Textual Membership Queries

Jonathan Zarecki
Textual Membership Queries

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

Jonathan Zarecki

Submitted to the Senate
of the Technion — Israel Institute of Technology
Tamuz 5778 Haifa June 2018
This research was carried out under the supervision of Prof. Shaul Markovitch, in the Faculty of Computer Science.

This research was not yet published in conferences and journals.

Acknowledgements

I would like to thank my advisor Prof. Shaul Markovitch. Shaul took me in very early in my Bachelor degree and lead me through a fascinating journey in the world of research.

In addition I would like to thank my parents which supported me throughout this journey and especially in it’s more challenging parts.

The Technion’s funding of this research is hereby acknowledged.
# Contents

List of Figures

Abstract

1 Introduction

2 Active Learning & Membership Queries

3 A Framework for Generating Textual Membership Queries
   3.1 Learning Setup
   3.2 Membership Query Synthesis in the Instance Space
      3.2.1 Modification Operators
      3.2.2 Instance Evaluation Functions
      3.2.3 Stochastic Query Synthesis
      3.2.4 Query Synthesis using Search Algorithms
   3.3 Textual Instance Space
      3.3.1 Modification Operators in the Textual Domain
      3.3.2 Computing the Semantic Distance

4 Empirical Evaluation
   4.1 Experimental Methodology
      4.1.1 Datasets
      4.1.2 Simulating the Human Oracle
      4.1.3 Experiment Design
      4.1.4 Compared Methods
   4.2 Experiment 1 - Batch Active Learning with Membership Queries
   4.3 Experiment 2 - The Effect of the Number of Operators Applied
   4.4 Experiment 3 - The Effect of the Synthesis Algorithm on Label Switches
   4.5 Qualitative Analysis

5 Related Work

6 Conclusions
# List of Figures

1.1 Diagram illustrating the difference between pool-based AL and MQ synthesis .................................................. 5

3.1 Modification operators for an instance .......................................... 10

3.2 Modification operators for a sentence ....................................... 14

4.1 Experiment Design Diagram ...................................................... 19

4.2 Comparison of accuracy achieved by the different methods
   HC-MQ (red circle), US-BS-MQ (blue star), S-MQ (green pentagon),
   IDEAL (gray down triangle), WNA (purple up triangle) ..................... 22

4.3 Final scores of all the approaches vs. IDEAL’s final score in each dataset 23

4.4 Friedman test diagram [Dem06] for the results with p-value ≤ 0.05, showing a statistical significance for search-based methods over the baseline method WNA ................................................................. 23

4.5 The number of operators vs. the accuracy of the final model ........... 24

4.6 The effect of the synthesis algorithm on the number of changed labels ... 25
Abstract

Human labeling of textual data can be very time-consuming and expensive, yet it is critical for the success of an automatic text classification system. In order to minimize human labeling efforts, we propose a novel active learning (AL) 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 synthesized by an algorithm for human labeling.

Our solution uses modification operators, functions from the instance space to the instance space that change the input to some extent. We apply the operators on the core set, thus 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 create desirable MQs. We implement this framework, along with its modification operators, in the textual domain. The implementation includes using methods such as WordNet and Word2vec, for replacing text fragments from a given sentence with semantically related ones.

We test our framework on several text classification tasks and show improved classifier performance as more MQs are labeled and incorporated into the training set. Furthermore, we show that our MQs can be competitive with a more traditional pool-based active-learning approach, which requires an additional pool of unlabeled instances. To the best of our knowledge, this is the first work on membership queries in the textual domain.
Chapter 1

Introduction

As text data becomes a major and highly accessible information source, many research efforts have been directed to the text classification task in an attempt to extract useful information. Practical applications in widespread use today include sentiment analysis, e-mail spam classification and document filtering [AZ12, YJTWC09]. In these applications, supervised learning algorithms use labeled documents to construct a classification model that predicts a class value given an unlabeled instance.

Learning algorithms require sufficient labeled data to produce a high-quality model. However, getting labeled data can pose a significant challenge in some problem domains and may require human labor. We can ask which examples will benefit the algorithm the most and classify those alone in an attempt to reduce labeling efforts. This question is addressed in the field of Active Learning [Set09], where we assume the existence of an oracle, capable of labeling any instance. Under these settings, an active learning algorithm (the “learner”) chooses which queries to present to the oracle, actively trying to choose only the most informative ones.

One of the first theoretical models proposed for active leaning was membership queries (MQs) [Ang88]. In this setting, the learner may request a label for any unlabeled instance in the instance space, including queries that the learner generates from scratch. This approach holds strong theoretical promise as its learning model is more robust than the standard PAC [Val84] learning model in many cases [Bsh95, Ang87], since it does not depend on the class distribution.

However, the MQ model suffers from a major drawback. Learning algorithms usually work in the feature space, whereas human labelers, work in the instance space. Hence, human oracles cannot evaluate and label new instances in the feature space. While the feature functions map from the instance space to the feature space, the reverse mapping usually does not exist. Therefore, MQs generated by the learner in the feature space are useless in classification tasks.

To illustrate this problem, let us look at a flower classification task, where the input is a black-and-white image of a flower and our goal is to classify it to the correct flower species present in the image. The instance space for this task is the set of all
possible black-and-white flower images. The feature space is all $2^{N \times N}$ binary matrices, representing all possible $2^{N \times N}$ binary feature vectors. Obviously, a major part of this feature space cannot be mapped into the instance space, and as such cannot be classified by a human oracle. A well-known illustration of this problem is the early work of Baum and Lang [LB92], who attempted to use MQs for handwritten digit recognition. Given two digits, they combined them into a new image and queried a human oracle for a classification. This process, however, often resulted in an unrecognizable digit. Since then, membership queries have been researched in a practical setting in several works [Das04, LG94], but most of the research has remained theoretical.

Another major problem with membership queries is that the mapping between the feature space (what learning algorithms receive as input) and the instance space (the domain recognizable for humans) is not 1-to-1. In the textual domain, for example, the bag of words (BOW) feature representation [Har54], is used to represent textual instances, and two different instances can be represented using the same feature vector. Thus, the sentences “Man bites dog” and “Dog bites man” would be represented in the same way despite their obviously different meanings. Moreover, sentences created from a feature vector may not be (and usually are not) legal sentences. The BOW vector with the words “I”, “went”, “to”, “school” can be converted to the sentence “School went I to”, which is not a syntactically correct English sentence. The problem is even more severe when including sentences such as ”The book ate the computer”, which are syntactically but not semantically correct.

To address these problems, the pool-based and stream-based approaches for active learning were suggested. Pool-based active learning [LG94] is the most popular approach and has been studied in many real-world machine learning tasks, such as text classification [LC94, HJL06], information extraction [TCM99] and image classification [CT02]. The approach assumes that a collection (pool) of unlabeled instances is available. Queries are selected from the pool, usually in a greedy fashion, to extract the instance estimated to be most useful. Pool-based active learning does pose some challenges. It requires a large pool of unlabeled data from an outside source, and its performance is highly dependent on the quality (with respect to the learning process) of the instances in the pool. A diagram comparing membership query synthesis to pool-based AL is shown in Figure 1.1.

Stream-based active learning [ACL90] assumes the data is presented as a stream of instances, and the learner chooses whether to request the label for the presented instance. Stream-based active learning suffers from similar drawbacks as it is also reliant on the quality of instances from an outside source.

There are, however, real-world scenarios where not enough data can be collected or the quality of the sampled instances is not sufficient. One example is domains where the data is extremely rare. Another is where negative instances are distributed widely across the instance space and most of them are located very far from the classification boundary. Such instances are poor in quality and hence are not beneficial to the learning
algorithm [GMR06]. In both scenarios, classic learning models will find it difficult to generalize.

These problems, associated with pool-based active learning and stream-based active learning do not affect the third type of active learning - membership queries. We have seen, however, that algorithmic generation of MQs is a difficult problem for which a practical algorithmic solution has not been found.

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 practical algorithm for synthesizing new instances. We define a set of operators over the instance space, where each operator maps an instance into another instance (within the instance space). Using these operators, we explore several methods for generating new members of the instance space. We present an implementation of a system for instance generation on textual data, and show that our model performs better than other MQ approaches on several datasets.

Our main contributions are as follows:

1. We present a new, practical way of synthesizing high-quality membership queries.
2. We present and study the instance space as a search space and define operators over this space.
3. We present the textual space as an instance space, defining its operators and implementing algorithms that utilize the textual instance space for the generation of new textual membership queries.
Chapter 2

Active Learning & Membership Queries

All active learning scenarios involve estimating how useful each unlabeled instance is to the learner [Set09]. Many strategies have been proposed for evaluating unlabeled instances. One common approach is Uncertainty Sampling [LG94], where the learner queries the instances it is least certain about. The Query-by-Committee (QBC) [SOS92] approach maintains a committee $C = \{\theta^{(1)}, \ldots, \theta^{(C)}\}$ of models trained on the current labeled set $L$ but representing different hypotheses. Each committee member votes on the true label of query candidates, and the most informative query is considered to be the instance about which they most disagree. Expected Model Change [LMR04, SCF08] is another general framework where we query the instance that would result in the greatest change to the current model if we knew its label.

Unlabeled instances can either be sampled from a given distribution, as in the pool-based and stream-based active learning approaches, or generated algorithmically using the membership query approach, which is our focus here. “A theory of the learnable” [Val84] provides an early definition of membership queries as examples that do not come from “nature”. This idea was later expanded by Angluin [Ang88], who coined the term ‘membership queries’ (MQ) and laid the theoretical foundations of this approach.

In practice, however, membership queries are almost never used, in part because it is difficult to generate new instances that are recognizable by the human teacher [LG94]. Angluin herself noted that it is difficult for an MQ generating system to build queries it deems relevant to the learning process:

"It may be difficult for the (MQ generating) system to generate fully instantiated cases (simulated X-rays) that embody the particular features the system has decided are relevant" [Ang88].

There have been a few attempts to use membership-query synthesis [KWJ+04, LB92]. One real-world application is the robot scientist project [KWJ+04], where an autonomous machine conducts a series of biological experiments. In each experiment, the setting is
synthesized as a membership query, and the experiment’s result is considered as the label. Because the oracle is not human, it can understand MQs generated in the *feature space*, which also means that the feature space is equivalent to the instance space by definition. Another well-known example is the work of Baum and Lang [LB92] on digit recognition, discussed in the introduction. In this work the human labeler was unable to label the membership queries, as they were not in the instance space.

Recent advances in the field [GMR06, AFK13] propose a solution to the problems raised by Baum [LB92] that modifies existing instances. These modifications restrict the learning algorithms to query only examples that are similar to examples drawn from the distribution. By questioning only these examples, queries will not appear random or artificial.

On the task of image classification Gurevich, Markovitch and Rivlin [GMR06] showed that making controlled local changes to original image instances will result in recognizable images. They also tried to actively move their queries closer to the classification boundary using their local changes, thus creating near-misses — examples that differ from the learned concept in only a few of significant points. Using only these queries resulted in good performance gain for an image classification task.

Theoretical works by Awasthi et al. [AFK13] and Bary [Bar15] introduced the term a $X$-local membership queries, defined as a query to any point for which there exists a point in the training sample with Hamming distance lower than $X$. In these cases the feature space also equivalent to the instance space. Bary proved that even a learning model that uses only 1-local MQs (a query for which there exists a point in the training sample with Hamming distance of 1) is stronger than the standard PAC learning model, providing additional theoretical credibility for the local MQ model. In the task of sentiment analysis, Bary moreover also showed that gathering additional information in a way that resembles 1-local queries is beneficial in practice.

The approach proposed in this paper will be implemented on textual data. While many works have applied pool-based active learning to textual tasks [TK02, MN98, Scu07], we are not aware of any work on membership queries in the textual domain. Several works have discussed textual data augmentation [ZL15, Ros17]: building new text instances based on existing examples from the training set. However, the variety of possible instances is very limited because the augmented instances cannot change their class. The scarcity of work on textual membership queries is mainly due to the difficulty of automatically generating well-formed textual queries [GK17] and the lack of 1-1 mapping between the feature space and the textual domain. Building a legal sentence from a feature vector is therefore not trivial. In this work, we propose a novel approach for synthesizing new membership queries in the textual domain. Our method is described in the following section.
Chapter 3

A Framework for Generating Textual Membership Queries

This research is focused on novel ways of synthesizing membership queries based on expanding a small set of existing labeled instances, which we call a core set. To achieve this goal, we look at the instance space as a search space, which includes different operators, and apply them as a sequence in order to generate new instances. We then utilize local search algorithms in order to maximize a utility function, thus creating highly desirable instances.

Before describing the algorithms and implementation, we shall first formalize the learning setup.

3.1 Learning Setup

As stated above, many works in the active learning field fail to distinguish between the feature space and the instance space. Let us first formally define all the components of an active learning system.

Let \( \Phi \) be a set of instances, called the instance space. Let \( o: \Phi \to \{0, 1\} \) be an oracle that can label instances from \( \Phi \) as positive (1) or negative (0).\(^1\) Let \( f_1(x), \cdots, f_N(x) \) be a set of feature functions, where \( f_i: \Phi \to D_i \). We denote the feature space by \( \Psi = D_1 \times \cdots \times D_N \). We denote \( f_\Psi(x) = (f_1(x), \cdots, f_N(x)) \) as a feature vector for each \( x \in \Phi \). It is important to note that different feature extractors may use different feature spaces.

Let \( S_\Phi = \{X_1, \cdots, X_N\} \subseteq \Phi \) be a set of training instances. The training instances are labeled by an oracle \( o \), producing a training set \( D = \{\langle X_i, o(X_i) \rangle \mid X_i \in S_\Phi \} \), which is then transformed using \( f_\Psi \) to a feature representation \( D_\Psi = \{\langle f_\Psi(X_i), o(X_i) \rangle \mid X_i \in S_\Phi \} \). \( D_\Psi \) is used as input to a learning algorithm \( L \). \( L \) produces a classifier \( L(D_\Psi): \Psi \to \{0, 1\} \), which classifies members from the feature space \( \Psi \) as positive or negative. An active

\(^1\)For simplicity we describe only binary classifications, but our framework can generalize to multi-class problems.
learning evaluation function $U_{al} : \Psi \rightarrow \mathbb{R}$ is used to evaluate a new instance $X_i$ after being transformed into a feature vector in $\Psi$.

These distinctions are important as the oracle $o$ can only classify objects in $\Phi$, and not from the feature space $\Psi$ used by $L(D)$. In the textual domain, $o$ is only able to receive well-structured sentences and will not classify BOW vectors. Furthermore, it is important to note that there is normally no inverse function $f_{\Psi}^{-1} : \Psi \rightarrow \Phi$ that is able to transform feature vectors to instances. In almost all cases, the transformation is not 1-1, and many feature vectors cannot originate from a member of $\Phi$, as discussed with regard to BOW features in the introduction. The lack of an inverse function from $\Psi$ to $\Phi$ implies that we cannot produce desirable feature vectors and then transform them back into instances to be labeled by the oracle.

### 3.2 Membership Query Synthesis in the Instance Space

In this work, we present a framework for automatic generation of local MQs in the instance space $\Phi$. Our framework uses modification operators ($op : \Phi \rightarrow \Phi$) in order to generate new examples from a core set of labeled instances.

#### 3.2.1 Modification Operators

As stated in section 3.1, generating examples in the feature space $\Psi$ and then transforming them into members of the instance space $\Phi$ is usually impossible, forcing us to generate instances exclusively in $\Phi$. We therefore introduce modification operators that are functions $op : \Phi \rightarrow \Phi$. A modification operator modifies some aspects of the input instance to produce another instance. We assume the availability of a set of modification operators $O$. The set is specific to each instance space, meaning that for each new problem domain, new operators must be defined.

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 also keeping their output within the instance space $\Phi$.

![Figure 3.1: Modification operators for an instance](image)

Using our notations of modification operators, we can define the closure of a given core set $C$ as: $cl(C) = \{s \mid \exists c \in C, o_1, \cdots, o_n \in O[op_n(\cdots op_1(c)) = s]\}$. We would like to
define the operators such that \( cl(C) \) is as large and diverse as possible. We will later discuss the implementation of the modification operators for the textual domain.

### 3.2.2 Instance Evaluation Functions

In order to evaluate the synthesized membership queries, an evaluation function \( U : \Phi \rightarrow \mathbb{R} \) is required. One option is to use existing active-learning functions \( U_{al} : \Psi \rightarrow \mathbb{R} \). These functions are designed to assign higher values to more informative instances. We can compose these functions with the feature mapping function \( f_{\Psi} : \Phi \rightarrow \Psi \) and build the instance evaluation as: \( U(x) = U_{al}(f_{\Psi}(x)) \).

As discussed in section 2, such functions include Uncertainty Sampling [LG94], Expected Model Change [LMR04] and Query-by-Committee [SOS92], which help score unlabeled instances according to their value to the learner. Pool-based active learning approaches use these utility functions greedily in order to decide which instances to label.

### 3.2.3 Stochastic Query Synthesis

A simple way of utilizing the modification operators is to apply them in random order to the core set of instances. The algorithm maintains a set of instances \( \Omega \). We initialize \( \Omega \) with the core set and in each step we randomly choose an instance from \( \Omega \) and apply a random operator to it. The resulting new instance is added to \( \Omega \). At the end of the algorithm we return the new instances in \( \Omega \) generated by the algorithm as the MQs.

We can expect this algorithm to produce a large set of new instances but in close proximity to the basic set. Since it is exponentially difficult to generate long sequences in this fashion, only short sequences will be applied to a single instance from the core set.

**Algorithm 1:** Stochastic query synthesis using modification operators

<table>
<thead>
<tr>
<th>Input</th>
<th>seed - core set of initial instances operators - instance modification operators ( K ) - the number of membership queries we want to synthesize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>set of membership queries</td>
</tr>
<tr>
<td>1 ( \Omega = \text{seed} );</td>
<td></td>
</tr>
<tr>
<td>2 while (</td>
<td>\Omega</td>
</tr>
<tr>
<td>3 \hspace{1em} base = randomly choose an instance from ( \Omega ) ;</td>
<td></td>
</tr>
<tr>
<td>4 \hspace{1em} op = randomly choose a modification op from operators ;</td>
<td></td>
</tr>
<tr>
<td>5 \hspace{1em} new_inst = apply(base, op) ;</td>
<td></td>
</tr>
<tr>
<td>6 \hspace{1em} ( \Omega = \Omega \cup {\text{new_inst}} ) ;</td>
<td></td>
</tr>
<tr>
<td>7 return ( \Omega \setminus \text{seed} ) ;</td>
<td></td>
</tr>
</tbody>
</table>

After the set of potential MQs is generated, an instance evaluation function is applied to select which instances to send to the oracle for labeling.
3.2.4 Query Synthesis using Search Algorithms

The stochastic synthesis algorithm can be improved by treating the instance space as a search space and actively seeking the more informative instances to generate. Let us define the instance search space as follows:

- **State Space**: A state is a member of the instance space.
- **Initial States**: Any member of the instance space.
- **Actions**: The modification operators as explained in 3.2.1.
- **Heuristic Value**: The score given to the state by the instance evaluation function, as discussed in 3.2.2.

A search algorithm that applies modification operators with the goal of maximizing some heuristic value is more likely than the stochastic synthesis algorithm to generate instances with a high evaluation score. It is also difficult, using stochastic synthesis to apply many modification operators to a single instance. As search algorithms apply operators one after the other, we expect that using them will solve this problem as well.

Our search-based algorithm is listed in Algorithm 2. Similarly to the stochastic synthesis algorithm, the search-based algorithm maintains a set of instances \( \Omega \). We initialize \( \Omega \) with the core set, and in each step we randomly choose an instance from \( \Omega \) and run a search algorithm, such as beam search [RN09] or hill climbing [RN09], with the chosen instance as the initial state. The resulting new instance is added to \( \Omega \). At the end of the algorithm we return the new instances in \( \Omega \) generated by the algorithm as the MQs.

The search algorithm uses the search space described earlier in order to generate the highest scoring instance instead of randomly walking in that space. This process is then repeated until a sufficient number of instances has been generated. To avoid generating the same instance multiple times, we prefer stochastic processes.

<table>
<thead>
<tr>
<th>Algorithm 2: Search-based query synthesis algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong>: seed - core set of initial instances</td>
</tr>
<tr>
<td>operators - instance modification operators</td>
</tr>
<tr>
<td>H - active-learning instance evaluation function</td>
</tr>
<tr>
<td>K - the number of membership queries we want to synthesize</td>
</tr>
<tr>
<td>searchAlg - local search algorithm</td>
</tr>
<tr>
<td><strong>Output</strong>: set of membership queries</td>
</tr>
<tr>
<td>1 ( \Omega = \text{seed} );</td>
</tr>
<tr>
<td>2 while (</td>
</tr>
<tr>
<td>3 \hspace{1cm} \text{base = randomly choose an instance from } \Omega ;</td>
</tr>
<tr>
<td>4 \hspace{1cm} \text{new}_\text{inst} = \text{apply(searchAlg, base, operators, H)} ;</td>
</tr>
<tr>
<td>5 \hspace{1cm} \Omega = \Omega \cup {\text{new}_\text{inst}} ;</td>
</tr>
<tr>
<td>6 return ( \Omega \setminus \text{seed} );</td>
</tr>
</tbody>
</table>
As we can see in Algorithm 2, the search-based algorithm can work with any local search algorithm, and with any active-learning evaluation function.

### 3.3 Textual Instance Space

In this work, we focus on using our approach in the textual domain. At this stage we limit our study to 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 representations, such as BOW, 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 1-1 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 for the text classification domain to be the set of all syntactically and semantically legal sentences in English.

Before we formally define the modification operators for the textual domain, let us look at the following example, from the task of hate speech detection, where our task is to classify each sentence as containing or not containing hate speech. We begin with the sentence “I hate all dogs”. Our modification operators will attempt to substitute words in this sentence. Our operators can switch the word “hate” with “adore”, resulting in a non-hate speech sentence with an opposite label to its original. However, substituting the word “dogs” with “cats” still results in a positive instance. A complete example of our modification operators in this case can be seen in Figure 3.2.

### 3.3.1 Modification Operators in the Textual Domain

To define our modification operators, we must first define a *semantic neighborhood* of a word and a *distance function* between words. 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.

For now, we assume the existence of a function \(d(w, w')\) that measures the semantic distance between two words. Let us 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 a replaceable word with a member of its k-semantic neighborhood. The candidate operators are then filtered to keep only those that retain the syntactic structure of the original sentence.

### 3.3.2 Computing the Semantic Distance

In order to calculate \( d(w, w') \), we can use existing methods for computing semantic distance. We have considered 4 existing methods: WordNet [Jar12], Word2vec [MSC+13], Glove [PSM14] and Dependency Word2vec [LG14].

Given a word, our goal is to find a diverse set of words that are semantically related to it. For a word to serve as a possible replacement within a specific context, it has to adhere to two general rules: First, it has to be functionally similar [TP10] to the original, meaning that the two words behave similarly in their context. Second, it has to be semantically related to the original word. For example, “book” and “dog” can be functionally similar to “cat”, but in the sentence “I want to pet this cat”, it is acceptable replace with “dog” but not with “book”.

Let us compare 4 different methods to measuring semantic distance with respect to these two rules. Throughout this comparison, we take as our example the sentence:

> “Batman is really awesome”.

'Batman' and 'awesome' are the replaceable words in this sentence, and their semantic neighborhoods will be compared.
WordNet [Mil95] is a well-known graph-based knowledge base of English words, organized by lexical similarity including different meanings for disambiguation. Its most important feature is the IS-A graph provided for many of the words in the database. Similarly to Jarmasz [Jar12], we defined the distance between words by the distance between their representative nodes in the WordNet graph.

<table>
<thead>
<tr>
<th>Batman</th>
<th>awesome</th>
</tr>
</thead>
<tbody>
<tr>
<td>courtier</td>
<td>amazing</td>
</tr>
<tr>
<td>orderly</td>
<td>awe-inspiring</td>
</tr>
<tr>
<td>squire</td>
<td>awful</td>
</tr>
<tr>
<td>bellboy</td>
<td>awing</td>
</tr>
<tr>
<td>bellman</td>
<td>impressive</td>
</tr>
</tbody>
</table>

Table 3.1: Semantic neighborhoods as defined by WordNet

Table 3.1 lists semantic neighborhoods of the replaceable words, using WordNet. We can see that the semantic neighborhood for the word “Batman” is off the mark, as WordNet completely missed the meaning of “Batman” conveyed in this sentence and went with a more obscure meaning of “an orderly assigned to serve a British military officer”. This is a well-known drawback of WordNet: new or rarely used meanings of words may not appear there at all (as we saw with “Batman”), while in other unsupervised methods, such as Word2vec, they appear naturally. The semantic neighborhood for “awesome”, however, proved to be quite fitting, including even words with opposite meaning like “awful”.

Word2vec [MSC+13] and Glove [PSM14] map each word into a vector space learned from a large corpus, such that related words will be mapped into close vectors. The distance between two words is computed by the Euclidean distance between their representation in the vector space.

<table>
<thead>
<tr>
<th>Batman</th>
<th>awesome</th>
<th>Batman</th>
<th>awesome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caped_Crusader</td>
<td>amazing</td>
<td>superman</td>
<td>incredible</td>
</tr>
<tr>
<td>Superman</td>
<td>unbelievable</td>
<td>superhero</td>
<td>amazing</td>
</tr>
<tr>
<td>Nightwing</td>
<td>fantastic</td>
<td>sequel</td>
<td>unbelievable</td>
</tr>
<tr>
<td>Batman_Begins</td>
<td>incredible</td>
<td>catwoman</td>
<td>fantastic</td>
</tr>
<tr>
<td>DC_Comics</td>
<td>unbelievable</td>
<td>joker</td>
<td>marvelous</td>
</tr>
<tr>
<td>Superman_Batman</td>
<td>terrific</td>
<td>comics</td>
<td>awful</td>
</tr>
</tbody>
</table>

(a) Word2vec

<table>
<thead>
<tr>
<th>Batman</th>
<th>awesome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batman</td>
<td>awesome</td>
</tr>
<tr>
<td>Caped_Crusader</td>
<td>amazing</td>
</tr>
<tr>
<td>Superman</td>
<td>unbelievable</td>
</tr>
<tr>
<td>Nightwing</td>
<td>fantastic</td>
</tr>
<tr>
<td>Batman_Begins</td>
<td>incredible</td>
</tr>
<tr>
<td>DC_Comics</td>
<td>unbelievable</td>
</tr>
<tr>
<td>Superman_Batman</td>
<td>terrific</td>
</tr>
<tr>
<td></td>
<td>Comics</td>
</tr>
</tbody>
</table>

(b) Glove

Table 3.2: Semantic neighborhoods as defined by Word2vec and Glove

Table 3.2 lists the semantic neighborhoods of the replaceable words, computed using Word2vec & Glove. We can see that the semantic neighborhoods for the word “Batman” are much better than the one produced by WordNet. It includes words such as
“Superman” and “Catwoman” that are clearly good candidates. Nonetheless, despite their advantages over WordNet, both Word2vec and Glove share the tendency to measure semantic similarity but not functional similarity [LG14] which is problematic for our intended usage. This problem can be seen in the neighborhoods of “Batman” that include words like “sequel” and “DC_Comics”. These words cannot be switched with “Batman” in the given sentence although they are related. The neighborhood for “awesome” in this case is as good as the one given by WordNet, with some advantage of using more common words.

Dependency Word2vec [LG14] is based on Word2vec but relies on the sentence’s dependency graph to measure distance when building the word vectors. This change results in word vectors that tend to measure functional similarity as well as domain similarity [LG14]. This can be seen in these neighborhoods:

<table>
<thead>
<tr>
<th>Batman</th>
<th>awesome</th>
</tr>
</thead>
<tbody>
<tr>
<td>superman</td>
<td>terrific</td>
</tr>
<tr>
<td>superboy</td>
<td>marvelous</td>
</tr>
<tr>
<td>supergirl</td>
<td>wonderful</td>
</tr>
<tr>
<td>catwoman</td>
<td>lousy</td>
</tr>
<tr>
<td>aquaman</td>
<td>awful</td>
</tr>
<tr>
<td>batgirl</td>
<td>crapp</td>
</tr>
</tbody>
</table>

Table 3.3: Semantic neighborhoods as defined by Dependency Word2vec

Table 3.3 lists the semantic neighborhoods of the replaceable words, using Dependency Word2vec. Dependency Word2vec shows a clear preference for functional similarity with the neighborhood for “Batman”; it contains only other superhero names which all behave similarly in this context. The neighborhood for “awesome” is also good. Although it is similar to that of the other methods, the neighborhood of “awesome” is more diverse, with more abundant negative words such as ‘lousy’ and ‘awful’ than in Word2vec and Glove. This diversity is valuable for our modification operators as they do not force the meaning of the sentence to be preserved, in contrast to augmentation methods.

With this analysis in mind, we chose to use Dependency Word2vec with a semantic neighborhood size of 10 for the empirical evaluations to follow.
Chapter 4

Empirical Evaluation

We analyzed the performance of our framework on 5 publicly available sentence classification datasets. The code for all experiments is available at https://github.com/jonzarecki/textual-membership-queries.

4.1 Experimental Methodology

In this subsection we will discuss the methodology of our experiments: the datasets used, the methods compared and our experimental design.

4.1.1 Datasets

We report results on 5 binary sentence classification datasets: three sentiment analysis datasets, one sentence subjectivity dataset, and one hate-speech detection dataset. A description of each dataset is given below.

**CMR**: Cornell sentiment polarity dataset [PL05].

**SUBJ**: Cornell sentence subjective / objective dataset [PL04].

**SST**: Stanford sentiment treebank, a sentence sentiment analysis dataset [SPW13].

**HS**: Hate speech and offensive language classification dataset [DWMW17].

**KS**: A Kaggle short sentence sentiment analysis dataset.

In addition, Table 4.1 provides key statistics for each dataset.

---

1The CMR and SUBJ datasets are available at: http://www.cs.cornell.edu/people/pabo/movie-review-data.

2The SST dataset is available at: https://nlp.stanford.edu/sentiment/index.html.

3The HS dataset is available at: https://github.com/t-davidson/hate-speech-and-offensive-language.

4The KS dataset is available at: https://www.kaggle.com/c/si650winter11.
### 4.1.2 Simulating the Human Oracle

As in most works on active learning, we require a human expert to query our unlabeled instances. 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 borrow an idea from a work on feature labeling [DSM09] and simulate a human labeler. To make the artificial setting as close as possible to a real-world setting, the artificial expert is a learning model trained on the entire dataset, and for each dataset we chose a model which is close to the state-of-the-art for the task.  

<table>
<thead>
<tr>
<th></th>
<th>CMR</th>
<th>SUBJ</th>
<th>SST</th>
<th>HS</th>
<th>KS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>5331</td>
<td>5000</td>
<td>5906</td>
<td>20620</td>
<td>773</td>
</tr>
<tr>
<td>Negative</td>
<td>5331</td>
<td>5000</td>
<td>5938</td>
<td>4162</td>
<td>639</td>
</tr>
<tr>
<td>Avg. word count</td>
<td>21.0</td>
<td>24.0</td>
<td>19.2</td>
<td>13.9</td>
<td>11.5</td>
</tr>
</tbody>
</table>

Table 4.1: Statistics for tested datasets

<table>
<thead>
<tr>
<th></th>
<th>CMR</th>
<th>SUBJ</th>
<th>SST</th>
<th>HS</th>
<th>KS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CV accuracy</td>
<td>86.8</td>
<td>94.4</td>
<td>86.2</td>
<td>91.0</td>
<td>94.5</td>
</tr>
</tbody>
</table>

Table 4.2: Cross-validation accuracy of the simulated oracle on each dataset

In Table 4.2 we can see the cross-validation accuracy of each artificial expert on its dataset. The expert receives close to state-of-the-art performance for each dataset and as such it is the closest possible simulation to a human expert. This method does not pose as training set contamination. The oracle is only used for labeling new instances, simulating the world knowledge of a human labeler used in other active-learning works. So, while we do use the test data to train the artificial expert, this information does not return to the learner through its training set but in a proper manner.

### 4.1.3 Experiment Design

Each of the experiments shown below follows the setup described here:

A diagram describing the process can be seen in Figure 4.1.

1. For each dataset $D$, we randomly sample 40% of the data as the test set $T$.

2. From the remaining 60%, we sample 10 instances to serve as the core set $C$ used to train the initial model. We use $C$ to generate a pool of membership queries.

3. The rest of the data is then used as the unlabeled pool $U$. Pool-based AL

---

For SST, CMR, SUBJ & KS we used the open-source implementation of Generating Reviews and Discovering Sentiment [RJS17], which achieved state-of-the-art results for CMR and SUBJ. For KS it achieved 94% accuracy. Available at: https://github.com/openai/generating-reviews-discovering-sentiment.

For the hate-speech (HS) dataset we used a BOW-based classifier, which achieved 91% accuracy.
competitors use U to create a pool of unlabeled instances (this will be discussed later).

4. The artificial oracle discussed in 4.1.2 is trained on the complete dataset D. This provides the oracle with the most information possible, making it as close as possible to real-world setting.

5. The "learner" used in our experiments is configured as following: We used standard pre-trained 300-dimension Glove\(^7\) word vectors as a basis for our feature space for this task, which was an average of the word vectors over all words in the sentence. The learning model for the learner is a standard scikit-learn logistic regression model.

We repeat the experiments 20 times with different random seeds while keeping the same T, C and U. This results in different generated instances across all methods. The results shown are the average results of these 20 runs.

4.1.4 Compared Methods

In the following experiments, we compared our methods with other approaches, as follows.

Our methods:

- **Uncertainty sampling hill-climbing MQ synthesis (US-HC-MQ):** The proposed search-based synthesis method, using hill-climbing as the search algorithm and uncertainty sampling as the heuristic function. We used an average depth of 4 for the hill-climbing algorithm. The reason for this choice will be discussed in 4.3.

- **Uncertainty sampling beam-search MQ synthesis (US-BS-MQ):** The proposed search-based synthesis method, using beam-search as the search algorithm and uncertainty sampling as the heuristic function. We used an average depth of 4 for the beam-search algorithm.

---

\(^7\)https://nlp.stanford.edu/projects/glove/
• **Stochastic Synthesis (S-MQ):** A degenerated version of our method, described in detail in 3.2.3.

As no other work attempted to build membership queries in the textual domain prior to this work, a direct comparison with similar works was not possible. We thus chose 3 somewhat similar approaches to generation/augmentation of textual instances to compare with.

**Competitor Methods:**

• **Original examples (IDEAL):** 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 a ‘upper-limit’ role, enabling us to see what would happen if we had unlabeled examples.

• **RNN Generator (RNN):** A method for generating sentences with an LSTM (RNN) model. We trained the models with only the core set of instances.

• **WordNet-based data augmentation (WNA):** A possible approach to text augmentation, where words are replaced with their respective synonyms from WordNet [ZL15].

We compare both US-HC-MQ and US-BS-MQ to examine the effect of the search algorithm on the resulting MQs, and compare with S-MQ to observe the gain of using utility functions when searching for valuable unlabeled instances. IDEAL is used to estimate the performance when using sampled original examples, while RNN and WNA are used for comparison with somewhat similar text generation/augmentation approaches.

### 4.2 Experiment 1 - Batch Active Learning with Membership Queries

We measured the performance of our MQ synthesis framework in a batch AL 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. The pseudo-code of the batch AL used in this experiment is described in Algorithm 3.

In each step, we use the pool generation function ‘gen_MQs’ to return a pool of unlabeled instances with size P. Then heuristic function H is used to extract the m most informative instances 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.
Algorithm 3: Algorithm for calculating results in batch AL experiment

**Input**: seed - core set of initial examples  
| gen_MQs - a function generating membership queries  
| H - active-learning instance evaluation function  
| K - the number of examples we want to add to the core set  
| P - the size of pool generated by gen_MQs  
| m - the batch size for the batch AL setup  

**Output**: list of test-set accuracies for each step in the batch AL setup

1. set current training set as the seed  
2. train model on the current training set  
3. calculate model accuracy on test set  
4. **while** did not generate K new examples **do**  
   5. generate a pool of P instances with gen_MQs  
   6. choose the m highest scoring instances as calculated by H  
   7. label chosen instances with oracle  
   8. add new labeled instances to training set  
   9. train model on the current training set  
   10. calculate model accuracy on test set

RNN results are not shown because the sentences generated using this approach were not comprehensible. Training a neural-network model with only 10 sentences is impractical, as usually these models are fed with large amounts of data\(^8\). Some examples of the sentences are: “r and lapaglia .”, ”spite the hated infidel ways of her father , she han kid , with monty pyris or .at kin .”, ”x this film .”

The following figures plot 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 heuristic function as H for all experiments.

The final results after the addition of 100 examples are shown in the chart below. The score for method A is calculated as the following:

$$A's\ ratio\ score = \frac{A's\ final\ accuracy}{IDEAL's\ final\ accuracy}$$

This scoring method serves as a regularization method between the datasets, as more difficult datasets will have lower accuracies across all methods. This results in similar ratios across the different datasets. Error bars represent one standard deviation around the mean.

First let us look at the CMR dataset. From Figure 4.2 we can see that all generation methods increase the accuracy. However, hill-climbing and beam-search improve the accuracy the most. The best improvement is 9%. In SUBJ the results follow a similar trend to the results for the CMR dataset. Hill-climbing and beam-search come out on

\(^8\)https://blog.openai.com/generative-models/
top, improving the accuracy by 5%. In SST we also see that the search-based methods are superior in this dataset, improving the accuracy by 8%. WNA, however, struggled in this dataset, reducing the initial model’s accuracy. HS shows the most impressive results, coming very close to IDEAL without using any existing unlabeled instances. Finally, the KS results are similar to previous datasets, improving the model by 8%.

We can see that our methods consistently increased the accuracy of the initial model. The search-based synthesis approaches also demonstrated impressive results across all datasets. Notably, we also showed that our method can be competitive with methods that use original examples and have access to a pool of unlabeled instances not available to our methods.

Our comparison of the two search-based approaches (US-HC-MQ and US-BS-MQ) showed no clear preference. Both exhibited excellent performance across the 5 datasets. Our 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. The limited amount of synonyms available from WordNet makes it unable to generate a large pool of instances. As we saw in all of the datasets, it was unable to generate all 100 sentences. In comparison, our method can theoretically generate as many instances as required.

As mentioned, the sentences generated by RNN were of very poor quality, and results for this method were not included in the plots. We believe the poor results are due to the large amounts of data required for this method. It could not learn well from a seed of only 10 sentences and generated ungrammatical sentences as a result.

Finally, the Friedman test diagram [Dem06] with Bonferroni-Dunn post-hoc testing and a p-value \( \leq 0.05 \) is shown in Figure 4.4. The statistical advantage of the search-based method over WNA is clear even when a relatively small number of datasets (5) was used for the test.

![Figure 4.4: Friedman test diagram [Dem06] for the results with p-value \( \leq 0.05 \), showing a statistical significance for search-based methods over the baseline method WNA](image)
4.3 Experiment 2 - The Effect of the Number of Operators Applied

Our MQ framework is based on applying modification operators. In this experiment, we examined the effect of the number of modification operators on the accuracy of the resulting model trained with these MQs.

To accurately evaluate the effect of the modification operators, we randomly applied \( N \) operators to random instances, collecting a pool of 50 unlabeled instances generated using this approach. These were then labeled by our expert and incorporated into our basic core set. The accuracy of the method was then measured on the test data. We repeated this experiment 20 times and show the averaged results.

Figure 4.5 shows the accuracy achieved by this method on the Kaggle short sentence sentiment dataset (KS) and the Cornell sentiment polarity dataset (CMR). As we can see, in the KS dataset, smaller numbers of modification operators are preferable. However, in the CMR, better results are achieved when more operators are used.

![Figure 4.5: The number of operators vs. the accuracy of the final model](image)

We believe these results can be explained by the average length of the sentences in each dataset. CMR has the longest sentences on average, where KS has the shortest. For short sentences the large number of operators means that the same word will be replaced multiple times, possibly resulting in words not related to the original, as we will discuss later in the qualitative analysis. For longer sentences, large number of operators are needed to replace each candidate word even once. These results led to our decision to use an average of 4 operators for the search algorithms in all the datasets, a good compromise for longer and shorter sentences.
4.4 Experiment 3 - The Effect of the Synthesis Algorithm on Label Switches

In our framework, instances are also able to change their original label. In this experiment we tested the effect of our synthesis algorithms on the number of label changes they generate.

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 changed their original label.

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 repeat the experiment 20 times with different core sets and show the average results on all available datasets.

![Figure 4.6: The effect of the synthesis algorithm on the number of changed labels](image)

Figure 4.6 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.

The instances that changed labels are “near-misses” [GMR06]. While they originally belonged to a certain class, our sequence of modification operators caused them to switch their original class without significant changes to the instance.

4.5 Qualitative Analysis

Let us look more carefully at the instances generated by our membership queries and the sequence of modification operators that were applied. First let us look at short sequences from the datasets:
• $S_0$: That’s why I love Brokeback Mountain, apart from Ang Lee.
  ↓ $op_1$: love → despise
$S_1$: That’s why I despise Brokeback Mountain, apart from Ang Lee.

• $S_0$: I thought The Da Vinci Code movie was really boring.
  ↓ $op_1$: boring → fascinating
$S_1$: I thought The Da Vinci Code movie was really fascinating.

• $S_0$: The Da Vinci Code was an awesome book, i just finished it.
  ↓ $op_1$: awesome → lousy
$S_1$: The Da Vinci Code was a lousy book, i just finished it.

As we can see, for the sentiment analysis task, short sequences are sufficient for the instances to change their label. This is the result of the sentiment task’s dependency on “positive” or “negative” words for the classification of a sentence. An example for this dependency can be seen in VADER [HG14], a rule-based sentiment classification model which used only a lexicon of “positive” and “negative” words in order to classify sentences without applying any machine learning techniques.

Now let us move to more complicated sequences.

• $S_0$: Never tell a bitch what u up to.
  ↓ $op_1$: bitch → slob
$S_1$: Never tell a slob what u up to.
  ↓ $op_2$: slob → bookworm
$S_2$: Never tell a bookworm what u up to.

  What’s really so appealing about the characters is their resemblance to everyday children.
  ↓ $op_1$: appealing → mystifying
$S_1$: What’s really so mystifying about the characters is their resemblance to everyday children.
  ↓ $op_2$: everyday → day-to-day
$S_2$: What’s really so mystifying about the characters is their resemblance to day-to-day children.
  ↓ $op_3$: mystifying → saddening
$S_3$: What’s really so saddening about the characters is their resemblance to day-to-day children.
  ↓ $op_4$: saddening → sickening
$S_4$: What’s really so sickening about the characters is their resemblance to day-to-day children.
These sequences demonstrate the ability of our MQ framework to switch a word multiple times, as a result of which the sentence departs further from its original meaning. In the first sequence we move from the offensive word “bitch” to the less offensive “slob” and then to the inoffensive “bookworm”, completing a transition from an offensive sentence to an inoffensive one. In the second sequence we can see a similar transition from “appealing” to “sickening”, which changes the sentiment expressed in the sentence. We can also see that not all the switched words change the meaning of the sentence: the switch from “everyday” to “day-to-day”, for example, preserves the meaning but does change the sentence’s vector representation. This will have an effect on the learning algorithms. As to the algorithm, these two examples will appear different.

Finally, let us look at two examples where our framework did not generate legal sentences.

- $S_0$: **Serviceable** at best, slightly less than serviceable at worst.
  \[ \downarrow \text{op}_2: \text{serviceable} \rightarrow \text{unseaworthy} \]
  $S_1$: **Unseaworthy** at best, slightly less than serviceable at worst.

- $S_0$: Da Vinci Code sucked, as **expected**.
  \[ \downarrow \text{op}_2: \text{expected} \rightarrow \text{forecasted} \]
  $S_1$: Da Vinci Code sucked, as **forecasted**.
  \[ \downarrow \text{op}_2: \text{forecasted} \rightarrow \text{preprogrammed} \]
  $S_2$: Da Vinci Code sucked, as **preprogrammed**.

As we can see, even when using a knowledge-base it is possible that words will be replaced with seemingly unrelated words. In the first example “serviceable” is replaced with “unseaworthy”. These words are only remotely related. This is a case where our knowledge-base did not perform well. Using more data to train it might have solved the problem.

In the second example we see that switching a word multiple times can have a negative effect on 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. Here, too, a better method for calculating semantic distance between words might have solved the problem. Another possible solution to this problem is to use multi-sense embeddings [IPN15] in conjunction with Dependency Word2vec to create different embeddings for each sense of the requested word while preserving the important functional similarity property.
Chapter 5

Related Work

In Section 2 we discussed works related to the topic of membership queries in the textual domain. In this section we will further discuss three works that are of particular relevance.

As discussed in section 2, local membership queries were first introduced by Awasthi et al. [AFK13]. Bary [Bar15] built on this idea and attempted using a learning model with only 1-local membership queries, meaning that only one feature in the original example can be changed. Bary proved impressive theoretical results about the power of such a 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 subsection 3.1, it is usually impossible 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.

In our previous work [GMR06] we 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 returns 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 algorithm and heuristic functions during the MQ generation process, but was mainly focused on generating near-miss examples.

Several works have discussed the topic of textual data augmentation [ZL15, Ros17], 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 [ZL15] to be less effective
than our suggested methods.
Chapter 6

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 uses 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 be competitive with an approach that utilizes a pool of unlabeled instance not available to our MQ framework. To further this line of research, we have released the implementation of our textual MQs as open source software at https://github.com/jonzarecki/textual-membership-queries.

Our results motivate three interesting directions for future work that we plan to explore. First, our results indicate that there is much more information to be found in examples already present in the training set. Thus it is possible for our modification operators to be used within an augmentation setup, where we assume the operators do not change the label of the resulting instance and then ask which instances are most beneficial to the learner, a question rarely asked when using data augmentation. In addition, augmentation operators can be used as a means of oversampling in highly skewed datasets, as a alternative to SMOTE [CBHK02] for example, which does the oversampling in the feature space.

Second, our results also indicate that using utility functions (such as uncertainty sampling) does result in more valuable instances. Thus we plan to explore ways to enrich existing pools of unlabeled instances using modification operators using a method similar to the one used here. This direction may enhance the results of existing methods for pool-based AL.

Third, we are interested in exploring the option of implementing our approach in different domains such as the image domain, or expanding our implementation in the textual domain to documents in addition to sentences. These options will require us to
define new modification operators to better fit these types of instances.
Bibliography


The document is in Hebrew. Due to the content, it seems to be discussing methods in natural language processing, specifically in the field of text encoding and image recognition. It mentions the use of operators to replace words in a sentence, maintaining its meaning while allowing it to change after the operation. Examples of such operators are CEV, DROW, and TEN.

The research involves a wide range of experiments to compare the effectiveness of different methods and parameters in algorithms. The results obtained from these experiments show that certain methods and operators can improve the performance of the system when used with the chosen method. Moreover, the effectiveness of these methods is visible to the naked eye.

The document also mentions the inclusion of techniques to improve the system's performance and the importance of visual quality in the models used.

The Technion - Computer Science Department - M.Sc. Thesis MSC-2018-31 - 2018
In recent years, we have seen a significant increase in the volume and source of text data. For a long time, spam or positive or negative classification was used to remove text data from our inboxes. Applications such as these have become very popular. Machine learning algorithms, which are based on known data, are used to perform their work well. This method has proven itself for many tasks, but in many cases it is difficult to obtain large known data sets. For example, in such cases, you do not have much data to work with. Classification of text data can be a great help in these cases, but it is critical for the success of automatic machines that rely on this data. As a response to the question of how to reduce the amount of known data, active learning is used. In some cases, the method is called "critical," as it is based on theoretical examples and cannot identify the boundaries of the world in a particular example. However, it lacks meaning, as active learning relies on labelling examples. We can use active learning for example, in the case of terrorists, where there is not much data to work with. The method of classification of text data can help this process, but it is critical for the success of automatic machines that rely on this data. As a response to the problem, we suggest a new method that is capable of creating examples while maintaining the characteristics of the existing examples. The method is based on the definition of "crucial" and its meaning is not its existence. We can use active learning for example, in the case of terrorists, where there is not much data to work with. The method of classification of text data can help this process, but it is critical for the success of automatic machines that rely on this data.
המחקר בוצע ברוחביו של פרופסור שאול מרקוביץ, בפקולטה למדעי מחשב.

נתאאתי מחקר זה עד לא פרשינו בכתיב בק טלטול אינדי מתכב בתכנית זה.

תודה

אני רצה להודות למנהלה של פרו' שאול מרקוביץ, אשר לקח אחוי בשבל מחקרנו משאר המочки והראשה.

ולא יכול להסיע מטעם ביעלון המחקר.

בנוסף אני רצה להודות לדר' אניר צהיר, אשר תמכו בי במ.arange בחול.scope החשוב שהוביל הטיות וה-runner קביש.

בר.

הכרת תודות מסורתי לleston על פימיק המחק.

Technion - Computer Science Department - M.Sc. Thesis  MSC-2018-31 - 2018
שאילתות שייכות morbוססות טקסט

יווחר על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
מוניטור crédit מסוים במדעי המחשב

יונתן ברקאי

רוגשلطנט секנין... מון ספטנונא ליסראל
תפומז הלשון"ע" חפף יוני 2018
שאילתה של שייכות מבוססות טקסט

יונתן ברק

ינון זרקי