, which achieved 91% accuracy.

in a batch to be labeled by the expert. These labels are then incorporated into the training set and used to train a model. The model's accuracy is then measured against the test set. We repeat the experiment 20 times with different core sets and report the average results on all available datasets.

Figure 2 plots the accuracy curves as a function of the number of queries generated by our algorithm or by the competitor methods. We used a core set of 10 sentences, a pool size of 20, an AL batch size of 5, and the uncertainty sampling-based [Lewis and Gale, 1994] heuristic function as U for all experiments. We chose a small core set size to emphasize our algorithm's ability to generate useful sentences even in this challenging setting.

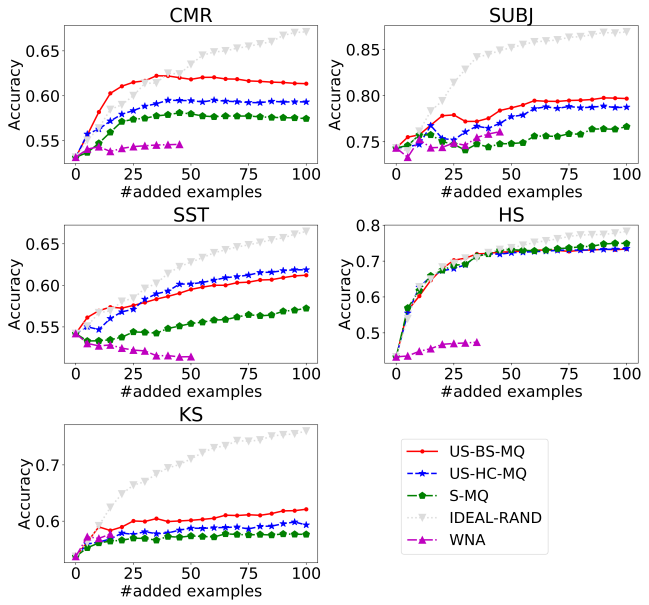

Figure 2: Comparison of accuracy achieved by the different methods

The two search-based approaches (US-HC-MQ and US-BS-MQ) both exhibited excellent performance across the 5 datasets. The comparison of the search-based approaches to S-MQ showed that, as we expected, more valuable examples are obtained when using the utility function in the generation process. WNA performed admirably considering that in principle it is using only a small semantic neighborhood and therefore receives only synonyms. However, its lack of diversity resulted in low accuracy on some datasets. Another significant disadvantage results from WNA using only synonyms is the limited amount of synonyms available from WordNet, making it unable to generate a large pool of instances. In comparison, our method can theoretically generate as many instances as required. RNN-LM's results were not shown as many sentences generated using this approach were marked as invalid by human labelers, as this was a requirement for our AL setup we were forced to remove them. Additional information is presented in 4.4.

4.3 Experiment 2 - The Effect of the Synthesis Algorithm on Label Switches

In our framework, after applying the modification operators, the resulting instances are able to change their original label and thus cross the classification boundary, becoming "near-misses" [Gurevich *et al.*, 2006], examples that originally belonged to a certain class, but our sequence of modification operators caused them to switch their original class without significant changes to the instance.

In this experiment, we tested the effect of our synthesis algorithms on the number of label changes they cause. Three algorithms were compared, Uncertainty hill-climbing (US-HC-MQ), Stochastic hill-climbing (S-HC-MQ), and Stochastic synthesis (S-MQ). US-HC-MQ uses a heuristic function to direct its search, S-HC-MQ applies multiple operators randomly, and S-MQ randomly applies only one operator at a time, just as described in Algorithms 1 and 2. We randomly chose a core set of 10 instances for each dataset and used each synthesis algorithm to generate 50 examples. The score for each algorithm is the portion of examples it generated that crossed the classification boundary. We repeat the experiment 20 times with different core sets and show the average results on all available datasets.

Figure 3 shows a clear hierarchy, where US-HC-MQ has the most label changes, followed by S-HC-MQ, and then by S-MQ. This result reinforces our hypothesis that using multiple operators on a single sentence as well as using heuristic functions during generation results in more diverse sentences as well shows that our framework does not only perform augmentations but significantly changes the meaning of the sentences.

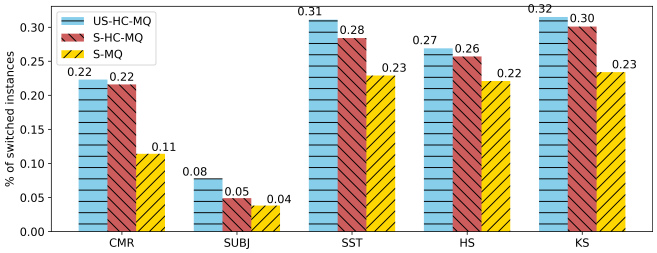

Figure 3: The effect of the synthesis algorithm on the number of changed labels

4.4 Human Evaluation - Sentence Generation

In order to test the quality of our model's generated sentences, we performed a simple readability test using university students as evaluators. 500 sentences generated by our US-BS-MQ, 500 sentences created by the pretrained RNN-LM and 500 original sentences from the data sets used were randomly selected for the test, human evaluators were asked whether each sentence was fully comprehensible to them. We performed 3000 of these tests where each sentence was shown twice. Aggregating the results, 96% of the original sentences and 95% of our sentences were labeled as "comprehensible", while the RNN-LM's sentences were labeled "comprehensible" only 21% of the time. This shows that our artificial sen-

tences are almost indistinguishable from real ones. This finding forced us to remove RNN-LM from experiment 1, as it was unable to generate valid instances for labeling, a requirement for an AL setup.

5 Qualitative Analysis

Let us look more carefully at the instances generated by our membership queries and the sequence of modification operators that were applied. Let us look at example sequences:

- S_0 : That's why I **love** Tangled, apart from Moore.
  $\downarrow$ op_1: love $\rightarrow$ despise
  S_1 : That's why I **despise** Tangled, apart from Moore.
- S_0 : I thought Star Wars was really **boring**.
  $\downarrow$ op_1: boring $\rightarrow$ fascinating
  S_1 : I thought Star Wars was really **fascinating**.
- S_0 : Never tell a **bitch** what u up to.
  $\downarrow$ op_1: bitch $\rightarrow$ slob op_2: slob $\rightarrow$ bookworm
  S_2 : Never tell a **bookworm** what u up to.

These sequences demonstrate the ability of our MQ framework to direct the word switches toward a sentence with a different and sometimes negative meaning, a result seen clearly in Experiment 2. Now, let us look at an example where our framework did not generate legal sentences.

- S_0 : Da Vinci Code sucked, as **expected**.
  $\downarrow$ op_1: expected $\rightarrow$ forecasted op_2: forecasted $\rightarrow$ preprogrammed
  S_2 : Da Vinci Code sucked, as **preprogrammed**.

In the negative examples, we see that switching a word multiple times can harm the resulting sentence's quality. We see "expected" being replaced with "forecasted", a suitable switch, and then with "preprogrammed" which is not related to the original meaning of the sentence. A possible solution is to use multi-sense embeddings [Iacobacci *et al.*, 2015] in conjunction with Dependency Word2vec to create different embeddings for each sense of the requested word while preserving the important functional similarity.

6 Related Work

In this section we will further discuss three works that are relevant to our method.

X-local membership queries were first introduced by Awasthi et al. [Awasthi *et al.*, 2013], defined as a query to any point for which there exists a point in the training sample with a Hamming distance lower than X. Bary [2015] built on this idea and proved that even a learning model that uses only 1-local MQs (with Hamming distance 1 to a training example) is stronger than the standard PAC learning model. In addition, Bary also employed a method similar to 1-local queries to gather information for the task of sentiment analysis.

However, neither Bary or Awasthi applied the idea of local membership queries to instances. Rather, they applied this idea to the feature vectors representing those queries. As we discussed in the Learning Setup subsection, it is usually impossible to present an altered feature vector to a human oracle, as was done in these works. Thus the idea of local membership queries has remained mainly theoretical.

Gurevich et al. [2006] presented the idea of modification operators that remain in the instance space. In contrast to the operators applied by Awasthi and Bary, these operators are applied to instances, easily read by a human expert, and return other instances. This work used a small seed of only positive instances to model the entire instance space, generating "near-miss examples" in order to effectively model the vast space of negative instances. However, this work was applicable only to the image domain, and its operators obviously could not be applied to textual sentences. Nor did this work fully explore the options of generation when using the modification operators, such as using search algorithms and heuristics during the MQ generation process.

Several works have discussed the topic of textual data augmentation [Zhang and LeCun, 2015; Rosario, 2017], where existing examples from the training set are augmented into other very similar instances. However, the augmented instances are not allowed to change their class and so are limited to synonyms which limits the variety of instances they generate. Indeed, our empirical evaluation showed Zhang & LeCun's method [Zhang and LeCun, 2015] to be less effective than our suggested methods.

7 Conclusions

The goal of this work was to show membership queries in a more practical light. We presented a novel *modification operator*-based framework for generating membership queries that are within the instance space and recognizable to the human oracle. Using this framework we presented a local-search-based algorithm for generating MQs that use information from an additional utility function to direct the search to highly valuable instances. We implemented our approach in the textual domain and demonstrated that our modification operators will result in legal sentences. We evaluated our methods on several datasets, finding high accuracy gains when using MQs. Somewhat surprisingly, the approach is sometimes competitive with an approach that utilizes a pool of unlabeled instances not available to our MQ framework.

Our results motivate a few interesting directions for future work that we plan to explore. First, our approach can be extended to replacing larger components of text. In the future, we intend to replace larger grammatical components such as noun phrases and verb phrases. To do so, however, we first need to develop methods for generating syntactically and semantically well-structured phrases. Second, our method can be also used for data augmentation as an alternative to methods of oversampling such as SMOTE [Chawla *et al.*, 2002] that work in the feature space. By restricting our modification operators to a very tight semantic neighborhood that uses only synonyms, the label of the modified instances is not likely to change, and we can use them for augmentation. Third, we plan to explore ways to enrich existing pools of unlabeled instances using heuristic-guided local search in a method similar to the one used here, which will result in new high-quality unlabeled examples to choose from. This direction may enhance the results of existing methods for pool-based AL.

References

[Angluin, 1987] Dana Angluin. Learning Regular Sets from Queries and Counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.

[Angluin, 1988] Dana Angluin. Queries and Concept Learning. *Machine Learning*, 2(4):319–342, 1988.

[Atlas *et al.*, 1990] Les E Atlas, David A Cohn, and Richard E Ladner. Training Connectionist Networks with Queries and Selective Sampling. In D S Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 566–573. Morgan-Kaufmann, 1990.

[Awasthi *et al.*, 2013] Pranjal Awasthi, Vitaly Feldman, and Varun Kanade. Learning using local membership queries. *COLT*, 30:1–34, 2013.

[Bary, 2015] Galit Bary. *Learning Using 1-Local Membership Queries*. PhD thesis, The Hebrew University of Jerusalem, 2015.

[Bshouty, 1995] N H Bshouty. Exact Learning Boolean Functions via the Monotone Theory. *Information and Computation*, 123(1):146–153, 1995.

[Chawla *et al.*, 2002] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[Davidson *et al.*, 2017] Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *ICWSM*, 2017.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[Druck *et al.*, 2009] Gregory Druck, Burr Settles, and Andrew McCallum. Active Learning by Labeling Features. EMNLP '09, pages 81–90, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[Gabrilovich and Markovitch, 2007] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, 2007.

[Gurevich *et al.*, 2006] Nela Gurevich, Shaul Markovitch, and Ehud Rivlin. Active learning with near misses. In *AAAI*, 2006.

[Harris, 1954] Zellig S. Harris. Distributional Structure. *Word*, 10(2-3):146–162, 1954.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[Houlsby *et al.*, 2011] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *ArXiv*, abs/1112.5745, 2011.

[Iacobacci *et al.*, 2015] Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. Sensembed: Learning sense embeddings for word and relational similarity. volume 1 of *EMNLP '09*, pages 95–105, 2015.

[Lang and Baum, 1992] Kevin J. Lang and Eric B Baum. Query Learning Can Work Poorly when a Human Oracle is Used. In *IJCNN 1992*, pages 335–340, 1992.

[Levy and Goldberg, 2014] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *ACL*, 2014.

[Lewis and Gale, 1994] David D Lewis and William A Gale. A Sequential Algorithm For Training Text Classifiers. *Proceedings of the 17th International Conference on Research and Development in Information Retrieval (SIGIR'94)*, 94:3–12, 1994.

[Lindenbaum *et al.*, 1999] Michael Lindenbaum, Shaul Markovitch, and Dmitry Rusakov. Selective sampling for nearest neighbor classifiers. *Machine Learning*, 54:125–152, 1999.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[Pang and Lee, 2004] Bo Pang and Lillian Lee. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In *ACL*, 2004.

[Pang and Lee, 2005] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, 2005.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[Radford *et al.*, 2017] Alec Radford, Rafal Józefowicz, and Ilya Sutskever. Learning to Generate Reviews and Discovering Sentiment. *CoRR*, abs/1704.0, 2017.

[Rosario, 2017] Ryan Rosario. *A Data Augmentation Approach to Short Text Classification*. PhD thesis, University of California, Los Angeles, 2017.

[Socher *et al.*, 2013] Richard Socher, Alex Perelygin, and Jy Wu. 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, 2013.

[Turney and Pantel, 2010] Peter D. Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188, 2010.

[Valiant, 1984] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 11 1984.

[Zhang and LeCun, 2015] Xiang Zhang and Yann LeCun. Text Understanding from Scratch. *CoRR*, abs/1502.0, 2015.