Master of Science Thesis

Enhanced Question Classification with Optimal Combination of Features

Babak Loni

Supervisors: Dr. M. Loog, Dr. D.M.J. Tax

Thesis Committee:

Prof. dr. ir. M.J.T. Reinders
Dr. D.M.J. Tax
Dr. ir. Pascal Wiggers
Yan Li

Sep 2010 – Aug 2011

Pattern Recognition Lab.
Department of Media and Knowledge Engineering
Faculty of Electronic Engineering, Mathematics and Computer Science
Delft University of Technology

TU Delft **Delft University of Technology**

Challenge the Future

Enhanced Question Classification with Optimal Combination of Features

Babak Loni

Department of Media and Knowledge Engineering
Delft University of Technology

Master Thesis

Enhanced Question Classification with Optimal Combination of Features

Babak Loni

Department of Media and Knowledge Engineering
Delft University of Technology

Abstract

An important component of question answering systems is question classification. The task of question classification is to predict the entity type of the answer of a natural language question.
Question classification is typically done using machine learning techniques. Different lexical, syntactical and semantic features can be extracted from a question. In this work we introduce two new semantic features which improve the accuracy of classification. Furthermore, we developed a weighed approach to optimally combine different features.
We also applied Latent Semantic Analysis (LSA) technique to reduce the large feature space of questions to a much smaller and efficient feature space. We adopted two different classifiers: Back-Propagation Neural Networks (BPNN) and Support Vector Machines (SVM). We found that applying LSA on question classification can not only make the question classification more time efficient, but it also improves the classification accuracy by removing the redundant features. Furthermore, we discovered that when the original feature space is compact and efficient, its reduced space performs better than a large feature space with a rich set of features. In addition, we found that in the reduced feature space, BPNN performs better than SVMs which are widely used in question classification. We tested our proposed approaches on the well-known UIUC dataset and succeeded to achieve a new record on the accuracy of classification on this dataset.

Categories and Subject Descriptors:
Question Answering Systems
Natural Language Processing
Machine Learning

Key words:
Question Classification, Question Answering Systems, Lexical Features, Syntactical Features, Semantic Features, Combination of Features, Feature Reduction, Latent Semantic Indexing, Support Vector Machines, Back-propagation Neural Networks

Contents

Chapter 1

Introduction

By the rapidly increasing amount of knowledge in the Web, search engines need to be more intelligent than before. In many cases the user only needs a specific piece of information instead of a list of documents. Rather than making the user to read the entire document, it is often preferred to give the user a concise and short answer. Question Answering (QA) systems are aimed to provide the exact piece of information in response to a question. An *open domain* question answering system should be able to answer a question written in natural language, similar to humans.

The study to build a system which answers natural language questions backs to early 1960s. The first question answering system, BASEBALL, (Green et al., 1961) was able to answer *domain-specific* natural language questions which was about the baseball games played in American league over one season. This system was simply a database-centered system which used to translate a natural language question to a canonical query on database.

Most of other early studies (Simmons, 1965; Woods, 1973; Lehnert, 1977) was mainly domain-specific systems or have many limitation on answering questions. Due to lack of enough back-end knowledge to provide answer to open domain questions, the research on question answering systems lay dormant for few decades until the emergence of the web. The huge amount of data on the web on one hand and the need for querying the web on the other hand, brought again the task of question answering into focus. The focus on question answering research increased specially when the Text REtrieval Conference (TREC) began a QA track in 1999 (Voorhees and Harman, 2000).

The simplest type of question answering systems are dealing with *factoid questions* (Jurafsky and Martin, 2008). The answers of this type of questions are simply one or more words which gives the precise answer of the question. For example questions like "What is a female rabbit called?" or "Who discovered electricity?" are factoid questions. Sometimes the question asks for a body of information instead of a fact. For example questions like "What is gymnophobia ?" or "Why did the world enter a global depression in 1929?" are of these type. To answer these questions typically a *summary* of one or more documents should be given to the user.

Many techniques from information retrieval, natural language processing and machine

learning have been employed for question answering systems. Some early studies were mainly based on querying structured data while the others used to apply pattern matching techniques. Androutsopoulos et al. (1995) provides an overview of the early question answering systems. Recent studies on open-domain QA systems are typically based on Information Retrieval (IR) techniques. The IR-based question answering systems try to find the answer of a given question by processing a corpus of documents, usually from the web, and finding a segment of text which is likely to be the answer of that question.

Some other recent works are founded on some pre-defined ontologies. These systems are based on *semi-structured* knowledge-bases and can not directly process free form documents on the web. They often demand the web documents to be represented in structured or semi-structured formats. *Semantic web* (Berners-Lee et al., 2001) was the most successful attempt to represent the web documents in a structured way; although it never achieved its desired state (Anderson, 2010). Systems such as START (Katz et al., 2002), and True Knowledge[1] are two question answering engines working on top of *semi-structured* data and semantic-web-based technologies. These systems have their own knowledge bases which are mainly created by semi-automated data annotation.

What is referred as a true *automated* question answering system, is an IR-based system which can understand natural language question, process free form text and extract the true answer from text documents. A QA system which finds the answers directly from documents is called *shallow* system. If the system is capable to do inference on the facts, it is referred as *deep* QA system. Majority of current research on question answering try to come up with ideas to build such an intelligent systems, either shallow systems or deep systems.

Typically an automated QA system has tree stages (Jurafsky and Martin, 2008): question processing, passage retrieval and answer processing. Figure 1.1 illustrates the common architecture of a factoid QA system. Below the task of each component is briefly described:

- **Question Processing:** the task of question processing is to analyze the question and create a proper IR query as well as detecting the *entity type* of the answer, a category name which specifies the type of answer. The first task is called *query reformation* and the second is called *question classification*.

- **Passage Retrieval:** the task of passage retrieval is to query over the IR engine, process the returned documents and return candidate passages that are likely to contain the answer. Question classification comes handy here: it can determine the search strategy to retrieve candidate passages. Depending on the question class, the search query can be transformed into a form which is most suited for finding the answer.

- **Answer Processing:** the final task of a QA system is to process the candidate passages and extract a segment of word(s) that is likely to be the answer of the

[1] www.trueknowledge.com

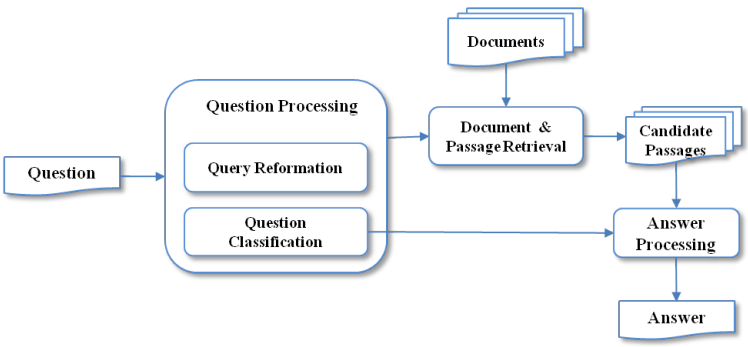

Figure 1.1: The common architecture of a factoid question answering system

question. Question classification again comes handy here. The candidate answers are ranked according to their likelihood of being in the same class as question class and the top ranked answer(s) will be considered as the final answer(s) of the question.

In this work we have focused on question classification, an important component of question answering systems. The task of question classification is to predict the entity type or category of the answer. This can be done by different approaches. Most of the early studies use hand-crafted rules to classify question. However, successful approaches are based on statistical learning methods. In these approaches different type of features from the lexical, syntactical and semantic structure of questions are extracted. In the supervised methods a classifier is trained on a training set and the accuracy of the classifier is tested with an independent test set.

1.1 Contributions of This Work

There are many challenges on question classification problem and there are many issues needed to be answered in this area. The main motivation of this work is to improve the performance of learning-based question classifier systems to contribute on the next generation of question answering systems. Following are the main challenges (research questions) in QC problem that we addressed in this work:

1. What kind of features can be extracted from a question written in natural language and what is the contribution of each feature in question classification?

2. How these features can be extracted?

3. Can combination of features always improve the accuracy of question classification? How can we efficiently combine features?

4. Can features reduction techniques be applied in the area of question classification? Can these techniques improve the classification accuracy?

5. What classifier(s) is suitable for question classification?
6. Why some questions are usually misclassified by a machine? Which kind of questions are more likely to be misclassified and what is the reason of misclassification?

One important question that we address in this thesis is to investigate the contribution of third party information sources such as WordNet in question classification. Third party information sources can help to understand the semantic of a natural language sentence. The following questions are also going to be addressed in this thesis:

7. Does a question alone have enough information to be classified and understand correctly by a machine?
8. Can adding information from WordNet help machine to better understand a natural language question? i.e. can it help to better classifying a question?
9. How can we effectively exploit information from WordNet?
10. Is it possible to expand feature vector with WordNet Hypernyms? If yes how? Can it help?

To answer these questions, we implemented a question classifier system. Many techniques from pattern recognition, machine learning, natural language processing and information retrieval have been used to implement this system. In this work we succeeded to introduce the following contributions on the QC problem:

1. More efficient set of features for question classification
2. A better way to combine features in question classification
3. A better way (or at least an alternating way) to exploit semantic information from WordNet
4. Using the Back-propagation Neural Network for question classification for the first time
5. Reducing the feature space to more efficient and effective space by successfully applying the latent semantic indexing method
6. Better understanding of misclassification causes in question classification

And:

7. A more accurate question classifier than previous works

This report is organized as follows: in chapter 2 we give an introduction on question classification problem and introduce necessary concepts in this area. Chapter 3 explains our question classification approach in details. We explain our experiment and implementation details as well as discussion on the results on chapter 4. In chapter 5 we introduce related work on question classification and compare our method with them. The conclusions and future directions are discussed in chapter 6.

Chapter 2

Question Classification

2.1 Introduction

The task of a question classifier is to assign one or more class labels, depending on classification strategy, to a given question written in natural language. For example for the question "What London street is the home of British journalism?", the task of question classification is to assign label "Location" to this question, since the answer to this question is a named entity of type "Location". Since we predict the *type* of the answer, question classification is also referred as *answer type prediction*. The set of predefined categories which are considered as question classes usually called *question taxonomy* or *answer type taxonomy*. In this section we discuss about motivations and some basic concepts in question classification.

2.2 Why Question Classification?

Question classification has a key role in automated QA systems. Although different types of QA systems have different architectures, most of them follow a framework in which question classification plays an important role (Voorhees, 2001). Furthermore, it has been shown that the performance of question classification has significant influence on the overall performance of a QA system (Ittycheriah et al., 2001; Hovy et al., 2001; Moldovan et al., 2003).

Basically there are two main motivations for question classification: locating the answer and choosing the search strategy.

- **Locating the answer:** knowing the question class can not only reduce the search space need to find the answer, it can also find the true answer in a given set of candidate answers. For example knowing that the class of the question "who was the president of U.S. in 1934?" is of type "human", the answering system should only consider the name entities in candidate passages which is of type "human" and does not need to test all phrases within a passage to see whether it can be an answer or not.

- **Choosing search strategy:** question class can also be used to choose the search strategy when the question is reformed to a query over IR engine. For example consider the question "What is a pyrotechnic display ?". Identifying that the question class is "definition", the searching template for locating the answer can be for example "pyrotechnic display is a ..." or "pyrotechnic displays are ...", which are much better than simply searching by question words.

Even in non-IR-based QA systems, question classification have an important role. Popescu et al. (2003) for example, developed a QA system over a structured database which uses question class to generate proper SQL query over the database.

2.3 Question Classification Approaches

There are basically two different approaches for question classification: rule-based and learning based. There is also some hybrid approaches which combine rule-based and learning based approaches (Huang et al., 2008; Ray et al., 2010; Silva et al., 2011).

Rule based approaches try to match the question with some manually hand-crafted rules (Hull, 1999; Prager et al., 1999). These approaches however, suffer from the need to define too many rules (Li and Roth, 2004). Furthermore, while rule-based approaches may perform well on a particular dataset, they may have quite a poor performance on a new dataset and consequently it is difficult to scale them. Li and Roth (2004) provided an example which shows the difficulty of rule-based approaches. All the following samples are same question which has been reformulated in different syntactical forms:

- What tourist attractions are there in Reims?
- What are the names of the tourist attractions in Reims?
- What do most tourist visit in Reims?
- What attracts tourists to Reims?
- What is worth seeing in Reims?

All the above questions refer to same class while they have different syntactical forms and therefore they need different matching rules. So it is difficult to make a manual classifier with a limited amount of rules.

Learning-based approaches on the other hand, perform the classification by extracting some features from questions, train a classifier and predicting the class label using the trained classifier. Many successful learning-based classification approaches have been proposed. Later in chapter 5 we will discuss about learning-based approaches in more details.

There are also some studies that use both rule-based and learning based approaches together. The study of Silva et al. (2011), which is one of the most successful works on question classification, first match the question with some pre-defined rules and then use

the matched rules as features in the learning-based classifier. The same approach is used in the work by Huang et al. (2008).

Since learning-based and hybrid methods are the most successful approaches on question classification and most of the recent works are based on these approaches, in this paper we mainly review the learning and hybrid approaches of question classification.

2.4 Question Type Taxonomies

The set of question categories (classes) are usually referred as *question taxonomy* or *question ontology*. Different question taxonomies have been proposed in different works, but most of the recent studies are based on a two layer taxonomy proposed by Li and Roth (2002). This taxonomy consists of 6 coarse-grained classes and 50 fine-grained classes. Table 2.1 lists this taxonomy.

Table 2.1: The coarse and fine grained question classes.

Coarse	**Fine**
ABBR	abbreviation, expansion
DESC	definition, description, manner, reason
ENTY	animal, body, color, creation, currency, disease, event, food, instrument, language, letter, other, plant, product, religion, sport, substance, symbol, technique, term, vehicle, word
HUM	description, group, individual, title
LOC	city, country, mountain, other, state
NUM	code, count, date, distance, money, order, other, percent, percent, period, speed, temperature, size, weight

There are also other well-known question taxonomies use for question classification. The taxonomy proposed by Hermjakob et al. (2002) consists of 180 classes which is the broadest question taxonomy proposed until now.

Most of the recent learning-based and hybrid approaches use the taxonomy proposed by Li and Roth (2002) since the authors published a valuable set of 6000 labeled questions. This dataset consists of two separate set of 5500 and 500 questions in which the first is used as training set and the second is used as an independent test set. This dataset[1] which first published in University of Illinois Urbana-Champaign (UIUC) usually referred as the UIUC dataset and sometimes referred as the TREC dataset since it is widely use in the Text REtrieval Conference (TREC).

Metzler and Croft (2005) enhanced UIUC taxonomy with two more classes namely *list* and *yes-no-explain*. They created a separate dataset of 250 questions collected from MadSci[2] questions archive. MadSci is a scientific website which provides a framework in which users can ask a scientific question and receive an answer from an expert.

[1] http://cogcomp.cs.illinois.edu/Data/QA/QC/

[2] http://www.madsci.org/

2.5 Decision Model

Many supervised learning approaches have been proposed for question classification (Li and Roth, 2002; Blunsom et al., 2006; Huang et al., 2008). These approaches mainly differ in the classifier they use and the features they extract.

Most of the studies assume that a question is unambiguous, i.e., it has only one class and therefore assign the question to the most likely class. Some other studies (Li and Roth, 2002, 2004) on the other hands, have more flexible strategy and can assign multiple labels to a given question.

If the set of possible classes represented by $C = \{c_1, c_2, ..., c_m\}$ then the task of a question classifier is to assign the most likely class c_i to a question q_j if the question can only belong to one class. If a question can belong to more than one class then the decision model will be different. For example in the work of Li and Roth (2002), they rank the classes according to posterior probabilities and select top k classes as class labels of a given question. The value of k will be chosen based on the following criteria:

$$k = \min(t, 5) \text{ s.t. } \sum_{i=1}^{t} p_i \geq T \tag{2.1}$$

such that p_i is the posterior probability of the i-th chosen label. The indexes are set in such a way that $p_1 \geq p_2 \geq ... \geq p_m$. The parameter T is a threshold parameter in $[0, 1]$ which is chosen experimentally. In their work, Li and Roth (2002), considered T as 0.95 implying that with probability of 95% the true label of the question is one of the k chosen labels.

Most of the studies however, consider only one label for a given question ($k = 1$) (Zhang and Lee, 2003; Huang et al., 2008; Silva et al., 2011).

2.6 Performance Metrics in Question Classification

Typically, the performance of a question classifier is measured by calculating the accuracy of that classifier on a particular test set. The accuracy in question classification is defined as follow:

$$Accuracy = \frac{no.\ of\ Correctly\ Classified\ Samples}{Total\ no.\ of\ Tested\ Samples} \tag{2.2}$$

There are also two class-specific performance metrics: *precision* and *recall*, which can be used in question classification problem. The precision and recall of a classifier on a particular class c are defined as follow:

$$Precision[c] = \frac{no.\ of\ Samples\ Correctly\ Classified\ as\ c}{no.\ of\ Samples\ Classified\ as\ c} \tag{2.3}$$

$$Recall[c] = \frac{no.\ of\ Samples\ Correctly\ Classified\ as\ c}{Total\ no.\ of\ Samples\ in\ Class\ c} \tag{2.4}$$

For the systems in which a question can only have one class, a question is correctly classified if the predicted label is the same as the true label. But for the systems which allow a question to be classified in more than one class (Li and Roth, 2002, 2004), a question is correctly classified, if one of the predicted labels is the same as the true label.

Chapter 3

Classification Model

3.1 Introduction

Question classification can be done by different approaches as it described in the previous chapter. In this work we developed a statistical learning-based question classifier which outperforms state-of-the-arts works on this task. In this chapter we provide a detailed explanation of our system and give a close examination of the techniques that we used to obtain the optimal classifier. In the next chapter we explain our experimental results and evaluate the performance of our system against a standard dataset.

In this chapter we first give a mathematical definition of question classification and describe the architecture of our system. We then describe our motivations to choose Support Vector Machines (SVM) and Back-propagation Neural Networks (BPNN) as our classifiers and give an explanation of their structure. After that, we give a detailed explanation of the features we used and propose a weighted approach to combine them. Furthermore, we propose an alternating approach for our system by applying a feature reduction technique and finally we give a summary of our system.

3.2 System Architecture

Given a question, our classifier extracts different features out of it, combine them and classify it to one of the predefined classes. Suppose that the combined feature space has d dimensions. A question can be represented as $\mathbf{x} = (x_1, ..., x_d)$ in which x_i is the i^{th} feature in combined space. The classifier is a function which maps the question $\mathbf{x}$ to a class c_i from the set of classes $C = \{c_1, ..., c_m\}$. This function is learned over a training set of labeled questions. Figure 3.1 illustrates the overall architecture of our question classifier system. The system first extracts different set of features from a question and then optimally combines them. The combined features will be given to the trained classifier and it predicts the most likely class label.

To build such a classifier system, two main challenges should be addressed: 1)what type of classifier to choose and how to train that, 2)how to extract features and optimally combine them. We tested two different classifiers: support vector machines and back-

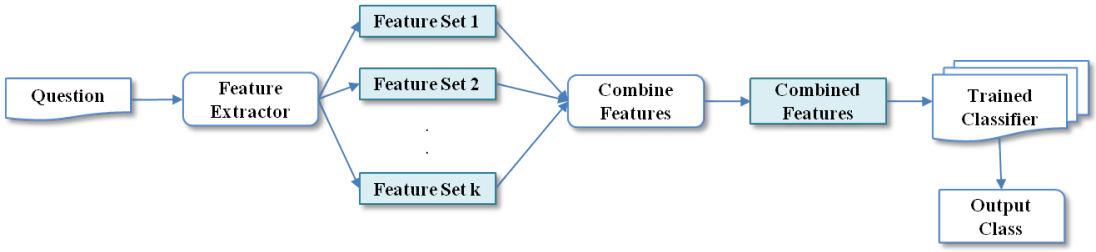

Figure 3.1: The overall architecture of our supervised question classifier system

propagation neural networks. We later describe our motivations for choosing these two classifiers. We extracted different type of lexical, syntactical and semantic features from a question. What make our system different from the state-of-the-art methods, are the richer feature space that is extracted from questions and the wighted approach to combine them. We also used neural networks for question classification for the first time. We provide a detailed comparison of our system with the state-of-the-art systems in chapter 5.

3.3 Classifiers

Question classification has been studied by using different type of classifiers. Most of the successful studies on this task uses support vector machines (Zhang and Lee, 2003; Huang et al., 2008; Silva et al., 2011; Loni et al., 2011). SVMs are very successful on high dimensional data since they are timely efficient especially when the feature vectors are sparse. Question classification has also been done by Maximum Entropy models (Huang et al., 2008; Blunsom et al., 2006), Sparse Network of Winnows (SNOW) (Li and Roth, 2004) and language modeling (Merkel and Klakow, 2007).

In this work we adopted SVMs as well as back-propagation neural networks. Training a neural network with high dimensional vectors such as questions, demands very large networks which make them very costly to train. However, by applying LSA feature reduction technique, we can train smaller yet efficient networks in a reasonable time which makes them suitable to be used for question classification. To our knowledge this is the first work which uses neural networks for question classification. In this section we briefly describe the classifiers we used.

3.3.1 Support Vector Machines

Support vector machine is a supervised learning method for classifying data. It is especially successful for high dimensional data. SVM is a linear discriminant model which tries to learn a hyperplane with maximum margin for separating the classes.

Suppose we are given a training set $(\mathbf{x}_i, y_i), i = 1, ..., n$, in which $\mathbf{x}_i = (x_{i1}, ..., x_{id})$ is a d-dimensional sample and $y_i \in \{1, -1\}$ is the corresponding label. The task of a support vector classifier is to find a linear discriminant function $g(x) = w^T\mathbf{x} + w_0$, such

that $w^T\mathbf{x}_i + w_0 \geq +1$ for $y_i = +1$ and $w^T\mathbf{x}_i + w_0 \leq -1$ for $y_i = -1$. Therefore we seek for a solution such that the following condition holds:

$$y_i(w^T\mathbf{x}_i + w_0) \geq 1 \quad i = 1, ..., n \tag{3.1}$$

The optimal linear function is obtained by minimizing the following quadratic programming problem (Vapnik, 1995):

$$\min \ \frac{1}{2}w^T w - \sum_{i=1}^{n} \alpha_i(y_i(w^T\mathbf{x}_i + w_0) - 1) \tag{3.2}$$

which leads to the following solution:

$$w = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{3.3}$$

where $\{\alpha_i, i = 1, ..., n; \alpha_i \geq 0\}$ are Lagrange multipliers. To be able to linearly separate data, typically the feature space should be mapped to a higher dimensional space. The mapping is done with a so-called *kernel function*.

The kernel is a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which takes two samples from input space and map it to a real number indicating their similarity. For all $\mathbf{x}_i$, $\mathbf{x}_j \in \mathcal{X}$, the kernel function satisfies:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \tag{3.4}$$

where ϕ is an explicit mapping from input space $\mathcal{X}$ to a *dot product* feature space $\mathcal{H}$ (Hofmann et al., 2008).

To apply kernel functions on SVM classifier, typically the dual form of (3.2) is solved:

$$\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i.\mathbf{x}_j \tag{3.5}$$

where $\mathbf{x}_i.\mathbf{x}_j$ is the inner product of two samples which is an implicit kernel in the equation measuring similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. This inner product can be replaced by another kernel function leading equation (3.5) to be in the following form:

$$\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{3.6}$$

There are four types of basic kernel functions: linear, polynomial, radial basis function and sigmoid. Other types of custom kernel functions can also be applied for question classification. The simplest type of linear kernel for two question $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as follows:

$$K_{LINEAR}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^{d} x_{il} x_{jl} \tag{3.7}$$

which is simply the inner product of the two questions.

Based on experimental results we noticed that linear kernels had higher performance compare to other types of kernel functions; therefore we choose linear kernels in the final model. We provide a comparison between different types of kernels in the next chapter.

Support vector machines have been widely used in question and text classification due to its good performance, both in time and accuracy, compare to other type of classifiers (Zhang and Lee, 2003; Metzler and Croft, 2005; Huang et al., 2008; Silva et al., 2011).

In QC problem, as you will see in the next section, questions are typically represented in a very high dimensional space although the feature vectors are very sparse. SVMs usually have good performance for high dimensional data. They are computationally cheap, because the main operation of a SVM is to calculate inner product of two vectors which is an easy process particularly when the vectors are sparse. Since SVMs are only applied in two-class classification problems, typically a so-called *one-against-all* strategy is chosen when number of classes is more than two (Webb, 2002).

For the cases where data are non-separable, the constrain (3.1) can be relaxed as follows:

$$y_i(w^T\mathbf{x}_i + w_0) \geq 1 - y_i\xi_i \quad i = 1, ..., n \tag{3.8}$$

where ξ_i is a positive *slack* variable. For an error to occur, the corresponding ξ_i should be increased unity. So $\sum_i \xi_i$ is an upper bound for the number of training errors (Burges, 1998). To minimize the training error this sum should be considered in the objective function. So equation (3.2) can be rewritten as follows:

$$\min \ \frac{1}{2}w^T w - \sum_{i=1}^{n} \alpha_i(y_i(w^T\mathbf{x}_i + w_0) - 1) + C\sum_{i=1}^{n} \xi_i \tag{3.9}$$

where C is a penalty parameter for the errors on training set. We experimentally found that $C = 1$ obtains the best classification accuracy (see chapter 4).

In this work, we adopted LIBSVM (Chang and Lin, 2001), a library for support vector machines which is implemented in Java.

3.3.2 Back-Propagation Neural Networks

Back-propagation neural networks are multi-layer feed forward neural networks which are trained with the back-propagation learning rule (Hu, 2000). They consist of an input layer, an output layer and one or more hidden layers. Each neuron has a forward connection to all neurons in the subsequent layer and the importance of connections is reflected by weight parameters. The input of each neuron is weighted sum of its input signals and the output is calculated as a function of input signals and an optional threshold parameter.

Training the BBNN

Consider we are given a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ such that $\mathbf{x}_i = (x_{i1}, ..., x_{id})$ is a d-dimensional input vector and y_i is its corresponding class label which takes one of the values from the set of labels $C = \{c_1, ..., c_m\}$. To build a network based on our training set, the number

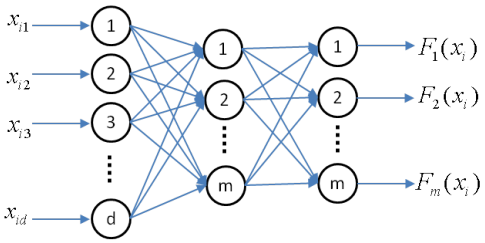

Figure 3.2: The structure of network we used in this work.

of input neurons should be equal to d, the number of features, and the number of output neurons should be set to m, the number of classes. The number of hidden layers and neurons in each layer should be learned or specified in advance. Figure 3.2 depicts the structure of a network with one hidden layer in which the number of hidden neurons is equal to the number of output neurons. For an input vector $\mathbf{x}_i = (x_{i1}, ..., x_{id})$, the input of each neuron in input layer is fed with exactly one feature of $\mathbf{x}_i$. The network generates m outputs. The class label of $\mathbf{x}_i$ is determined by a max rule:

$$c = \arg\max_{k=1}^{m} F_k(\mathbf{x}_i), \tag{3.10}$$

where $F_k(\mathbf{x}_i)$ is the output value generated by neuron k in output layer and c is the index of the predicted class.

According to the defined notations, for a given input vector $\mathbf{x}_i$, the input of a hidden node j is defined as the following weighted sum:

$$\phi_j = \sum_{k=1}^{d} w_{kj} x_{ik}, \tag{3.11}$$

where w_{kj} indicates the weight in the link between input neuron k and hidden neuron j. The output of a hidden unit is calculated as follow:

$$\psi_j = f(\phi_j + \theta_j), \tag{3.12}$$

such that θ_j is the threshold parameter of hidden unit j and f is a non-linear transformation which is referred as the *activation function*. Our experimental results show that the sigmoid activation function performs better than other types of functions (see chapter 4). The sigmoid function is defined as:

$$f(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{x})} \tag{3.13}$$

The input and output of the output layer are also calculated using (3.11) and (3.12). The back-propagation learning rule initializes weight and threshold parameters randomly

and iteratively updates these, using a *gradient descent method* such that the error on the training set converges to a small value. The error on the training set is defined as:

$$E = \sum_{k=1}^{m} \sum_{i=1}^{n} (F_k(\mathbf{x}_i) - Y_k(\mathbf{x}_i))^2 \tag{3.14}$$

where $F_k(\mathbf{x}_i)$ is the output generated by neuron k in the output layer for sample $\mathbf{x}_i$ and $Y_k(\mathbf{x}_i)$ is the desired output of the same neuron for $\mathbf{x}_i$ which is defined as follows:

$$Y_k(\mathbf{x}_i) = \begin{cases} 1 & \text{if} \quad y_i = k \\ 0 & \text{o.w.} \end{cases} \tag{3.15}$$

The gradient descent method updates the weight values as follows:

$$W^{t+1} = W^t + \Delta W^{t+1} \tag{3.16}$$

such that:

$$\Delta W^{t+1} = \alpha \frac{\partial E}{\partial W^t} + \beta \Delta W^{t-1} \tag{3.17}$$

where W^t is the matrix of weights in iteration t, $\alpha \in [0, 1]$ is the learning rate which specifies how fast the gradient descent method should update the weight values and $\beta \in [0, 1]$ is a constant which specifies the contribution of previous iteration. The weights are updated until the error converges to a small value or the algorithm reaches the maximum number of iterations. We used Neroph[1], a Java framework for neural networks, to implement our classifier.

3.4 Features

In question classification problem, different set of features can be extracted. The features in question classification can be categorized into 3 different types: lexical, syntactical and semantic features.

In order to obtain the best feature space for question classification, we extracted several lexical, syntactical and semantic features from a question. In this section we explain our approach to extract features from a question and investigated the role of each feature in question classification.

3.4.1 Lexical Features

Lexical features of a question are generally extracted based on the *context words* of the question, i.e., the words which appear in a question. In question classification task, a question is represented similar to document representation in *vector space model*, i.e., a question is a vector which is described by the words inside it. Therefore a question $\mathbf{x}$ can be represented as:

[1] http://neuroph.sourceforge.net/

$$\mathbf{x} = (x_1, x_2, ..., x_N) \tag{3.18}$$

where x_i is defined as the frequency of term i in question $\mathbf{x}$ whereas N is total number of terms. Due to sparseness of feature vector only non-zero valued features are kept in feature vector. Therefore sometimes the question is also represented by the following form:

$$\mathbf{x} = \{(t_1, f_1), ..., (t_p, f_p)\} \tag{3.19}$$

where t_i is the i^{th} term in question $\mathbf{x}$ and f_i is its frequency in the question, respectively given that the question $\mathbf{x}$ has p unique terms. This feature space is called *bag-of-words* features. As the name suggests, the order of the words is the question is not important in this form of representation. Representing the questions in the form of (3.19) makes the size of samples quite small despite the huge size of feature space. As an example consider the question "How many Grammys did Michael Jackson win in 1983 ?" This question can be represented as follows based on representation (3.19):

$$\begin{aligned}\mathbf{x} = \{&(\text{How}, 1), (\text{many}, 1), (\text{Grammys}, 1), (\text{did}, 1), (\text{Michael}, 1), (\text{Jackson}, 1),\\ &(\text{win}, 1), (\text{in}, 1), (1983, 1), (?, 1)\}\end{aligned} \tag{3.20}$$

The frequency of the words in question (feature values) can be viewed as a weight value which reflects the importance of a word in a question. We later exploited this characteristic to weight the features based on their importance when we combine different feature sets.

Bag-of-word feature space is also referred as *unigrams*. Unigrams is a special case of the so-called *n-gram* features. To extract n-gram features, any n consecutive words in a question is considered as a feature. For example "How-many" in the above example is a *bigram* features and can be added to the feature vector. All the lexical, syntactical and semantic features can be added to the feature space and expand the above feature vector. In fact the expanded feature vector still can be represented similar to (3.19), while the new features can be considered as new terms. For example the bigram feature "How-many" can be viewed as a new term and the pair {(How-many, 1)} will be added to the feature vector when bigram features are extracted. Of course this will increase the size of feature space and the questions will be represented with higher dimensional vectors.

Bigram features however, are very high dimensional since all two consecutive terms in our dataset should be considered as features, of which most are redundant and do not show up in the data. We found that considering only the first two words of a question as bigram features, performs as good as all bigrams while the size of feature space is much smaller. For example consider the question "How many people in the world speak French?". The only meaning bigram in this question is "How-many" while the rest is not useful. That is also true in the questions in which the wh-word is one word, because the combination of wh-word and the immediate word next to it is an informative feature in most cases. For example most of the questions which starts with "what is/are" are asking for a definition. In the rest of the paper, we call our limited bigrams feature space as *limited bigrams*.

Other type of lexical features can be extracted from a question. Huang et al. (2008, 2009) considers question *wh-words* as a separate feature. They adapted 8 types of wh-words, namely *what*, *which*, *when*, *where*, *who*, *how*, *why* and *rest*. For example the wh-word feature of the question "What is the longest river in the world?" is *what*. Considering wh-words as a separate feature can improve the performance of classification according to the experimental studies.

Yet another kind of lexical feature is *word shapes*. It refers to apparent properties of single words. Huang et al. (2008) introduced 5 categories for word shapes: *all digit*, *lower case*, *upper case*, *mixed* and *other*.

Blunsom et al. (2006) introduced question's length as a separate lexical feature. It is simply the number of words in a question. Table 3.1 lists the lexical features of the sample question "How many Grammys did Michael Jackson win in 1983 ?". The features are represented in same form as equation (3.19).

Table 3.1: Example of lexical features

Feature Space	**Features**
unigram	{(How, 1) (many, 1) (Grammys, 1) (did, 1) (Michael, 1) (Jackson, 1) (win, 1) (in, 1) (1983, 1) (?, 1)}
bigram	{(How-many, 1) (many-Grammys, 1) (Grammys-did, 1) (did-Michael, 1) (Michael-Jackson, 1) (Jakson-win, 1) (win-in, 1) (in-1983, 1) (1983-?, 1) }
trigram	{(How-many-Grammys, 1), (many-Grammys-did, 1), ..., (in-1983-?, 1)}
wh-word	{(How, 1)}
word-shapes	{(lowercase, 4) (mixed, 4) (digit, 1) (other, 1)}
question-length	{(question-len, 10)}

3.4.2 Syntactic Features

A different class of features can be extracted from the syntactical structure of a question. We extracted different type of syntactical features.

POS Tags and Tagged Unigrams

POS tags indicate the part-of-speech tag of each word in a question such as NN (Noun), NP (Noun Phrase), VP (Verb Phrase), JJ (adjective), and etc. The following example shows the question "How many Grammys did Michael Jackson win in 1983 ?" with its POS taggs:

How_WRB many_JJ Grammys_NNPS did_VBD Michael_NNP Jackson_NNP win_VBP in_IN 1983_CD ?_.

POS tagging can be done with different approaches. There are many successful learning-based approaches including unsupervised methods (Clark, 2000) and Hidden Markov Models (Schütze and Singer, 1994) with 96%-97% accuracies. In this work we used Stanford log-linear POS tagger (Toutanova et al., 2003).

Some studies in question classification add all the POS tags of question in feature vector (Li and Roth, 2004; Blunsom et al., 2006). This feature space sometimes referred as *bag-of-*POS *tags*. The bag-of-POS features for the aforementioned example is as follows:

$$\{(\text{WRB}, 1), (\text{JJ}, 1), (\text{NNPS}, 1), (\text{VBD}, 1), (\text{NNP}, 1), (\text{NNP}, 1), (\text{VBP}, 1), (\text{IN}, 1), (\text{CD}, 1)\} \tag{3.21}$$

We introduced a feature namely *tagged unigram* which is simply the unigrams augmented with POS tags. Considering the tagged unigrams instead of normal unigrams can help the classifier to distinguish a word with different tags as two different features. Following is the aforementioned example represented with tagged unigram features:

$$\{(\text{How\_WRB}, 1), (\text{many\_JJ}, 1), (\text{Grammys\_NNPS}, 1), (\text{did\_VBD}, 1), (\text{Michael\_NNP}, 1), (\text{Jackson\_NNP}, 1), (\text{win\_VBP}, 1), (\text{in\_IN}, 1), (\text{1983\_CD}, 1), (?\_, 1)\} \tag{3.22}$$

POS tag information can also be used for extracting semantic features. As you can see in the next section, POS tags can be used to disambiguate the meaning of a word to extract semantic features.

Head Words

A head word is usually defined as the most informative word in a question or a word that specifies the object that question seeks (Huang et al., 2008). Identifying the headword correctly, can significantly improve the classification accuracy since it is the most informative word in the question. For example for the question "What is the oldest city in Canada ?" the headword is "city". The word "city" in this question can highly contribute the classifier to classify this question as "LOC:city". Table 3.2 lists 20 sample questions from TREC dataset together with their class label. The headwords are identified by boldface. This table shows the strong relation between headwords and class label. As you might see there is no suitable headword for questions of type "Definition" or "reason".

Extracting question's headword is quite a challenging problem. The headword of a question usually extracted based on the syntactical structure of the question. To extract the headword we first need to parse the question to form the *syntax tree*. The syntax (parse) tree is a tree that represents the syntactical structure of a sentence base on some grammar rules. For natural language sentences written in English language, English grammar rules are used to create syntax tree. Figure 3.3 is an example of syntax tree for the question "What is the oldest city in Canada?".

There are successful parsers that can parse a sentence and form the syntax tree (Klein and Manning, 2003; Petrov and Klein, 2007). These parsers are statistical-based parsers

Table 3.2: Sample questions from TREC dataset together with their class label. The question's headword is identified by boldface.

Question	Category
What **county** is Modesto , California in ?	LOC:city
Who was **Galileo** ?	HUM:desc
What is an **atom** ?	DESC:def
What is the name of the chocolate **company** in San Francisco ?	HUM:gr
George Bush purchased a small interest in which baseball **team** ?	HUM:gr
What is Australia 's national **flower** ?	ENTY:plant
Why does the moon turn orange ?	DESC:reason
What is autism ?	DESC:def
What **city** had a world fair in 1900 ?	LOC:city
What is the average **weight** of a Yellow Labrador ?	NUM:weight
Who was the first **man** to fly across the Pacific Ocean ?	HUM:ind
What day and **month** did John Lennon die ?	NUM:date
What is the life **expectancy** for crickets ?	NUM:other
What **metal** has the highest melting point ?	ENTY:substance
Who developed the **vaccination** against polio ?	HUM:ind
What is epilepsy ?	DESC:def
What **year** did the Titanic sink ?	NUM:date
What is a biosphere ?	DESC:def
What **river** in the US is known as the Big Muddy ?	LOC:other
What is the **capital** of Yugoslavia ?	LOC:city

which parse an English sentence based on *Probabilistic Context-Free Grammars* (PCFG) in which every rule is annotated with the probability of that rule being used. The rule's probabilities was learned based on a supervised approach on a training set of 4,000 parsed and annotated questions known as treebank (Judge et al., 2006). These parsers typically maintain an accuracy of more than 95%. Jurafsky and Martin (2008) provided a detailed overview of parsing approaches. The list of English POS tags which is used for parsing syntax tree is listed in appendix A. In this work we used Standford PCFG parser (Petrov and Klein, 2007).

The idea of headword extraction from syntax tree first was introduced by Collins (Collins, 1999). He proposed some rules, known as Collins rules, to identify the headword of sentence. Consider a grammar rule $X \rightarrow Y_1...Y_n$ in which X and Y_i are non-terminals in a syntax tree. The head rules specifies which of the right-hand side non-terminals is the head of rule X. For example for the rule SBARQ $\rightarrow$ WHNP SQ, Collins rules specifies that the head is in the SQ non-terminal. This process continues recursively until a terminal node is reached.

To find the headword of a sentence, the parse tree is traversed top-down and in each level the subtree which contains the headword is identified with Collins head rules. The algorithm continues on the resulting subtree until it reaches a terminal node. The resulting

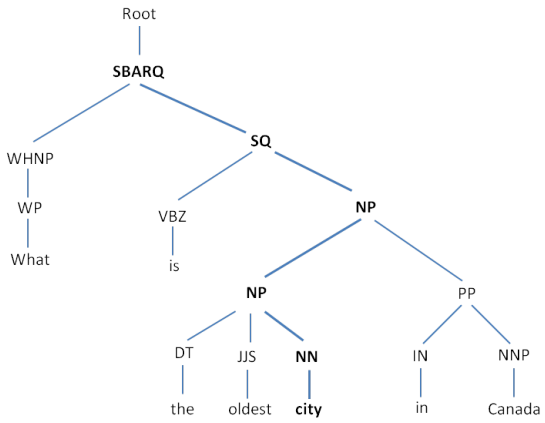

Figure 3.3: The syntax tree of a sample question in which the head childs are specified by boldface

node is the sentence's headword.

For the task of question classification, however, Collins rules are not suitable since they have preferences for verb phrases over noun phrases whereas in a question the headword should be a noun. We modified the Collins rules to properly extract a question's headword. In fact, in the modified rules we set a preference of noun phrases over verb phrases. Table 3.3 lists the modified rules. The first column of the table is the non-terminal on the left side of a production rule. The second column specifies the direction of search in the right hand side of a production rule. The search can be either by *category*, which is the default search method, or by *position*. If the direction of search is left by category then the algorithm starts from the left-most child and checks it against items in priority list (column 3 in table 3.3) and if it matches any, then the matched item will be returned as head. Otherwise if the algorithm reaches the end of the list and the child does not match with any of the items, it continues the same process with the next child.

On the other hand, if the search direction is left by position, then the algorithm first starts checking the items in priority list and for each item it tries to match it with every child from left to right. The first matched item is considered as head.

Algorithm 1 lists the headword extraction algorithm based on Collins modified rules (Silva et al., 2011).

To follow the algorithm consider the parse tree of the question "What is the oldest city in Canada ?". The parse tree of this question is depicted in figure 3.3 in which the path of finding the headword is specified by boldface. The procedure Apply-Rules, finds a child of the parse tree which contains the headword, based on the modified Collins rules.

Now if we trace the algorithm for the sample in figure 3.3, it starts from top of the tree with the production rule SBARQ $\rightarrow$ WHNP SQ. The direction of search for the rule SBARQ is left by category. Therefore the algorithm starts with WHNP and checks it against items in the priority list of the rule SBARQ. Because none of the items in this list match with WHNP the algorithm continues with next child. Since the next child appears

Table 3.3: Modified Collins rules for determining question's headword

Parent	**Direction**	**Priority List**
ROOT	Left by Category	S, SBARQ
S	Left by Category	VP, FRAG, SBAR, ADJP
SBARQ	Left by Category	SQ, S, SINV, SBARQ, FRAG
SQ	Left by Category	NP, VP, SQ
NP	Right by Position	NP, NN, NNP, NNPS, NNS, NX
PP	Left by Category	WHNP, NP, WHADVP, SBAR
WHNP	Left by Category	NP, NN, NNP, NNPS, NNS, NX
WHADVP	Left by Category	NP, NN, NNP, NNPS, NNS, NX
WHADJP	Left by Category	NP, NN, NNP, NNPS, NNS, NX
WHPP	Right by Category	WHNP, WHADVP, NP, SBAR
VP	Right by Category	NP, NN, NNP, NNPS, NNS, NX, SQ, PP
SINV	Left by Category	NP
NX	Left by Category	NP, NN, NNP, NNPS, NNS, NX, S

Algorithm 1 Headword extraction algorithm

```
procedure Extract-Question-Headword (tree)
  if IsTerminal(tree) then
    return tree
  else
    head-child ← Apply-Rules(tree)
    return Extract-Question-Headword (head-child)
  end if
end procedure
```

in the priority list it is considered as head. With similar way the non-terminal NP will be selected in the production rule SQ → VBZ NP as the head child. The algorithm continues until it reaches the terminal node "city" and return it as the headword.

The aforementioned algorithm for extracting a question's headword can not always determine the true headword. For example for the question "Which country are Godiva chocolate from ?" the true headword is "country" while the algorithm will return "chocolate" as the headword. Figure 3.4 depicts the syntax tree of this question in which the head children are specified by boldface. Applying the trivial rules of algorithm 1 will choose SQ in the production rule SBARQ → WHNP SQ which leads the procedure to determine an incorrect headword.

To tackle this problem, Silva et al. (2011) introduced some *non-trivial* rules which are applied to a parse tree before applying the trivial rules. For example if SBARQ rule contains a WHXP[2] child with at least two children then WHXP is returned as head child. Considering this rule leads to correctly identifying headword in the sample of figure 3.4.

[2]WHXP refers to Wh-phrases: WHNP, WHPP, WHADJP, WHADVP

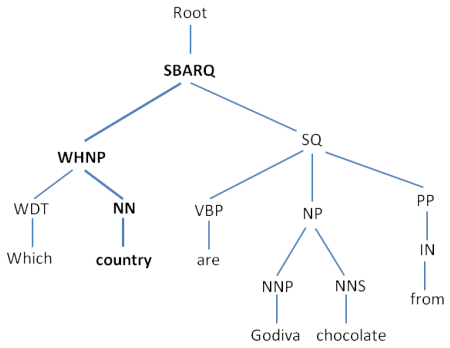

Figure 3.4: The syntax tree of a sample question in which the head childs are specified by boldface. The headword in this question can not be determined correctly using the trivial rules

A question's headword can not only be used directly as a feature, but it is also used to enhance the feature space with semantic features.

Head Rules

As we mentioned before some questions inherently do not have any head word. For example for the question "What is biosphere ?" there is no suitable head word as the entity type of the only noun in this question (biosphere) does not contribute to classify this question as "definition". Same problem exist for the question "Why does the moon turn orange ?". None of the words in this question except the wh-word help the classifier to classify this question as "reason".

To define an alternate feature instead of head word for these type of questions, Huang et al. (2008) introduces some regular expression patterns to map these types of questions to a pattern and then uses the corresponding pattern as a feature. Table 3.4 lists the patterns from Huang et al. (2008).

We have also implemented head-rules with the same set of rules to investigate the contribution of this feature space in our classifier. The above 8 patterns are considered as 8 features. If a question matches with any of the above rules the corresponding feature will be true. The representation of these features is similar to (3.19), i.e., the pattern name can be viewed as a term and the feature vale would be 1 if the question match with that pattern. For example for the question "What is biosphere ?", its head rule feature can be represented as follows:

$$\{(\text{DESC:def-pattern1}, 1)\} \tag{3.23}$$

Table 3.5 lists the syntactical features discussed in this section for the sample question "What is the oldest city in Canada ?". The features are represented same as equation 3.19.

Table 3.4: Regular expression rules to identify pattern in questions; taken from Huang et al. (2008)

Name	**Pattern**
DESC:def pattern 1	the question begins with a *what is/are* and follows by an optional *a, an* or *the* and then follows by one or two words
DESC:def pattern 2	the question begins with *what do/does* and ends with *mean*
ENTY:substance pattern	the question begins with *what does* and end with *do*
ENTY:term pattern	the question begins with *what do you call*
DESC:reason pattern 1	the question begins with *what causes/cause*
DESC:reason pattern 2	the question begins with *what is/are* and ends with *used for*
ABBR:exp pattern	the question begins with *what does/do* and ends with *stand for*
HUM:desc pattern	the question begins with *who is/was* and follows by a word starting with a capital letter

Table 3.5: Example of syntactic features

Feature Space	**Features**
tagged unigram	{(What_WP, 1) (is_VBZ, 1) (the_DT, 1) (oldest_JJS, 1) (city_NN, 1) (in_IN, 1) (Canada_NNP, 1) (?_, 1)}
POS tags	{(WP, 1) (VBZ, 1) (DT, 1) (JJS, 1) (NN, 1) (IN, 1) (NNP, 1)}
headword	{(city, 1)}
head rule	{}

3.4.3 Semantic Features

Semantic features are extracted based on the semantic meaning of the words in a question. We extracted different type of semantic features. Most of the semantic features requires a third party data source such as WordNet (Fellbaum, 1998), or a dictionary to extract semantic information for questions.

Hypernyms

WordNet is a lexical database of English words which provides a lexical hierarchy that associates a word with higher level semantic concepts namely *hypernyms*. For example a hypernym of the word "city" is "municipality" of which the hypernym is "urban area" and so on. Figure 3.5 shows the hypernym hierarchy of sense 3 of the word "capital".

As hypernyms allow one to abstract over specific words, they can be useful features for question classification. Extracting hypernyms however, is not straightforward. There are four challenges that should be addressed to obtain hypernym features:

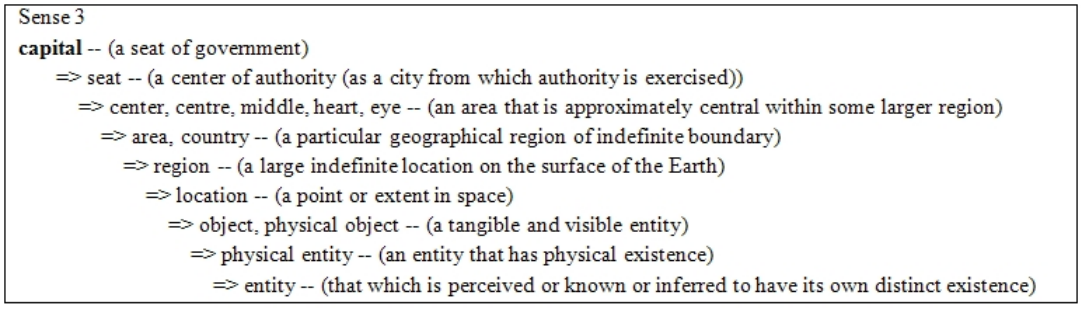

Figure 3.5: WordNet Hypernyms hierarchy for sense 3 of the word "capital"

1. For which word(s) in the question should we find hypernyms?
2. For the *candidate* word(s), which part-of-speech should be considered?
3. The candidate word(s) augmented with their part-of-speech may have different senses in WordNet. Which sense is the sense that is used in the given question?
4. How far should we go up through the hypernym hierarchy to obtain the optimal set of hypernyms?

To address the first challenge we considered two different scenarios: either to consider the headword as the candidate word for expansion or expanding all the words in the question by their hypernyms. We found out that the second approach can introduce noisy information to the feature vector and therefore decided to consider only the headword of a question, if it has, as the candidate for expansion.

For the second issue the POS tag which extracted from syntactical structure of question is considered as the target POS tag of the chosen candidate word.

To tackle the third issue, the right sense of the candidate word should be determined to be expanded with its hypernyms. For example the word "capital" with noun POS can have two different meanings. It can either interpreted as "large alphabetic character" or "a seat of government". Each sense has its own hypernyms. For example "character" is a hypernym of the first sense while "location" is a hypernym of the second sense. In the question "What is the capital of Netherlands ?" for example, the second sense should be identified.

We adopted Lesk's Word Sense Disambiguation (WSD) algorithm to determine the true sense of word according to the sentence it appears. The Lesk's algorithm (Lesk, 1986) is a dictionary-based algorithm which works based on the assumption that words in a given context tends to share common topic. Algorithm 2 is the adopted Lesk's WSD algorithm which is borrowed from Huang et al. (2008). We used WordNet to find the definition of the words in this algorithm.

Algorithm 2 Adopted Lesk's WSD algorithm taken from Huang et al. (2008)

```
procedure Lesk-WSD (question, headword)
  int count ← 0
  int maxCount ← -1
  sense optimum = null
  for each sense s of headword do
    count ← 0
    for each contextWord w in question do
      int subMax ← maximum no. of common words in s definition and definition of any
      sense of w
      count ← count + subMax
    end for
    if count > maxCount then
      maxCount ← count
      optimum ← s
    end if
  end for
  return optimum
end procedure
```

For a given headword of a question, algorithm 2 computes the maximum number of common words between the gloss (definition) of each sense and the gloss of all senses of all context words. The sense with the maximum common words is considered the true sense.

To address the fourth challenge we found that expanding the headword with hypernyms of maximum dept 6 will have the best result. In the next chapter we will show the influence of hypernym's dept on classification accuracy.

Now consider for example the headword of the question "What is the capital of Netherlands ?". The true sense of the word "capital" according to its context is sense 3 of WordNet. The hypernym features of this word according to representation (3.19) with value 6 as the maximum dept will be as follow:

$$\{(\text{capital}, 1)(\text{seat}, 1)(\text{center}, 1)(\text{area}, 1)(\text{region}, 1)(\text{location}, 1)\} \tag{3.24}$$

The word "location" in the above features in fact can contribute the classifier to classify this question to LOC.

Related Words

Another semantic feature that we implemented is related words which was based on the idea of Li and Roth (2004). They defined groups of words, each represented by a category name. If a word in the question exists in one or more groups, its corresponding categories will be added to the feature vector. For example if any of the words {birthday, birthdate,

day, decade, hour, week, month, year} exists in a question, then its category name, *date*, will be added to the feature vector.

To expand the feature vector with related words, still we can choose to only consider the head word or the whole question. Our experimental results show that considering the whole question would have better results.

Question Category

We extracted a successful semantic feature namely *question category* which is obtained by exploiting WordNet hierarchy based on the idea of Huang et al. (2008).

We used WordNet hierarchy to calculate the similarity of question's headword with each of the classes. The class with highest similarity is considered as a feature and will be added to the feature vector. In fact this is equal to a *mini-classification* although the acquired class will not be used as final class; since it is not as accurate as the original classifier.

The similarity of the words will be calculated based on an information content metric using WordNet *hyponyms*. The subordinates of a particular word in WordNet hierarchy are called hyponyms. For example "city" is a hyponym of "municipality" based on the WordNet hierarchy. We used the metric proposed by Seco et al. (2004) to calculate words similarity. To obtain the similarity of two words, we first obtain the *Most Specific Common Abstraction* (MSCA) using WordNet hierarchy. The similarity then will be calculated on how much informative the MSCA word is. The similarity of two words w_1 and w_2 based on this idea is calculated as follows:

$$sim_{\text{MSCA}} = \max_{w \in S(w_1, w_2)} ic_{wn}(w) \tag{3.25}$$

where $S(w_1, w_2)$ are the set of words which subsumes w_1 and w_2 and $ic_{wn}(w)$ is the information content of the word w and is calculated as follows using WordNet hierarchy:

$$ic_{wn}(w) = \frac{\log(\frac{hypo(w)+1}{max_{wn}})}{\log(\frac{1}{max_{wn}})} = 1 - \frac{\log(hypo(w)+1)}{\log(max_{wn})} \tag{3.26}$$

where $hypo(w)$ returns the number of hyponyms of a given word and max_{wn} is the total number of word entries in WordNet. In order to calculate similarity of two words w_1 and w_2 we first extract the list of hypernyms of each word, i.e $S(w_1, w_2)$. The information content of the word which has the largest ic value will be considered as the similarity of w_1 and w_2. The information content of each element is calculated based on equation (3.26). The ic value will be a number in [0,1]. The larger the ic value is, the more informative it is. Based on the above equation the top node of WordNet hierarchy will have an ic value of 0. The words which are closer to this top node have smaller value compare to the node which are close to the leaf nodes. In fact the nodes which are located near the bottom of the hierarchy are more specific and consequently should be more informative compare to the top nodes.

To calculate the number of hyponyms of a given word, we first use Lesk's WSD algorithm to disambiguate the meaning of the word and then we recursively count the

hyponyms of the disambiguated word. We already calculated the hyponym counts of all the words in WordNet so that this value can simply be retrieved by a simple lookup.

Now consider the example "What metal has the highest melting point ?". The headword of this question is "metal". To find the question category feature, the similarity of this word will be compared with the similarity of all question categories. The category with the highest similarity will be added to the feature vector. In this example the most similar category is "substance" and therefore the question category feature will be {(substance, 1)}.

Query Expansion

We introduce a feature namely *query expansion* which is basically very similar to hypernym features. As we explained before, we add hypernym of a headword to the feature vector with maximum distance of 6 from the original headword in the WordNet hierarchy. Instead of imposing this limitation, we defined a weight parameter which decreases by increasing the distance of a hypernym from the original word. Consider that t_i is the headword in question $\mathbf{x}$. We define the set of hypernyms of t_i as follows:

$$H(t_i) = (h_{t_i,1}, ..., h_{t_i,q}) \tag{3.27}$$

such that $h_{t_i,j}$ is a hypernym of t_i with distance j from it and q is the total number of hypernyms of t_i. For example based on figure 3.5, $H(capital)$ =(seat, center, area, region, location, object, physical-object, entity) where $h_{capital,1} = city$ and $q = 8$. Using the distance parameter j we define the weight value for $h_{t_i,j}$ as follows:

$$W(h_{t_i,j}, j) = W_0(t_i)\gamma^{d(t_i,j)} \tag{3.28}$$

where $W_0(t_i)$ is the weight (frequency) of t_i in feature vector, $d(t_i, j)$ is a distance function which calculates the distance of t_i from its j^{th} hypernym and γ is a constant in [0,1]. There are different possibilities to define the distance function $d(t_i, j)$, but we simply defined it as the path length from t_i to $h_{t_i,j}$, that is $d(t_i, j)$ = j given that $j \leq q$. Therefore in the mentioned example, the distance between "capital" and city is 1 while the distance between "capital" and "entity" is 8. Having the γ value in [0,1], makes sure that the weight value of hypernyms will be decreased when the distance from the main word is increased. Our experiments shows that considering $\gamma = 0.6$ will obtain the best results.

Now consider the question "What river in the US is known as the Big Muddy ?" whose headword is "river". The query expansion features of this question will be as follows, given that the weight of "river" is considered as 1:

$$\{(\text{river}, 1)(\text{stream}, 0.6)(\text{body-of-water}, 0.36)(\text{thing}, 0.22)(\text{physical-entity}, 0.13)(\text{entity}, 0.08)\} \tag{3.29}$$

The hypernym features of the above example will be the same as (3.29) but all the feature values would be 1. The advantage of query expansion to the hypernym features is that the importance of the features is reflected by the weight values and this causes that the noisy information have less contribution on classification.

Table 3.6 lists the semantic features discussed in this section for the sample question "What is the oldest city in Canada ?". The features are represented same as equation (3.19).

Table 3.6: Example of semantic features

Feature Space	**Features**
headword hypernyms	{(city, 1) (municipality, 1) (urban area, 1) (geographical area, 1) (region, 1) (location, 1)}
related words	{(Rel_be, 1) (Rel_location, 2) (Rel_InOn, 1)}
question category	{(city, 1)}
query expansion	{(city, 1) (municipality, 0.6) (urban area, 0.36) (geographical-area, 0.22) (region, 0.13) (location, 0.08) (physical-object, 0.05) (physical-entity, 0.03) (entity, 0.02) }

3.4.4 Combining Features

The three types of features we described, each takes a different perspective on the question. We explored whether combining different feature sets will improve the classification accuracy. Unlike related work in which the augmented features are blindly added to the feature vector, we suggest a weighted concatenation of the various feature sets:

$$f = (w_1 f_1^{\mathrm{T}}, \ldots, w_m f_m^{\mathrm{T}})^{\mathrm{T}} \tag{3.30}$$

where f_i is the i^{th} feature set, w_i is its weight, m is the number of feature sets that are extracted and f is the final feature set. In total we implemented 12 types of different features, i.e, $m = 12$. If $w_i = 0$ it means that the i^{th} feature set will not be added to the final feature set.

An important question that may rise now is how to learn the optimal values for the weight parameters. To optimize the weight values in equation (3.30), we would need an exhaustive search of all possible weight assignments. As this is time-consuming, we chose a greedy approach instead. For each feature set we searched for the optimal weight when it was combined with the unigram features only. The weight value with highest accuracy will be chosen as the weight parameter of the corresponding feature space. In the next chapter we show what is the best combination of weight values and lists our classification results.

3.4.5 Feature Reduction

Features in question classification are very high dimensional which typically is due to n-grams over vocabularies. We applied a feature reduction technique namely *Latent Semantic Indexing* (LSA) (Deerwester et al., 1990), to see whether it can improve the per-

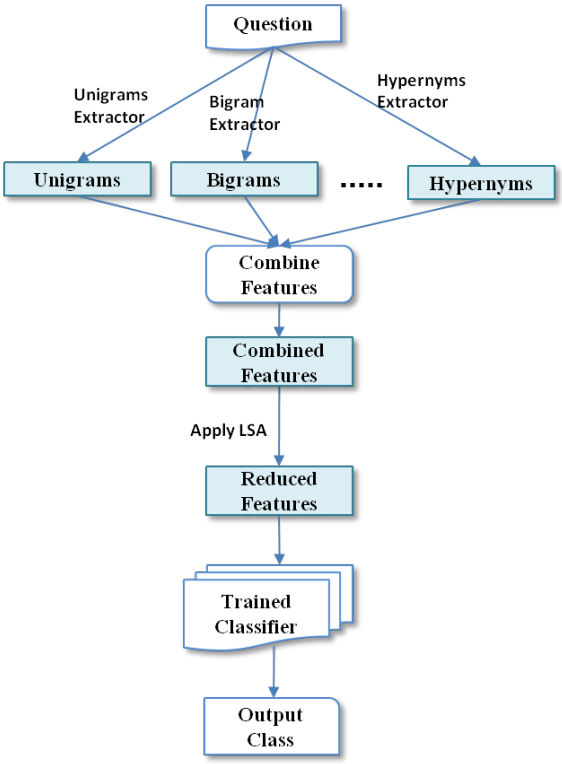

Figure 3.6: The alternative architecture of our question classifier system using LSA feature reduction technique.

formance of our classifier or not. Figure 3.6 illustrates the alternative architecture of our question classifier system when we apply the LSA technique to reduce the feature space.

LSA maps the features space to a reduced space using *singular value decomposition* (SVD). This technique have been widely used in text classification (Yu et al., 2008; Lam and Lee, 1999; Zelikovitz and Hirsh, 2001).

To apply SVD to question classification, we define the feature-by-question matrix $\mathbf{Q}$ in which the rows represent the features and the columns represent questions. That is, if our feature space has d dimensions and the total number of training samples is n, then $\mathbf{Q}$ would be a $d \times n$ matrix in which $\mathbf{Q}_{i,j}$ represents the frequency (weight) of feature f_i in question $\mathbf{x}_j$. SVD decomposes $\mathbf{Q}$ into tree matrices: $\mathbf{Q} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices whose columns are eigenvectors of $\mathbf{Q}\mathbf{Q}^T$ and $\mathbf{Q}^T\mathbf{Q}$ respectively and $\mathbf{\Sigma}$ is a diagonal matrix containing the eigenvalues of $\mathbf{Q}\mathbf{Q}^T$ in the diagonal which are sorted in descending order. To reduce the feature space to k dimensions, we define matrix $\mathbf{U}_k$ to be a $d \times k$ matrix containing the first k column of $\mathbf{U}$. We now defined the reduced matrix as follows:

$$\mathbf{R} = \mathbf{Q}^T\mathbf{U}_k \tag{3.31}$$

where $\mathbf{R}$ is the $n \times k$ reduced matrix, in which each row corresponds to a question which

is described by k features. This technique is very similar to *principle component analysis*. The reduced space is called *latent semantic space* and matrix $\mathbf{U}$ is used to transform a vector to this space.

Once we train our classifiers with the reduced questions, for a given independent question $\mathbf{x}$, it first transforms to the reduced space as follows:

$$\hat{\mathbf{x}} = \mathbf{x}^T \mathbf{U}_k \tag{3.32}$$

where $\hat{\mathbf{x}}$ is a $1 \times k$ vector in the reduced space. This vector then is fed to our classifier, and the output is generated.

Chapter 4

Experimental Results and Analysis

4.1 Introduction

In this chapter we explain our experiment on a well-known dataset step by step. Our goal is to investigate the contribution of different features on question classification. Furthermore, we want to test our weighted approach on combining features to see whether it is a good alternative for combining features or not. We also want to see whether reducing the feature space with the latent semantic analysis method can improve the performance of our classifier or not.

Our step by step experiments start by a brief explanation of the dataset we used. We explain our method to represent features and our implementation details in section 4.2.2. We setup the parameters of our classifiers by testing all possibilities and choosing the best parameters. In section 4.2.3 we explained our tested scenarios to find the optimal classifiers. In section 4.2.4 we investigate the contribution of different lexical, syntactical and semantic features when they are combined together. The experiments to test our weighted approach is discussed in section 4.2.5. In section 4.2.6 we investigate the LSA feature reduction technique for question classification. We finally discuss about the obtained results in section 4.4.

4.2 Experiment

4.2.1 The dataset

The dataset which we used in this work is the one created by Li and Roth (2002). They provided a question dataset which is widely used in question classification studies and known as UIUC or TREC dataset. It consist of 5500 labeled question which is used as training set and 500 independent labeled questions which is used as test set. The datasets are simply text files in which each row consists of a label followed by a question. The following is an example from TREC training set:

HUM:ind Who was The Pride of the Yankees ?
The taxonomy which is used to label questions is the two layer taxonomy which is explained in chapter 2. It consist of 6 coarse grained classes and 50 fine grained classes.

4.2.2 Implementation

As we explained in previous chapter, we represent a question in the sparse form which is described in equation (3.19). The features which are extracted from questions will be added to the feature vector with a (feature, value) pair. If we only extract unigram features, the above equation will be translate to the following form:

$$\{(\text{Who}, 1)(\text{was}, 1)(\text{the}, 2)(\text{Pride}, 1)(\text{of}, 1)(\text{Yankees}, 1)(?, 1)\} \tag{4.1}$$

However, instead of using string, each term (feature) is mapped to a unique number, indicating feature number. Furthermore, the class name also mapped to a unique number. The following form is the same sample from TREC dataset which is translated to the form which is accepted by LIBSVM library (Chang and Lin, 2001):

$$44\ 1{:}1\ 15{:}2\ 24{:}2\ 98{:}1\ 235{:}1\ 1934{:}1\ 4376{:}1 \tag{4.2}$$

where the first number (44) indicates the class number and the rest are (feature, value) pairs which are separated by a white space while the pairs are separated by a colon. Furthermore, the feature pairs should be sorted ascending based on feature number.

When all the samples from training and test set are translated to the above format, then we train our classifiers with training set and test it against the independent test set.

4.2.3 Classifiers Parameters Setup

We used two different classifiers in this work: Support Vector Machines (SVMs) and Back-Propagation Neural Networks (BPNN). To obtain the best classifier, the parameters and structure of them should be configured to its optimal values.

Support Vector Classifier

We tested our SVM classifier with 4 types of kernel functions: linear, polynomial, sigmoid and radial basis function among which linear was best. Table 4.1 lists the definition of each kernel for two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$:

Table 4.1: Mathematical definition of four basic kernel functions

Kernel	**Definition**
Linear	$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i.\mathbf{x}_j$
Polynomial (degree d)	$k(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i.\mathbf{x}_j + C_0)^d$
Radial Basis	$k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \lvert\mathbf{x}_i - \mathbf{x}_j\rvert)^2$
Sigmoid	$k(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i.\mathbf{x}_j + C_0)$

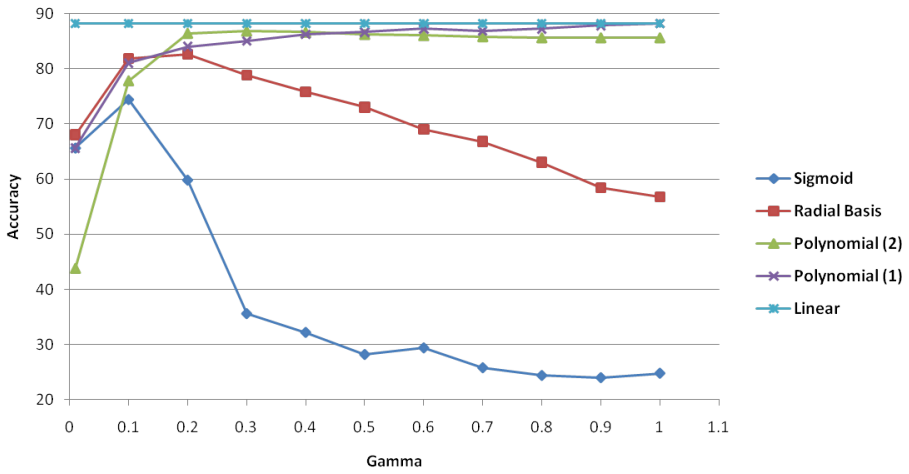

Figure 4.1: The accuracy of the SVM classifier on the coarse grained classes with Unigram features based on different values of γ and different kernels.

In the above definition γ and C_0 are the coefficient and the constant value which defines the kernel functions. To obtain the best values for these two parameters we tested the SVM classifier with different values of γ and C_0. If C_0 is considered as 0 the resulting function is called *homogeneous* kernel and otherwise it is $non - homogeneous$. In our case both homogeneous and non-homogeneous kernels obtain same results. Figure 4.1 illustrates the accuracy of linear, polynomial, sigmoid and radial basis kernels based on different values of γ for the coarse grained classes when unigram features are used. However compare to linear kernel function, the best accuracy of these 3 kernels still is smaller than the simple linear kernel. Table 4.2 list the best accuracies obtained by each kernel type for the coarse grained classes. Since in all cases linear kernel performs better, we use this kernel as the final choice of kernel function.

Table 4.2: The accuracy of SVM classifier based on different kernels for the coarse grained classes.

Kernel	Linear	Polynomial(1)	Polynomial(2)	RBF	Sigmoid
Accuracy	88.2%	88.2%	86.8%	82.6%	72.4%

Another parameter of SVM classifier that can influence on classification accuracy is the penalty parameter C (see equation 3.9). We test our SVM classifier based on different values of C. Figure 4.2 illustrates the influence of penalty parameter on classification accuracy. This experiment is done with linear kernel function and based on unigram features for coarse grained classes.

Figure 4.2 reveals that choosing parameter C to be equal to 1 obtains the best performance.

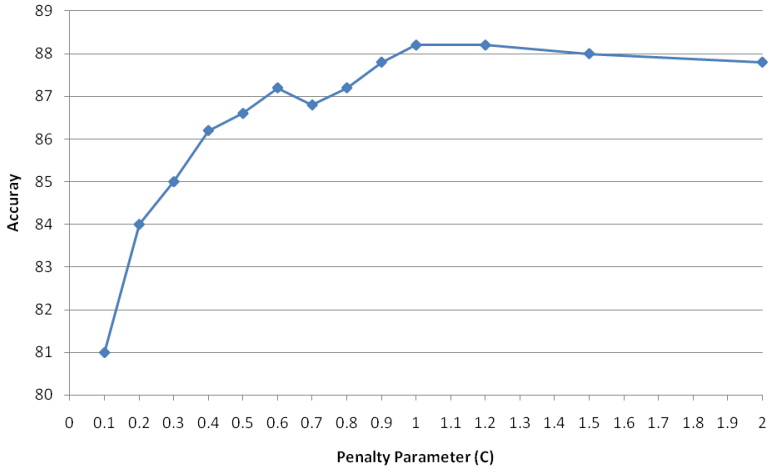

Figure 4.2: The accuracy of the SVM classifier on the coarse grained classes with Unigram features based on different values of penalty parameter (C).

Back-Propagation Neural Network

The second classifier we used in this work is BPNN classifier. To find the optimal BPNN classifier, number of hidden layers, number of units in each layer, activation function, maximum number of iterations and learning rate should be specified.

As we described in the previous chapter, in a BPNN classifier, if our samples have d dimensions and the number of classes is m the number of input neurons and output neurons should be set to d and m, respectively. Our BPNN classifier uses one hidden layer in which the number of hidden units are set to the number of classes (see figure 3.2). The reasons of choosing this architecture are both accuracy and efficiency. In the tested scenarios, having more than 1 hidden layers do not necessarily improves the performance while the network takes more time to be trained. The maximum number of iterations in the gradient descend method is set to 500 and the learning rate is set to 0.7 since with this combination of parameters in most cases the error converges to a fixed value.

For the choice of activation function we tested 4 different type of activation functions: sigmoid, step, Gaussian and tanh among which sigmoid performs the best. Table 4.3 lists the definition of the activation functions we used in this work.

We tested our BPNN classifier in a particular feature space with the above 4 kernels on the coarse grained classes. Table 4.4 lists the best accuracy obtained by each activation function.

As the above table shows, step function performs the worst since it is a discrete function and therefore a neuron is either completely activated or completely deactivated. Gaussian kernel also shows bad performance since its output is in the rang $[0, \infty]$ which can not monotonically reflect neuron's activations. On the other hand sigmoid and tanh functions have good results since they are both concrete functions and their output is in the range

Table 4.3: Mathematical definition of four basic activation functions

Kernel	**Definition**
Sigmoid	$f(\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{x})}$
Step	$f(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} > 0 \\ 0 & \mathbf{x} \leq 0 \end{cases}$
Gaussian	$f(\mathbf{x}) = \exp(-\frac{\mathbf{x}^2}{2\sigma^2})$
Tanh	$f(\mathbf{x}) = \frac{e^{2\mathbf{x}-1}}{e^{2\mathbf{x}+1}}$

Table 4.4: The accuracy of BPNN classifier on the coarse grained classes based on different activation functions.

Kernel	Sigmoid	Step	Gaussian	Tanh
Accuracy	85.2%	18.8%	27.6%	83.2%

[0,1]. (Note that we mapped the output rang of tanh function from [-1,1] to [0,1] by adding the output with 1 and then dividing the resulting value by 2. This leads our results to be much better.) So in the rest of our experiments we used the sigmoid activation function. Table 4.5 summarized the parameters that we choose for our classifiers.

Table 4.5: Summary of the configuration of our classifiers

	Parameter	**Value**
SVM	Kernel	Linear
	Penalty Parameter (C)	1
BPNN	Activation Function	Sigmoid
	Structure	$d - m - m$
	Learning Rate	0.7
	Maximum Iterations	500

4.2.4 Incremental Features Combination

We did our experiments in two different scenarios: either to train the classifiers in the original space or applying LSA feature reduction technique and train the classifiers in the reduced space. For the first scenario we only used our SVM classifier since training the BPNN classifier in the original space demands a very large network which take much time to train. But in the second scenario we used both SVM classifier and BPNN classifier.

We totally extracted 12 different feature sets. We created different feature space by combining these feature sets. Combining all features sets is not necessarily the best option. We incrementally combined them to obtain the best combination of features. Table 4.6 lists the feature sets we extracted in this work together with their type, abbreviation and

Table 4.6: All lexical, syntactical and semantic features we used in this work.

no.	Feature Set	Abbreviation	Type	#Dimensions
1	Unigrams	U	Lexical	9775
2	Bigrams	B	Lexical	30721
3	Limited-Bigrams	LB	Lexical	1010
4	Word-Shapes	WS	Lexical	5
5	Wh-words	WH	Lexical	8
6	Headwords	H	Syntactical	1964
7	Head-Rules	HR	Syntactical	37
8	Tagged-Unigram	TU	Syntactical	10391
9	Hypernyms	HY	Semantic	5774
10	Query Expansion	QE	Semantic	5791
11	Question Category	QC	Semantic	1967
12	Related-Words	R	Semantic	78

Table 4.7: The accuracy of SVM classifier on TREC dataset based on different combination of lexical features.

no.	Features	Dimensions	Accuracy	
			Coarse	Fine
1	U	9775	88.2	80.2
2	U+WS	9780	88.8	80.6
3	U+WH	9783	88.2	80.4
5	B	30721	86.8	75.2
6	U+WS+B	40501	91.2	81.0
7	U+WS+LB	10790	91.2	82.2

dimensionality.

Contribution of Lexical Features

In the first set of our experiments with SVM classifier, we investigate the role of lexical features on classification accuracy. While we are seeking for the highest accuracy, we are also interested in the feature spaces with lower dimensions since they are more time efficient. Table 4.7 lists the accuracy of SVM classifier on different combination of lexical features.

An interesting result from table 4.7 is that the limited-bigram feature set perform as accurate as the bigram feature space although its dimension is much smaller than the dimension of bigrams. This reveals that the main contribution of bigram feature set is due to the first bigram of a question, i.e., the wh-word and the word next to it.

Table 4.8: The accuracy of SVM classifier on TREC dataset based on different combination of lexical and syntactical features.

no.	**Features**	**Dimensions**	**Accuracy**	
			Coarse	Fine
1	TU	10391	87.4	80.6
2	TU+WS+LB	11406	91.4	81.8
3	U+WS+LB+H	10790	91.0	83.8
4	U+WS+LB+HR	10797	90.8	81.6
5	U+WS+LB+H+HR	10797	91.4	84.8
6	WH+WS+H	1977	88.6	77.0
7	WH+WS+H+HR	1984	88.0	77.4
8	WH+WS+LB+H	2987	87.6	77.2
9	WH+WS+B+H	32698	90.8	81.0

Adding Syntactical Features

In the next set of experiments we added syntactical features to the lexical feature sets. Table 4.8 lists the accuracy as well as the dimensionality of different combinations of lexical and syntactical feature sets.

In the second half of table 4.8 we used wh-word feature set instead of unigrams. An interesting result from this table is that the combination of wh-word and headword performs almost as good as unigrams while its dimensionality is much smaller. This result shows that the wh-word and the headword of a question are two most informative features of that question.

Another interesting result from the above table is that our limited bigram feature set does not improve classification accuracy when it is combined with low-dimensional feature sets such as wh-words and word-shapes (row 8 of table 4.8). This is most likely due to the high dimensionality of the limited-bigram feature space compare to wh-word and word-shapes. The interesting point is that when we use bigram instead of limited-bigram the accuracy is improved (row 9 of table 4.8). These two results comes to the conclusion that limited-bigram feature set is only useful when it combined with high-dimensional feature spaces.

Adding Semantic Features

To obtain the best combination of feature sets, we added semantic features to some candidate feature sets. In the previous experiments the combination of unigrams, word-shapes, limited-bigrams, headwords and head-rules obtains the best performance and the combination of wh-words, word-shapes and headwords obtains a good accuracy compare to other combinations while its number of dimensions is relatively low. We choose these two combinations to be augmented by semantic features. Table 4.9 lists the accuracy and dimensionality of different combination of lexical, syntactical and semantic features.

Table 4.9: The accuracy of SVM classifier on TREC dataset based on different combination of lexical and syntactical and semantic features.

no.	**Features**	**Dimensions**	**Accuracy**	
			Coarse	Fine
1	U+WS+LB+H+HR+HY	14607	92.0	85.6
2	U+WS+LB+H+HR+QE	14614	92.0	86.4
3	U+WS+LB+H+HR+QC	10799	91.2	85.6
4	U+WS+LB+H+HR+R	10875	92.6	89.4
5	U+WS+LB+H+HR+QE+R	14692	92.8	89.4
6	U+WS+LB+H+HR+QE+R+QC	14694	93.0	90.0
7	U+WS+B+H+HR+QE+R+QC	44405	**94.2**	**90.4**
8	WH+WS+H+R	2055	89.6	87.8
9	WH+WS+LB+H+R	3065	91.4	88.2
10	WH+WS+H+QE+R	5872	90.2	88.0
11	WH+WS+H+QE+R+QC	5875	90.6	87.8
12	WH+WS+H+HR+QE+R	5879	90.8	88.0
13	WH+WS+LB+H+QE+R	6882	92.4	89.0
14	WH+WS+LB+H+HR+QE+R	6889	93.0	89.0

The results from table 4.9 shows that semantic features can significantly improve classification accuracies. It is also worth mentioning that our query-expansion feature set performs better than hypernyms (rows 1 and 2). So in the rest of our experiments we used query-expansion feature set instead of hypernyms.

As it reveals from table 4.9, the combination of unigrams, bigrams, word-shapes, headwords, head-rules, query-expansion, related-words and question-category with total feature space of size 44405 dimensions (row 7) obtains the best classification accuracy. Furthermore, the combination of wh-word, word-shapes, limited-bigram, headwords, head-rules, query-expansion and related-words (row 14) obtains completive accuracies while the size of feature space is relatively small. This result is very close to the state-of-the-art work in this task (Silva et al., 2011) although our feature space is much smaller.

4.2.5 Weighted Combination of Features

In the previous sub-section we obtained the best combination of features when we combined different feature sets with equal weights, i.e. all the feature sets have equal contribution to the final classification task.

The idea of weighted combination of feature sets is to consider a biased contribution for different feature sets. We implemented this bias by a weight parameter which reflects the importance of feature set. Back to section 3.4, a question is represented by a vector space model in which the values are a function of word frequencies. The larger a word frequency (feature weight) is, the higher influence it has on the final classification.

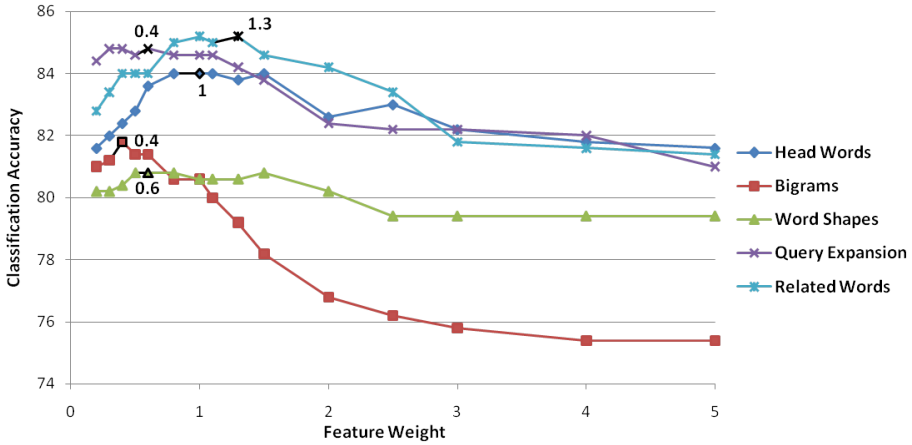

Figure 4.3: The classification accuracy of unigrams combined with different feature sets based on weight values.

To implement our weighted approach, consider that a question $\mathbf{x}$ is represented in feature space A and B with equations (4.3) and (4.4) respectively:

$$\mathbf{x} = \{(t_1, f_1) \cdots (t_r, f_r)\} \tag{4.3}$$

$$\mathbf{x} = \{(t'_1, f'_1) \cdots (t'_s, f'_s)\} \tag{4.4}$$

where r and s are number of non-zero features of $\mathbf{x}$ in feature spaces A and B, respectively. If we combine these two feature spaces with weight values of w_A and w_B respectively, then the combined features are represented as follows:

$$\mathbf{x} = \{(t_1, w_A f_1) \cdots (t_r, w_A f_r)(t'_1, w_B f'_1) \cdots (t'_s, w_B f'_s)\} \tag{4.5}$$

If there is a same feature t_i in both feature spaces, then their corresponding values will be sum up:

$$\mathbf{x} = \{(t_i, w_A f_i + w_B f'_i)\} \tag{4.6}$$

To decide on the weight values we need to do an exhaustive search to test all possible combination of weight values and choose the one with highest accuracy. Since it is a time-consuming procedure, we instead used a greedy approach in which each feature set is combined with unigrams based on different weight values. The weight value with highest classification accuracy is chosen as the final weight for the corresponding feature set. Figure 4.3 illustrates the classification accuracy of our SVM classifier when a combination of unigrams and another feature set is used based on different wight values. Weight values which lead to highest accuracy are chosen as the final weight parameters.

Table 4.10 lists the best combination of weight values which are obtained base on our greedy approach for the candidate feature sets. Using the resulting weight values,

we again tested our classifier with combination of unigrams, bigrams, word-shapes, head-words, head-rules, query-expansion, related-words and question-category. The accuracies improved. We reached an accuracy of 91.0% on the fine grained classes and 94.8% on the coarse grained classes which is higher that the state-of-the-art accuracies on this task (Silva et al., 2011).

Table 4.10: The best combination of weights obtained by our greedy approach.

Feature Set	U	B	WS	H	QE	QC	R
Weight	1	0.4	0.6	1	0.4	1	1.3

4.2.6 Comparison in the Reduced Space

The next set of our experiment is to test our classifiers in the reduced feature space. We first want to investigate the behavior of different feature sets in the reduced space and then to find out what is the best size for the reduced space. We tested the accuracy of different feature sets on the reduced space with both SVM and BPNN classifiers. To do so we choose two candidate combinations of features one with relatively high dimensionality and one with relatively low dimensionality, both having high accuracy in the original feature space with our support vector classifier. Table 4.11 lists these two candidate combinations. The first combination uses unigrams while the second uses wh-words instead of it. The reasons that we choose these two candidates are both accuracy and efficiency.

Table 4.11: Two proper combination of feature sets.

no.	**Features**	**#Dimensions**	**Coarse**	**Fine**
1	U+WS+LB+H+HR+QE+QC+R	14694	93.0	90.0
2	WH+WS+LB+H+R	3065	91.4	88.2

The next step is to reduce the dimensionality of these two candidate sets and see the classification accuracy in the reduced space. We applied the latent semantic analysis method to reduce the dimensionality of feature spaces. To obtain the optimal dimensionality for the reduced feature space we tested our classifiers based on different dimensionalities in the reduced space. Figure 4.4 compares the accuracy of SVM and BPNN classifiers on the candidate feature sets in table 4.11 for the coarse grained classes and figure 4.5 compares the accuracy of second candidate on the fine grained classes in the reduced space. The horizontal axes in the figures are the number of features that results from the LSA reduction and the vertical axis are classification accuracies.

As the figures reveals, BPNN performs better in the reduced space for coarse grained classes while for the fine grained classes SVM performs better in the reduced space. Compare to the original space, SVM has lower accuracy in the reduced space while BPNN performs better than SVM in the original space. The most interesting result from figure 4.4, is that feature set 2 has higher accuracy than feature set 1 in the reduced space even

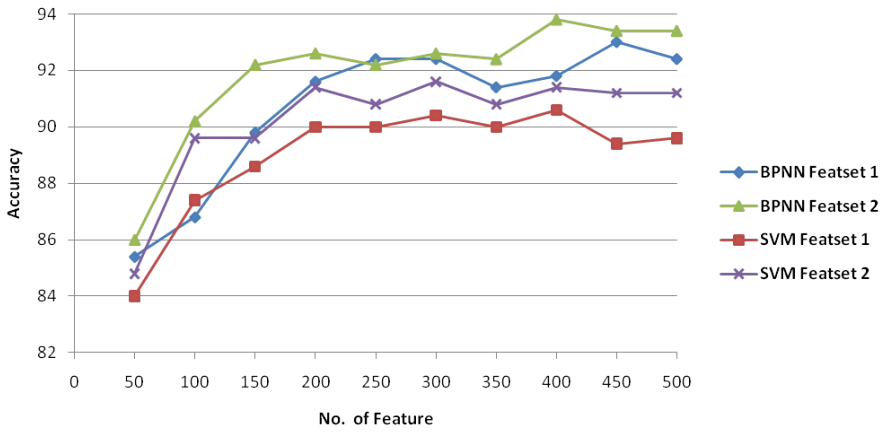

Figure 4.4: Comparison of SVM and BPNN classifiers in the reduced space based on the coarse grained classes.

Table 4.12: The accuracy of SVM and BPNN classifier on the 400 dimensional reduced space for the coarse grained classes compare to the accuracy of SVM in the original space.

Features	**Original Space**	**Reduced Space**	
	SVM	**SVM**	**BPNN**
WH+WS+H	88.6	85.6	88.8
WH+WS+H+R	89.6	88.4	90.2
WH+WS+LB+H+R	91.4	90.4	**93.8**

though in the original space feature set 1 has a higher accuracy. The reason may be that feature set 2 has lower dimensions and describes the samples in a more compact space and therefore looses less information when it is reduced to a lower dimensional space.

The best accuracy which is obtained for coarse grained classes in the reduced space is 93.8% with 400 features using BPNN classifier. This result is not only better than the accuracy of SVM in the original space (93.0%), but also uses only 400 features which is much less than the dimensionality of the original space which is 14696. Table 4.12 compares accuracies of more feature sets with SVM and BPNN classifiers in the 400 dimensional reduced space for the coarse grained classes. As this table reveals, in all cases BPNN in the reduced space has a higher accuracy than the SVM classifier in the original and the reduced space.

Our results shows that feature reduction can improve accuracy in the question classification problem under some settings. While it can improve classification accuracy for the coarse grained classes, it is not useful for classifying fine grained classes. This result leads to a conclusion that when the number of classes is relatively large, more features are needed to increase the discrimination power of classifier and therefore feature reduction is

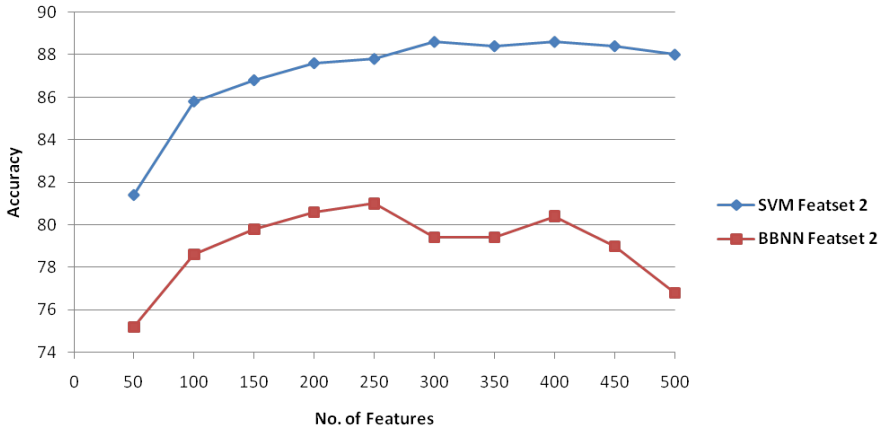

Figure 4.5: Comparison of SVM and BPNN classifiers in the reduced space based on the fine grained classes.

not useful. On the other hand when the number of classes is smaller, LSA feature reduction technique can improve classification accuracy and this is most likely due to removing redundant features.

4.3 Stability of the Results

In previous sections we did several experiments on the TREC dataset based on a fixed training set and an independent test set. A question that can be raised is that how we can make sure that we have not been trapped in the problem of *overtraining*. In fact one may ask that the features that we extracted and used to evaluate the performance of our system can be very fitted to our training and test set and it is possible that our features may have not such a good performance on other training and test sets.

To investigate the stability of our results we used a well-know pattern recognition technique, known as *cross validation*. We applied this technique on TREC dataset. The total amount of samples (training and test samples) in this dataset is about 6000 questions of which 500 are chosen as an independent test set. Instead of testing the performance of our system on the independent test set, we divided our dataset to 12 random sets of 500 questions to experiment a 12-folds cross validation on this dataset. The reason that we choose 12 folds is have the same size for cross validation test set and the independent test set. We did 12 experiments in which one of the random sets are considered as a test set and the rest is considered as the training set. Table 4.13 lists the mean and variance of the 12-fold cross validation using our SVM classifier based on two different combination of features. The first feature space is unigarm which is relatively large feature space and the second is a combination of low dimensional features.

As the table shows, expect for the coarse grain classes on the unigram feature space,

Table 4.13: 12-fold cross validation on TREC dataset

	Unigrmas		**WH+WS+H+R**	
	Coarse	Fine	Coarse	Fine
Accuracy (Test set)	88.2	80.2	89.6	87.8
Mean (12 Fold)	85.1	78.8	89.8	86.6
Var (12 Fold)	2.4	3.1	1.72	2.2

the accuracy of the independent test set falls into the cross validation accuracy interval. Nevertheless, the differences between the independent test set accuracy and the cross validation accuracy are insignificant which actually indicates that our results do not vary very much on other train and test sets. However, the variance on the unigram feature space is relatively higher than the variance on the other low dimensional feature space. This result suggests that the results on the unigram feature space is more unstable compare to the other feature space. The reason may lie on the fact that the unigram feature space has higher dimensions compare to the other feature space.

4.4 Analysis of Results

Question classification is a hard problem. As you saw in the previous sections, a series of complicated tasks should be done to extract good set of features, combine them and possibly reduce them to a lower dimension space. Furthermore, different type of classifiers with different parameters can be used for this task.

In previous sections we tried to find the best configuration for our classifier and the best solution for the question classification problem. This was done by covering the following 6 challenges:

1. Choosing a proper classifier for QC problem
2. Finding the best parameters for the classifier(s)
3. Extracting good set of features from lexical, syntactical and semantic structure of questions
4. Improving the algorithms of extracting features
5. Combining different feature sets in an optimal way
6. Possibly reducing the feature space to a more efficient and effective space

After finishing all the above 6 challenges, the next step is to analyze the obtained results more in dept to see *why* our results was like that. We did this by a deep analysis of the errors and a detail exploration of our dataset.

Metzler and Croft (2005) explored TREC dataset and discovered 4 issues which cause misclassification in the QC problem. These issues are:

1. **Inconsistent and ambiguous labeled data**: some of the samples in TREC dataset have ambiguous label. For example the question "What does CNN stands for ?" is labeled with "ABBR:exp" while it can also be labeled by "HUM:org".

2. **Inherently difficult questions**: Some questions are even difficult for human to correctly classify them. For example consider the question "What is the name of the Lion King's son in the movie the Lion King ?". Classifying this question to type "animal" is even difficult for human.

3. **POS tagger and WordNet expansion error**: The POS taggers, parsers and WordNet expanders are not infallible and it happens that they have errors or introduce noisy information. For example consider the question "What U.S. Government agency registers trademarks ?". The POS tagger may tag this question as follow:

   "What_WP U.S._NNP Government_NN agency_NN registers_NNS trademarks_NNS ?_."

   which incorrectly tags the word "registers" as plural noun. Consequently, the headword will be misidentified to "trademarks" instead of "agency" and the incorrect headword will be expanded by WordNet which can lead to misclassification.

4. **WordNet insufficiencies**: Although expansion of the questions headword with WordNet can improve classification accuracy, it always introduces a certain amount of noise to the feature vector. Sometimes this noise can cause the question to be misclassified. For example the question "What do bats eat ?" can be correctly classified to "ENTY:food" using unigram features but when it's headword, "bats", is expanded via WordNet, it introduce noisy information which leads the question to be misclassified to "ENTY:sport".

The above misclassification causes reveal that most of the errors are due to difficulties in understanding the question. We found that some types of questions are more difficult to be classified compare to the others. Based on the confusion matrix of the tested samples, we found that classifying samples of type "ENTY" and "LOC" in TREC dataset, is more difficult compare to other 4 categories of coarse grained classes. Tables 4.14 and 4.15 show the confusion matrix of the TREC dataset for coarse grained classes based on SVM and BPNN classifiers, and table 4.16 lists the precision and recall of the coarse grained classes.

Huang et al. (2008) performed a detailed analysis on TREC dataset and discovered that "what" type questions are more difficult to classify compare to other type of questions. The reasons mainly back to the *ambiguity* on classifying "what" type questions which is not the case for other type of questions. For example the question "What is mad cow disease ?" can be both classified to "ENTY:disease" and "DESC:def'. Huang et al. (2008)

Table 4.14: Confusion matrix showing the classifications of the TREC dataset for the coarse categories based on SVM classifier

		Predicted labels					
		ABBR:*	DESC:*	ENTY:*	HUM:*	LOC:*	NUM:*
True	ABBR:*	9					
	DESC:*		134	2		1	1
	ENTY:*		10	83	1		
	HUM:*		1	1	63		
	LOC:*		1	9		71	
	NUM:*		3	2			108

Table 4.15: Confusion matrix showing the classifications of the TREC dataset for the coarse categories based on BPNN classifier

		Predicted labels					
		ABBR:*	DESC:*	ENTY:*	HUM:*	LOC:*	NUM:*
True	ABBR:*	6	3				
	DESC:*		130	4	1	1	2
	ENTY:*		6	84	3		1
	HUM:*		1	3	61		
	LOC:*	1	1	10	2	67	
	NUM:*		1	4			107

also listed inconsistent labeling and parse errors as two other reasons of misclassifying "what" type questions. Table 4.17 lists the classification accuracy on TREC test set based on question types and two different classifiers. As it can be seen in this table, most of the questions are of the type "what" while they are the most difficult questions to be correctly classified.

4.5 Summary

In this chapter we explained the detailed steps of our experiments. We first tuned the parameters of our classifiers and then extracted different features and combined them with different approaches.

We found that extracting features from syntax and semantic of questions can improve the classification accuracy by adding more information to the feature vectors.

We found that out weighted method for combining features can improve the classification accuracy. Furthermore, we found that using neural network classifier, the LSA feature reduction technique can improve the classification accuracy.

By a detailed exploration of our dataset, we found that the questions which are of type "ENTY" are harder to classify compare to the other type of questions. This is most likely due to lack of samples in this class. We expect that by increasing the size of training data, the classification accuracy of these type of questions also increases. Furthermore,

Table 4.16: Precision and Recall of coarse grained classes of TREC dataset based on SVM and BPNN classifiers

		ABBR	DESC	ENTY	HUM	LOC	NUM
SVM	Precision	100%	89.9%	85.6%	98.4%	98.6%	99.1%
	Recall	100%	97.1%	88.3%	96.9%	87.6%	95.6%
BPNN	Precision	85.7%	91.5%	80.0%	91.0%	98.5%	97.2%
	Recall	66.6%	94.2%	89.3%	93.8%	82.7%	95.6%

Table 4.17: Classification accuracy of SVM and ME classifiers based on question types. The results are taken from Huang et al. (2008)

Question type	**#Questions**	**Coarse**		**Fine**	
		SVM	ME	SVM	ME
what	349	90.5%	91.1%	86.2%	86%
which	11	100%	100%	90.9%	100%
when	26	100%	100%	100%	100%
where	27	100%	100%	92.6%	92.6%
who	47	100%	100%	100%	100%
how	34	100%	100%	97.1%	91.2%
why	4	100%	100%	100%	100%
rest	2	100%	50.0%	0.0%	50.0%

classifying the *what* type questions are more difficult than other type of questions. The reason lie on the fact that the "what" wh-word is less informative than other wh-questions. In other words, a broader type of questions are start by "what", compare to other wh-words such as "when" and "why".

Chapter 5

Related Work

5.1 Introduction

Question classification problem has already been studied in many previous works. In this chapter we review some previous works on question classification together with their results. We first overview the supervised learning methods which have been used in question classification in section 5.2. In section 5.3 we mention some other features which are used in other studies, in addition to the features that we used. We then compare different supervised learning studies with our work in section 5.4. In section 5.5 we review some semi-supervised approaches on question classification.

5.2 Supervised Learning Approaches in Question Classification

Most of the recent works on question classification are based on a supervised learning method. Supervised learning approaches learn a classifier from a given training set consisting of labeled questions. Supervised methods mainly differ in the classification model and the features which are extracted from questions.

The choice of classifier highly influences the final question classifier system. Different studies choose different classifiers. Support Vector Machines (SVM), Maximum Entropy Models and Sparse Network of Winnows (SNoW) are the most widely used classifiers in question classification. Some studies used language modeling for question classification. A few studies adopted other types of classifiers. In this section we categorized different studies based on the classifiers they used and briefly describe each classifier in different subsections.

5.2.1 Support Vector Machines

Support vector machines are non-probabilistic learning models for classifying data. They are especially successful for high dimensional data. SVM is a linear discriminant model

which tries to find a hyperplane with maximum margin for separating the classes. The detailed explanation of SVMs can be found on chapter 3.

5.2.2 Advanced Kernel Methods

Some studies adopt SVMs with customized kernel function. Zhang and Lee (2003) defined a tree kernel which is constructed based on the syntactical structure of question. In their approach, a given question first is parsed to its syntactic tree and then the question will be represented based on some tree fragments which are subtrees of the original syntax tree. They define a custom kernel function which maps the feature vector to a higher dimension space. In section 4.2 we will further discuss the syntactical structure of a question.

A similar approach is used to define kernel function in the study of Pan et al. (2008). They defined a semantic tree kernel which is obtained by measuring the semantic similarities of tree fragments using semantic features. They reported an accuracy of 94.0% on coarse-grained classes while Zhang and Lee (2003) obtained an accuracy of 90.0% on the same dataset.

Kernel methods have also been applied in semi-supervised style. Tomas and Giuliano (2009) defined a semantic kernel for question classification which is obtained by using unlabeled text. They used latent semantic indexing method (Deerwester et al., 1990) to reduce the feature space to much more effective space by defining a latent semantic kernel.

In their experiment, Tomas and Giuliano (2009) reduced the feature space to 400 dimensions. They also defined a semantic kernel function based on a manually constructed list of related words. The semantic related kernel K_{Rel} is defined as follow:

$$K_{Rel}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \mathbf{P}\mathbf{P}^T \mathbf{x}_j^T = \acute{\mathbf{x}}_i \acute{\mathbf{x}}_j^T \tag{5.1}$$

where $\mathbf{P}$ is proximity matrix which reflects the similarity between the words in the list. Tomas and Giuliano (2009) do their experiment on TREC dataset by applying different kernels on the input feature space. Table 5.1 lists the accuracy of their experiment on TREC dataset with different combination of kernels (K_{LS} is the latent semantic and K_{BOW} is the bag-of-words kernel). The best result is obtained by combination of all three kernels.

Table 5.1: The accuracy of kernel methods on TREC dataset based on different kernel functions. The results are taken from Tomas and Giuliano (2009)

Kernel	**Accuracy**	
	Coarse	Fine
K_{BOW}	86.4%	80.8%
K_{LS}	70.4%	71.2%
$K_{\text{BOW}} + K_{LS}$	90.0%	83.2%
$K_{\text{BOW}} + K_{Rel}$	89.4%	84.0%
$K_{\text{BOW}} + K_{LS} + K_{Rel}$	90.8%	85.6%

5.2.3 Maximum Entropy Models

Maximum Entropy (ME) models which are also known as Log Linear models is another successful classifier used in question classification. In contrast to SVMs, maximum-entropy model is an statistical approach which can calculate the probability of belonging to each class for a given sample. Additionally, ME models can be used for multiple class assignment strategy (see equation 2.1) while SVMs can only be used for single class assignment. Furthermore the uncertainty of the assigned label can be used later to rank the final answer.

ME models are very useful when there are many overlapping features, i.e., when the features are highly correlated. In the case of question classification as you will see in the next section, it often happens that features are very dependent.

In ME model the probability that sample $\mathbf{x}_i$ belongs to class y_j is calculated as following (Berger et al., 1996):

$$p(y_j|\mathbf{x}_i, \lambda) = \frac{1}{Z(\mathbf{x}_i|\lambda)} \exp \sum_{k=1}^{n} \lambda_k f_k(\mathbf{x}_i, y_j) \tag{5.2}$$

where f_k is feature indicator function which is usually binary-valued function defined for each feature; λ_k is weight parameter which specifies the importance of $f_k(\mathbf{x}_i, y_j)$ in prediction and $Z(\mathbf{x}_i|\lambda)$ is a normalization function which is determined by the requirement $\sum_j p(y_j|\mathbf{x}_i, \lambda) = 1$ for all $\mathbf{x}_i$:

$$Z(\mathbf{x}_i|\lambda) = \sum_j \exp \sum_{k=1}^{n} \lambda_k f_k(\mathbf{x}_i, y_j) \tag{5.3}$$

Typically, in question classification f_k is a binary function of questions and labels and defined by conjunction of class label and predicate features (Blunsom et al., 2006). The following equation is a sample of feature indicator function in question classification:

$$f_k(\mathbf{x}, y) = \begin{cases} 1 & \text{if word who in } \mathbf{x} \ \&\ y\text{=HUM:individual} \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

To learn the parameters of the model (λ), ME tries to maximize the log-likelihood of the training samples:

$$\mathcal{LL} = \sum_i \log \frac{\exp \sum_{k=1}^{N} \lambda_k f(\mathbf{x}_i, y_i)}{\sum_j \exp \sum_{k=1}^{N} \lambda_k f(\mathbf{x}_i, y_j)} \tag{5.5}$$

where N is number of features, $\mathbf{x}_i$ is the i^{th} training sample, y_i is its label respectively. To avoid overfitting in ME model, usually a prior distribution of the model parameters is also added to the above equation. Blunsom et al. (2006) defined a Gaussian prior in their model:

$$p(\lambda_k) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{\lambda_k^2}{2\sigma^2}) \tag{5.6}$$

By considering the Gaussian prior, the log-likelihood objective function will be as follows:

$$\mathcal{LL} = \sum_{i=1}^{n} \log \frac{\exp \sum_{k=1}^{N} \lambda_k f_k(\mathbf{x}_i, y_i)}{\sum_j \exp \sum_{k=1}^{N} \lambda_k f_k(\mathbf{x}_i, y_j)} + \sum_{k=1}^{n} \log p(\lambda_k) \tag{5.7}$$

The optimal parameters of the model (λ) will be obtained by maximizing the above equation.

Several studies adopted ME model in their work. Kocik (2004) did his experiment on TREC dataset and obtained accuracy of 85.4% on fine and 89.8% on coarse-grained classes. By extracting better features, Blunsom et al. (2006) reached an accuracy of 86.6% on fine-grained and 92.0% on coarse-grained classes on same dataset. In more recent work, Huang et al. (2008) yet obtained better results due to better feature extraction techniques. They reached an accuracy of 89.0% on fine and 93.6% on coarse-grained classes on the same dataset.

Le Nguyen et al. (2007) proposed a sub-tree mining approach for question classification. In their approach a question is parsed and the subtrees of the parsed tree are considered as features. They used ME model for classification and reported an accuracy of 83.6% on the fine-grained classes of TREC dataset. They used more compact feature space compare to other works. With same feature space their result outperforms the SVM with tree kernel (Zhang and Lee, 2003).

5.2.4 Sparse Network of Winnows

Sparse Network of Winnows (SNoW) is a multi-class learning architecture which is specially useful for learning in high dimensional space (Roth, 1998). It learns separate linear function for each class. The linear functions are learned by an update rule. Several update rules such as naive Bayes, Perceptron and Winnow (Littlestone, 1988) can be used to learn the linear functions.

Li and Roth (2002, 2004) used SNoW architecture to learn a question classifier. They introduced a hierarchical classifier which first assigns a coarse label to a question and then uses the assigned label together with other features, as input features for the next level classifier.

Similar to ME model, SNoW can assign density values (probabilities) to each class for a given sample and therefore make it possible to assign multiple labels to a given sample (equation 2.1). Li and Roth (2002, 2004) used the multiple class assignment strategy in their model. They used same model in both studies but in the latter they extracted reacher semantic features. They obtained an accuracy of 89.3% on fine-grained classes of TREC dataset in the latter work. They also reported an accuracy of 95.0% on fine and 98.0% on coarse-grained classes when multiple labels can be assigned to a question according to decision model in equation 2.1.

5.2.5 Language Modeling

The basic idea of language modeling is that every piece of text can be viewed as being generated by a language. Language modeling has been widely used for document classification (Ponte and Croft, 1998; Jurafsky and Martin, 2008). The idea is that a document D is viewed as a sequence $w_1, ..., w_N$ of words and the probability of generating this sequence is calculated for each class. The class label is determined using the Bayes rule.

Same idea have been used for question classification (Li, 1999; Murdock and Croft, 2002; Merkel and Klakow, 2007). A question $\mathbf{x}$ can be viewed as a sequence $w_1, ..., w_m$ of words such that w_i is the i^{th} word in the question. In fact a question can be viewed as a *mini-document*. The probability of generating question $\mathbf{x}$ by a language model given class c, can be calculated as follow:

$$p(\mathbf{x}|c) = p(w_1|c)p(w_2|c, w_1)...p(w_n|c, w_1, ..., w_{m-1}) \tag{5.8}$$

such that $p(w_i|c, w_1, ..., w_{i-1})$ is the probability that the word w_i appears after the sequence of $w_1, ..., w_{i-1}$ given class c. Since learning all these probabilities needs a huge amount of data usually a *unigram* assumption is made to calculate the probabilities, i.e., the probability of appearing w_i in a question only depends on the immediate words before w_i. Applying this assumption to (5.8) will lead to the following simpler form:

$$p(\mathbf{x}|c) = \prod_{i=1}^{m} p(w_i|c, w_{i-1}) \tag{5.9}$$

The most probable label is determined by applying the Bayes rule:

$$\hat{c} = \arg\max_{c} p(\mathbf{x}|c)p(c) \tag{5.10}$$

where $p(c)$ is the prior probability of class c which usually is calculated as a unigram language model on the specific class c (Merkel and Klakow, 2007) or can simply be considered equal for all classes (Zhai and Lafferty, 2001).

Li (1999) used this approach for question classification. He compared the result of language modeling classification with a rule based regular expression method on the old TREC dataset and the results reveal that language modeling approach perform much better than traditional regular expression method. Merkel and Klakow (2007) proposed same approach for question classification and reported an accuracy of 80.8% on TREC dataset. The main difference of language modeling method compare to other classification approaches is that there is no need to extract complex features from a question. To obtain better results, it would be useful if the language model is trained with larger training sets.

5.2.6 Other Classifiers

In addition to the mentioned classifiers, other types of classifiers have also been used for question classification. Li et al. (2008) adopted the SVM together with Conditional Random Fields (CRFs) for question classification. CRFs are a type of discriminative probabilistic model which is used for labeling sequential data. In the model proposed by Li

et al. (2008), a question is considered as a sequence of semantically related words. They use CRFs to label all the words in a question and the label of the *head word* is considered as the question class (head word extraction is described in section 4). Their approach differs with other question classification approaches in the sense that a question is considered as *sequential* data. Therefore it can extract features from *transition* between states as well as other common syntactic and semantic features. They reported an accuracy of 85.6% on fine-grained classes of TREC dataset.

Zhang and Lee (2003) compared the accuracy of question classification by 5 different classifiers on same feature space. They compared SVMs with Nearest Neighbor (NN), Naive Bayes (NB), Decision Tree (DT) and SNoW among which SVM performed the best. Their results on fine-grained classes of TREC dataset is listed in table 5.2.

Table 5.2: The accuracy of 5 different classifiers on TREC dataset with bag-of-word features; taken from Zhang and Lee (2003)

Approach	**Accuracy(fine)**	**Accuracy(coarse)**
NN	68.4%	75.6%
NB	58.4%	77.4%
DT	77.0%	84.2%
SNoW	74.0%	66.8%
SVM	80.2%	85.8%

The results from table 5.2 reveal that SVMs perform better compare to other classifiers when same feature space are used. However, depending on the extracted features, other classifiers may perform better. For example SVMs perform better rather than ME when semantic features are used (Huang et al., 2008), but on the other hand ME shows better performance when syntactical sub-trees are used as features (Le Nguyen et al., 2007). Therefore no specific classifier can always be preferred to other classifiers for question classification. Depending on feature space and other parameters, the optimal classifier can be different.

5.2.7 Combining Classifiers

Question classification has also been studied by combining different classifiers. Combination of classifiers can be done by different approaches. Xin et al. (2005) trained four SVM classifier based on four different type of features and combined them with various strategies. They compared Adaboost, (Schapire, 1999), Neural Networks and Transition-Based Learning (TBL) (Brill, 1995) combination methods on the trained classifiers. Their result on TREC dataset reveals that using TBL combination method can improve classification accuracy upto 1.6% compare to a single classifier which is trained on all features.

5.3 Features

Different studies extracted different lexical, syntactical and semantic features. In addition to the features that we mentioned in chapter 3, still there are some features that are used in other works. Blunsom et al. (2006) introduced question's length as a separate lexical feature. It is simply the number of words in a question. For example for the question "How many Grammys did Michael Jackson win in 1983 ?" the following feature represents the question-length features based on the vector space model:

$$\{(\text{question-len}, 10)\} \tag{5.11}$$

Different syntactical features are also extracted in different studies. Some studies added the POS tags of all the word in the feature vector and consider this as a feature set. This feature set is also referred as *bag-of-pos tags*. For the same question its bag-of-POS features are as follow:

$$\{(\text{WRB}, 1)\ (\text{JJ}, 1)\ (\text{NNPS}, 1)\ (\text{VBD}, 1)\ (\text{NNP}, 1)\ (\text{NNP}, 1)\ (\text{VBP}, 1)\ (\text{IN}, 1)\ (\text{CD}, 1)\} \tag{5.12}$$

In addition to the mentioned syntactic features, Blunsom et al. (2006) also considered the POS tag of the headword as a separate feature. Li and Roth (2004) introduced *head chunk* as a syntactical feature. The first noun chunk and the first verb chunk after the question word are considered as head chunk. For example for the question "What is the oldest city in Canada ?" the first noun chunk is "the oldest city in Canada" since it is the first noun phrase appearing after question word.

Krishnan et al. (2005) introduced a feature namely *informer span* which is defined as short subsequent of words that are adequate clue for question classification. They extract it based on a sequential graphical model with features derived from parse tree. For example for the question "What is the tallest mountain in the world ?" the informer span is "tallest mountain". Informer span and head chunks features are added to feature vector in the same way as unigrams, i.e., all the words in head chunk or informer span are considered as a feature (table 3.5). Williams (2010) also considered the bigram and trigram of informer span as separate features.

Head chunks and informer span are very similar to headwords but they are usually a *sequence* of words instead of a single word. The extra words usually can introduce noisy information leading to lower accuracy rate. For example consider the question "What is a group of turkeys called ?". The headword of this question is "turkeys" while both head chunk and informer span are "group of turkeys" (Huang et al., 2008). The word "turkeys" can truly contribute to the classification of type ENTY:animal while the word group can mislead the classifier to classify this question to HUM:group. Therefore usually a single and exact headword is preferred to head chunk or informer span.

Xin et al. (2005) introduced a feature namely *words dependency* which is extracted using syntactical structure of question. Dependent words are very similar to Bigram but they are not limited to consecutive words. For example in the question "Which company created the Internet browser Mosaic ?", "Internet" and "Mosaic" are two dependent word that can not be determined by Bigram. Dependency features are treated similar to Bigram

when they added to feature vector. In the mentioned example "Internet-Mosaic" is a single feature that can be added to feature vector.

In addition to the semantic features that we used in this work, there are still some semantic features which are used in other studies. Since they are not powerful features for question classification, we have not used them in our work. One of these features which is used in some studies (Li and Roth, 2004; Blunsom et al., 2006) is *named entities*. Named entities are semantic categories which can be assigned to some words in a given sentence.

Successful approaches such as Markov Models (Punyakanok and Roth, 2001) and unsupervised methods (Collins and Singer, 1999) have been employed for Named Entity Recognition (NER). Punyakanok and Roth (2001) introduced 34 semantic categories for named entity recognition and reported an accuracy of more than 90.0% on determining named entities. For example for the question "Who was the first woman killed in Vietnam War ?", their NER system identifies the following named entities: "Who was the [**number** first] woman killed in [**event** Vietnam War] ?"

In question classification the identified named entities can be added to the feature vector. Based on the representation (3.20) the named entity features for the aforementioned sample will be as follow: {(number, 1) (event, 1)}.

Blunsom et al. (2006) considered the named entity of the headword as a separate feature set due to importance of this word.

In addition to the mentioned semantic features, some studies *indirectly* used WordNet to extract semantic features. Huang et al. (2008) measured the similarity of question's headword with all question classes using WordNet hierarchy and considers the most similar category as a semantic feature.

Li and Roth (2004) uses WordNet to extract synonyms of the context words of a question and adds them to feature vector. Ray et al. (2010) uses Wikipedia to find the description of the words in a question and identifies semantic categories (named entities) of them with a rule-based algorithm

Table 5.3 lists the semantic features discussed in this section for the sample question "What is the oldest city in Canada ?". The features are represented same as equation (3.20). Note that if a feature value is larger than 1, it means that the corresponding feature is extracted from more than one word. For example in the mentioned sample there are two different words (city and Canada) both have name entity "location". Therefore the named entity features of this question will be {(location, 2)}.

Table 5.3: Example of semantic features

Feature Space	**Features**
named entities	{(location, 2)}
headword named entity	{(location, 1)}
indirect hypernym	{(LOC:city, 1)}

5.4 Comparison of Supervised Learning Approaches

All the methods described till now are supervised learning approaches which mainly differ in the classifier they used and the features they extract. Table 5.4 compares some studies on supervised learning question classification with our work. All of these studies uses TREC dataset for evaluation. Our weighted approach to combine features achieves the highest accuracy.

From the result in table 5.4 it is not easy to say which classifier or which combination of features is the best choice for question classification as each method has its own advantages and disadvantages. It is however obvious that when the classifiers are trained on a richer feature space (not necessarily higher dimensional feature space), they can give a better performance. Syntactical and semantic features can usually add more information to feature space and improve classification accuracy. Since features in question classification are very dependent, usually combining all features together is not an optimal choice of features and depending on the decision model the best combination of features can be differ.

5.5 Semi-Supervised Learning in Question Classification

Providing labeled questions is a costly process since it needs human effort to manually label questions while unlabeled question can be easily obtained from many web resources. Semi-supervised learning tries to exploit unlabeled information as well as labeled data. In this section we introduced semi-supervised techniques which have been used for question classification.

5.5.1 Co-Training

A successful semi-supervised learning algorithm which is widely used in natural language processing is *Co-training* (Blum and Mitchell, 1998). Consider we are given a training set D which consist of a labeled part $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and unlabeled part $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$. Co-training makes the strong assumption that each instance $\mathbf{x}_i$ has two views: $\mathbf{x}_i = [\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(2)}]$ such that each view consists of separate features. The co-training algorithm trains two different classifiers with labeled data for each view. The remaining unlabeled samples are classified with both classifiers and the top most-confident predictions of first view will be added to the labeled samples of second view and vice versa. The classifiers then will be re-trained and the same process continues until all unlabeled samples are used up. Algorithm 3 lists the co-training style semi-supervised learning (Zhu and Goldberg, 2009).

Yu et al. (2010) applied co-training to question classification. They adopted two tree-based classifiers each of which are trained based on separate features. Their result on a Chinese questions dataset reveals that under 40% rate of unlabeled data, the classification accuracy can improve up to 4 percent compare to supervised approach.

A slightly different version of co-training which have also been used in question classification is *Tri-training* (Li et al., 2008). Tri-training uses 3 classifiers instead of two. If

Algorithm 3 Co-training Algorithm

input labeled data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and unlabeled data $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, a learning rate k
  $L_1 = L_2 = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_l, y_l)\}$
  repeat
    train a view-1 classifier $f^{(1)}$ from L_1 and a view-2 classifier $f^{(2)}$ from L_2
    classify the remaining unlabeled data with $f^{(1)}$ and $f^{(2)}$ separately
    add top k most-confident prediction $(\mathbf{x}, f^{(1)}(\mathbf{x}))$ to L_2
    add top k most-confident prediction $(\mathbf{x}, f^{(2)}(\mathbf{x}))$ to L_1
    remove the added samples from unlabeled data
  until unlabeled data is used up

two of the three classifiers agree to label an unlabeled instance, that instance is used for re-training other classifier.

Thanh et al. (2008) adopted tri-training for question classification. In their experiment they trained tree different classifiers: the first classifier is an SVM classifier with bag-of-word features, the second is another SVM classifier with bag-of-POS features and the third classifier is a maximum entropy classifier with both bag-of-words and bag-of-POS features. They divided TREC dataset into labeled and unlabeled parts. They compared the classification accuracy when only labeled instances are used with the situation where both labeled and unlabeled instances are used. The result shows that irrespective to the ratio of unlabeled data, the classification accuracy always increases when unlabeled samples are exploited. Table 5.5 lists the classification accuracy on TREC dataset when only labeled data is used and when both labeled and unlabeled data are used with tri-training algorithm.

5.6 Summary

In this chapter we introduced some related studies on question classification. The numerous studies on question classification show the importance of this problem in question answering systems. We briefly introduced different statistical learning approaches which have been used in other works together with features which have been extracted in different studies.

We compared the performance of related works with our work based on their accuracy on the standard TREC dataset. Our linear SVM classifier achieved the highest accuracy on this task. Furthermore, we tried to represent the question as much compact as possible. Our BPNN classifier achieves a competitive accuracy compare to other works, given that it only uses 400 features which is much less than the size of feature space in other works.

Table 5.4: Comparison of different supervised learning studies on question classification on TREC dataset. The abbreviation of features are:
U: Unigrams, **B**: Bigrams, **LB**: Limited-Bigrams, **T**: Trigrams, **NG**: N-grams, **WH**: Wh-word, **WS**: Word-Shapes, **L**: Question-Length, **P**: POS-tags, **H**: Headword, **HC**: Head-Chunk, **IS**: Informer-Span, **HY**: Hypernyms, **IH**: Indirect-Hypernyms, **QE**: Query-Expansion, **QC**: Question-Category, **S**: Synonyms, **NE**: Name-Entities, **R**: Related-Words

Study	**Classifier**	**Features**	**Accuracy**	
			Coarse	Fine
Li and Roth (2002)	SNoW	U+P+HC+NE+R	91.0%	84.2%
Zhang and Lee (2003)	Tree kernel SVM	U+NG	90.0%	-
Li and Roth (2004)	SNoW	U+P+HC+NE+R+S	-	89.3%
Metzler et al. (2005)	RBF kernel SVM	U+B+H+HY	90.2%	83.6%
Krishnan et al. (2005)	Linear SVM	U+B+T+IS+HY	94.2%	88.0%
Blunsom et al. (2006)	ME	U+B+T+P+H+NE+more	92.6%	86.6%
Merkel et al. (2007)	Language Modeling	U+B	-	80.8%
Li et al. (2008)	SVM+CRF	U+L+P+H+HY+NE+S	-	85.6%
Pan et al. (2008)	Semantic tree kernel SVM	U+NE+S+IH	94.0%	-
Huang et al. (2008)	ME	U+WH+WS+H+HY+IH	93.6%	89.0%
Huang et al. (2008)	Linear SVM	U+WH+WS+H+HY+IH	93.4%	89.2%
Silva et al. (2011)	Linear SVM	U+H+HY+IH+more	**95.0**%	90.8%
Loni et al. (2011)	Linear SVM	U+B+WS+H+HY+R	93.6%	89.0%
This Work	Linear SVM	U+B+WS+H+QE+QC+R	94.8%	**91.0**%
This Work	BPNN	WH+LB+WS+H+R	93.8%	-

Table 5.5: Comparison of classification accuracy of supervised and semi-supervised learning based on different ratio of unlabeled data.

no. labeled	**no. unlabeled**	**Supervised**	**Tri-training**
1000	4452	71.0%	71.2%
2000	3452	76.4%	78.2%
3000	2452	79.0%	79.2%
4000	1452	80.8%	81.4%

Chapter 6

Conclusions and Future Works

6.1 Conclusions

Question classification is a hard problem. In fact the machine needs to understand the question and classify it to the right category. This is done by a series of complicated steps. In this work we extracted 12 different lexical, syntactical and semantic features and used two different classifiers, support vector machines and back-propagation neural networks, to classify a natural language question to a pre-defined category.

Enhancing the feature space with syntactic and semantic features can usually improve the classification accuracy. However, augmenting the feature space with more complicated features can sometimes introduce noisy information to the feature space leading to misclassification. Furthermore, due to high correlation of syntactical and semantic features, combining all possible feature spaces does not necessarily lead to higher classification accuracy.

In this work we found that semantic information which is obtained from third party sources are in fact useful to better classify questions. However they can also add redundant information to the feature vectors. To reduce the influence of redundant features and to intensify the influence of useful features, our weighted mechanism is employed.

In this work we tested different combination of features to see the contribution and influence of each feature set in classification accuracy. We introduced two enhanced semitic features namely query-expansion and question-category. These features perform better than similar semantic features such as hypernyms which introduced in Huang et al. (2008). Furthermore, we introduced a weighted approach to combine different lexical, syntactical and semantic features in an optimal way. Our weighted method also improves the accuracy of classification. In fact, considering a weight parameter for features which reflects the importance of features can better represent the questions. Our query-expansion feature set also benefits from this assumption.

In this work we used neural networks for question classification for the first time. Our back-propagation neural network classifier performed better than the linear SVM, on the reduced space for the coarse grained classes.

An interesting result that we obtained from the LSA feature reduction technique is

that when the original feature space is compact, its reduced space performs better than a rich feature space with many dimensions. In fact reducing a compact feature space to a smaller space can loose less information while the redundant information is removed.

Analyzing the misclassification causes reveals that "what" type questions are typically more difficult to be classified compare to other type of questions. More accurate feature extraction techniques are needed to deal with this type of questions. Similar to the study of Li et al. (2008), a separate classifier can be used for classifying what type questions.

Analysis of the results also reveals that some samples which are misclassified by a particular classifier can be correctly classified when another classifier is used. A *dynamic classifier selection* approach can be applied for QC problem in future works to deal with this issue.

6.2 Future Works

Question classification still attracts a notable amount of research and different extension to this work can be made. One possible extension to the current works is to extract features in a *dynamic* way so that the questions which have enough information for classification not been expanded by more complicated features. For example, we found a few samples which are correctly classified with simple unigrams but they are misclassified when semantic features are added to them. It is of course not simple for our classifier to decide which samples are augmented with semantic features and which not, but one can for example introduce a confident value for the assigned label and using this value, only the samples whose confident is lower than a threshold are expanded with semantic features.

Another extension to this work is to augment our latent semantic space with semantic information from third party sources to make our semantic space more informative. Our weighted method can also be applied in the latent semantic space to reflect the importance of features in the reduced space. Also a combination of original and latent semantic space can be examined in the future works. For example for the high dimensional features such as unigrams and bigrams we can apply the LSA technique to reduce them to a small space and then combine this space with low dimensional features such as word-shapes, head-words and related words.

In this work we obtained better performance by combining different feature sets. Combining different *classifiers* can also be done in future studies to see whether they can improve classification accuracy or not. For example,we found some samples which are misclassified with our SVM classifier while they are correctly classified with our BPNN classifier. Classifier combination strategies such as dynamic classifier selection can be used to benefit from both classifiers.

Exploiting unlabeled data with semi-supervised approaches can usually improve the classification accuracy. It should be noticed that unlabeled information can sometimes introduce noisy samples to the training samples and this noise can be amplified by retraining the classifiers. Therefore semi-supervised approaches should always been used with a proper percentage of labeled and unlabeled samples on a certain amount of confident. The co-training method has been successfully applied on question classification. It is

still possible to apply other semi-supervised approaches such as *expectation maximization* on question classification to see whether they are useful for this task or not.

The problem of question classification still is in the cutting edge of question answering systems. By extracting richer set of features and improving current feature extraction techniques together with more advance techniques such as semi-supervised learning, we hope that more powerful systems can be developed for question classification.

Appendices

Appendix A: Part of Speech Tags

Tables 1, 2 and 3 list clause-level, phrase-level and word-level POS tags of English grammar, respectively[1]. Bies (1995) provided a detailed overview of English POS tags and their application in natural language parsing.

Table 1: The list of clauses-level POS tags

1	S	simple declarative clause
2	SBAR	Clause introduced by a (possibly empty) subordinating conjunction
3	SBARQ	Direct question introduced by a wh-word or a wh-phrase
4	SINV	Inverted declarative sentence
5	SQ	Inverted yes-no question, or main clause of a wh-question, following the wh-phrase in SBARQ

[1] http://bulba.sdsu.edu/jeanette/thesis/PennTags.html

Table 2: The list of phrase-level POS tags

1	ADJP	Adjective Phrase
2	ADVP	Adverb Phrase
3	CONJP	Conjunction Phrase
4	FRAG	Fragment
5	INTJ	Interjection
6	LST	List marker
7	NAC	Not a Constituent
8	NP	Noun Phrase
9	NX	Used within certain complex NPs to mark the head of the NP
10	PP	Prepositional Phrase
11	PRN	Parenthetical
12	PRT	Particle Category for words that should be tagged RP
13	QP	Quantifier Phrase
14	RRC	Reduced Relative Clause
15	UCP	Unlike Coordinated Phrase
16	VP	Vereb Phrase
17	WHADJP	Wh-adjective Phrase
18	WHAVP	Wh-adverb Phrase
19	WHNP	Wh-noun Phrase
20	WHPP	Wh-prepositional Phrase
21	X	Unknown, uncertain, or unbracketable

Table 3: The list of word-level POS tags

1	CC	Coordinating conjunction
2	CD	Cardinal number
3	DT	Determiner
4	EX	Existential "there"
5	FW	foreign word
6	IN	Preposition or subordinating conjunction
7	JJ	Adjective
8	JJR	Adjective, comparative
9	JJS	Adjective, superlative
10	LS	List item marker
11	MD	Modal
12	NN	Noun, singular or mass
13	NNS	Noun, plural
14	NNP	proper noun, singular
15	NNPS	proper noun, plural
16	PDT	Predeterminer
17	POS	Possessive ending
18	PP	Personal pronoun
19	PP$	Possessive pronoun
20	RB	Adverb
21	RBR	Adverb, comparative
22	RBS	Adverb, superlative
23	RP	Particle
24	SYM	Symbol
25	TO	"to"
26	UH	Interjection
27	VB	Verb, base form
28	VBD	Verb, past tense
29	VBG	Verb, gerund or present participle
30	VBN	Verb, past participle
31	VBP	Verb, non-3rd person singular present
32	VBZ	Verb, 3rdperson singular present
33	WDT	Wh-determiner
34	WP	Wh-pronoun
35	WP$	Possessive wh-pronoun
36	WRB	Wh-adverb

Bibliography

Janna Anderson. Those who understand the semantic web are split on its future, May 2010. URL `http://www.pewinternet.org/Press-Releases/2010/Semantic-Web.aspx`.

Ion Androutsopoulos, Graeme D. Ritchie, and Peter Thanisch. Natural language interfaces to databases—an introduction. *Natural Language Engineering*, 1(1):29–81, 1995.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22:39–71, 1996.

Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web (Berners-Lee et. al 2001). May 2001.

A. Bies. Bracketing Guidelines for Treebank II Style Penn Treebank Project, 1995.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, COLT' 98, pages 92–100, New York, NY, USA, 1998. ACM. ISBN 1-58113-057-0.

Phil Blunsom, Krystle Kocik, and James R. Curran. Question classification with log-linear models. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 615–616, New York, NY, USA, 2006. ACM.

Eric Brill. Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. *Comput. Linguist.*, 21:543–565, December 1995.

Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2:121–167, June 1998. ISSN 1384-5810.

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. URL `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Alexander Clark. Inducing syntactic categories by context distribution clustering. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on*

Computational natural language learning - Volume 7, ConLL '00, pages 91–94, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

Michael Collins. *Head-Driven Statistical Models for natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110, 1999.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press, 1998.

B.F. Green, A.K. Wolf, C. Chomsky, and K. Laughery. Baseball: An automatic question answerer. In *Proceedings Western Computing Conference*, volume 19, pages 219–224, 1961.

Ulf Hermjakob, Eduard Hovy, and Chin yew Lin. Automated question answering in webclopedia - a demonstration. In *In Proceedings of ACL-02*, 2002.

Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel methods in machine learning. July 2008.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin yew Lin, and Deepak Ravichandran. Toward semantics-based answer pinpointing, 2001.

Yu Hen Hu. *Handbook of Neural Network Signal Processing*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2000.

Zhiheng Huang, Marcus Thint, and Zengchang Qin. Question classification using head words and their hypernyms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, (EMNLP '08), pages 927–936, 2008.

Zhiheng Huang, Marcus Thint, and Asli Celikyilmaz. Investigation of question classifier in question answering. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, (EMNLP '09), pages 543–550, 2009.

David A. Hull. Xerox TREC-8 question answering track report. In *In Voorhees and Harman*, 1999.

A. Ittycheriah, M. Franz, W. J. Zhu, A. Ratnaparkhi, and R. J. Mammone. IBM's statistical question answering system. In *Proceedings of the 9th Text Retrieval Conference, NIST*, 2001.

John Judge, Aoife Cahill, and Josef van Genabith. Questionbank: creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 497–504, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition) (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 2 edition, 2008.

Boris Katz, Sue Felshin, Deniz Yuret, Ali Ibrahim, Jimmy Lin, Gregory Marton, Alton Jerome McFarland, and Baris Temelkuran. Omnibase: Uniform access to heterogeneous data for question answering. In *In proceeding of the 7th international workshop on applications of natural language to information systems (NLDB)*, 2002.

Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *In Proceeding og the 41st annual meeting of the association for Computational Linguistic*, pages 423–430, 2003.

Krystle Kocik. Question classification using maximum entropy models. Technical report, 2004.

Vijay Krishnan, Sujatha Das, and Soumen Chakrabarti. Enhanced answer type inference from questions using sequential models. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 315–322, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

Savio L. Y. Lam and Dik Lun Lee. Feature reduction for neural network based text categorization. In *Proceedings of the Sixth International Conference on Database Systems for Advanced Applications*, DASFAA '99, pages 195–202, Washington, DC, USA, 1999. IEEE Computer Society.

Minh Le Nguyen, Thanh Tri Nguyen, and Akira Shimazu. Subtree mining for question classification problem. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 1695–1700, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

Wendy G. Lehnert. A conceptual theory of question answering. In *Proceedings of the 5th international joint conference on Artificial intelligence - Volume 1*, pages 158–164, San Francisco, CA, USA, 1977. Morgan Kaufmann Publishers Inc.

Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26, 1986.

Fangtao Li, Xian Zhang, Jinhui Yuan, and Xiaoyan Zhu. Classifying what-type questions by head noun tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 481–488, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

Wei Li. Question classification using language modeling, 1999.

Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics*, COLING '02, pages 1–7. Association for Computational Linguistics, 2002.

Xin Li and Dan Roth. Learning question classifiers: The role of semantic information. In *In Proc. International Conference on Computational Linguistics (COLING*, pages 556–562, 2004.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Mach. Learn.*, 2:285–318, April 1988.

Babak Loni, Gijs van Tulder, Pascal Wiggers, Marco Loog, and David Tax. Question classification with weighted combination of lexical, syntactical and semantic features. In *Proceedings of the 15th international conference of Text, Dialog and Speech*, 2011.

Andreas Merkel and Dietrich Klakow. Improved methods of language model based question classification. In *In Proceedings of Interspeech Conference*, 2007.

Donald Metzler and W. Bruce Croft. Analysis of statistical question classification for fact-based questions. *Inf. Retr.*, 8:481–504, May 2005.

Dan Moldovan, Marius Paşca, Sanda Harabagiu, and Mihai Surdeanu. Performance issues and error analysis in an open-domain question answering system. *ACM Trans. Inf. Syst.*, 21:133–154, April 2003.

Vanessa Murdock and W. Bruce Croft. Task orientation in question answering. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 355–356, New York, NY, USA, 2002. ACM.

Yan Pan, Yong Tang, Luxin Lin, and Yemin Luo. Question classification with semantic tree kernel. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 837–838, New York, NY, USA, 2008. ACM.

Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics, Proceedings of the Main Conference*, pages 404–411, 2007.

Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. pages 275–281, 1998.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, IUI '03, pages 149–157, New York, NY, USA, 2003. ACM.

John Prager, Dragomir Radev, Eric Brown, and Anni Coden. The use of predictive annotation for question answering in trec8. In *In NIST Special Publication 500-246:The Eighth Text REtrieval Conference (TREC 8*, pages 399–411. NIST, 1999.

Vasin Punyakanok and Dan Roth. The use of classifiers in sequential inference. *Computing Research Repository*, 2001.

Santosh Kumar Ray, Shailendra Singh, and B. P. Joshi. A semantic approach for question classification using wordnet and wikipedia. *Pattern Recogn. Lett.*, 31:1935–1943, October 2010.

Dan Roth. Learning to resolve natural language ambiguities: a unified approach. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, AAAI '98/IAAI '98, pages 806–813, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.

Robert E. Schapire. Theoretical views of boosting and applications. In *Proceedings of the 10th International Conference on Algorithmic Learning Theory*, ALT '99, pages 13–25, London, UK, 1999. Springer-Verlag. ISBN 3-540-66748-2.

Hinrich Schütze and Yoram Singer. Part-of-speech tagging using a variable memory markov model. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, ACL '94, pages 181–187, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics.

N. Seco, T. Veale, and J. Hayes. An intrinsic information content metric for semantic similarity in WordNet. *Proc. of ECAI*, 4:1089?1090–1089?1090, 2004.

João Silva, Luísa Coheur, Ana Mendes, and Andreas Wichert. From symbolic to subsymbolic information in question classification. *Artificial Intelligence Review*, 35(2): 137–154, February 2011.

R. F. Simmons. Answering english questions by computer: a survey. *Commun. ACM*, 8: 53–70, January 1965. ISSN 0001-0782.

Nguyen Tri Thanh, Nguyen Le Minh, and Shimazu Akira. Using semi-supervised learning for question classification. *Information and Media Technologies*, 3(1):112–130, 2008. ISSN 1881-0896.

David Tomas and Claudio Giuliano. A semi-supervised approach to question classification. In *The European Symposium on Artificial Neural Networks*, 2009.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1073445.1073478. URL `http://dx.doi.org/10.3115/1073445.1073478`.

Vladimir N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.

Ellen M. Voorhees. Overview of the trec 2001 question answering track. In *In Proceedings of the Tenth Text REtrieval Conference (TREC*, pages 42–51, 2001.

Ellen M. Voorhees and Donna Harman. Overview of the eighth text retrieval conference (trec-8). pages 1–24, 2000.

Andrew R. Webb. *Statistical Pattern Recognition, 2nd Edition.* John Wiley & Sons, October 2002.

Olalere Williams. High-performance question classification using semantic features. Standford University, 2010.

W. A. Woods. Progress in natural language understanding: an application to lunar geology. In *Proceedings of the June 4-8, 1973, national computer conference and exposition*, AFIPS '73, pages 441–450, New York, NY, USA, 1973. ACM.

Li Xin, HUANG Xuan-Jing, and WU Li-de. Question classification using multiple classifiers. In *Proceedings of the 5th Workshop on Asian Language Resources and First Symposium on Asian Language Resources Network*, 2005.

Bo Yu, Zong-ben Xu, and Cheng-hua Li. Latent semantic analysis for text categorization using neural network. *Know.-Based Syst.*, 21:900–904, December 2008.

Zhengtao Yu, Lei Su, Lina Li, Quan Zhao, Cunli Mao, and Jianyi Guo. Question classification based on co-training style semi-supervised learning. *Pattern Recogn. Lett.*, 31: 1975–1980, October 2010. ISSN 0167-8655.

Sarah Zelikovitz and Haym Hirsh. Using lsi for text classification in the presence of background text. In *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, pages 113–118, New York, NY, USA, 2001. ACM. ISBN 1-58113-436-3.

Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 334–342, New York, NY, USA, 2001. ACM.

Dell Zhang and Wee Sun Lee. Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 26–32, New York, NY, USA, 2003. ACM.

Xiaojin Zhu and Andrew B. Goldberg. *Introduction to Semi-Supervised Learning.* Morgan & Claypool Publishers, 2009.