Question Answering using Vector Based Information Retrieval Paradigm with Word Sense Disambiguation

Kavita A. Ganesan

University of Southern California
Computer Science Department
Los Angeles, CA, 90089
ganesan@usc.edu

ABSTRACT

Vector Based Text Classification in the Question Answering field has long been explored. However, there has not been any attempt so far to take word senses into consideration in the development of the feature sets in the classifier. This paper aims to investigate the performance of a question answering text classifier built using not just the root form of words but also taking the senses of those words into thought. Having done a 10-folded cross validation, the classification error rate using the tri-gram model actually shows that there is a significant improvement when the word sense of a word is actually known. A chi squared analysis performed to see the correlation between the use of word sense and the affect on the classification accuracy shows a 97.5% confidence level. This simply tells us that the usage of word sense in building the classifier has indeed a strong association with the classification accuracy.

1. INTRODUCTION

Automated question answering essentially involves providing precise answers to questions posed to the system. Question answering can be either factoid based where there is a single answer to a particular question or it could be connected dialogue where you have a series of related questions and answers.

The objective of this paper is to analyze the performance of a text classifier (built from the scratch) that provides answers to questions in the Sergeant Blackwell domain [1]. See figure 1 for an example of a possible question and possible answers to that particular question in the Sergeant Blackwell domain. Although much work in the Question Answering field using text classification has already been done, there has not been an attempt so far to take word senses into consideration in the development of the text classifier. Classifiers so far have been simply built using the root forms of the words.

The notion of including word senses in developing a text-classifier is that by knowing the part of speech usage of a particular word, the power of discrimination of that word would increase. This then would lead to the improvement in classification accuracy.

The next section briefly touches on related work done in text classification. Section 3, gives you a short description about the Sergeant Blackwell corpus used in building the text classifier. Section 4 walks you through the entire training process right from removal of noise words through the feature vector construction. The similarity measure used to compute the similarity between the document vector and query vector is discussed in section 5 and the section that follows talks about how the classification accuracy is evaluated. The paper ends with a summary of work done in this paper and a short discussion on ways to extend this work.

2. RELATED WORK

The work done in this paper is similar to the Vector-Based Natural Language Call Routing System [7]. However the developed system builds on the fact that every word has a different sense and hence word senses should be used for disambiguation during the classification process. This work also uses different evaluation methods as compared to [7].

3. CORPUS ANALYSIS

The corpus that was used in building the text classifier is known as the Sergeant Blackwell corpus. This corpus is from a project funded by the United States Army being carried out by the Institute for Creative Technologies (ICT). The corpus consists of 1596 questions and 79 answers (after the removal of duplication). Figure 1 illustrates a sample question and

its corresponding answer from the Sergeant Blackwell corpus. Each answer from the corpus has been grouped by all possible questions that can lead to that particular answer. For the purpose of simplicity each answer is referred to as a "class". See figure 2 for an example of questions grouped by answers.

Q1: Are there more virtual soldiers like you

ANS: *Technologically, I am made up of natural language dialogue and understanding. It is how we are talking right now. And my expressions - my face is done with state of the art facial animation research. And basically this presentation is made with a transcreen projection.*

Figure 1: Example of possible questions to a particular answer in the Sergeant Blackwell corpus

@CLASS1
Q1: Are there more virtual soldiers like you
Q2: Tell me something about your self.
:
@CLASS79
Q1: do you know what fort sill is?
Q2: how long will you be in the army?
:

Figure 2: Example of questions grouped by classes they belong to

4. TRAINING

The training process starts with the creation of one document per answer. This document will basically, contain all the questions that have a particular class of answer. There are several parts to the training process. The very first part is the removal of noise words from the corpus. Noise words are words uttered in spoken language but do not really carry any significant meaning. The next part is parsing the corpus using a dependency parser. This step is basically to be able to extract root form of words and determine the different word senses. This is the most important part in the development of the text classifier. The parsing is followed by a morphological filtering wherein irrelevant words are replaced with a placeholder.

The first three parts of the training process is explained in section 4.1. The fourth part basically involves term extraction in order to build the document feature vector and this is explained in section 4.2. Section 4.3 explains the feature vector representation used. Figure 3 illustrates the overall flow of the training process.

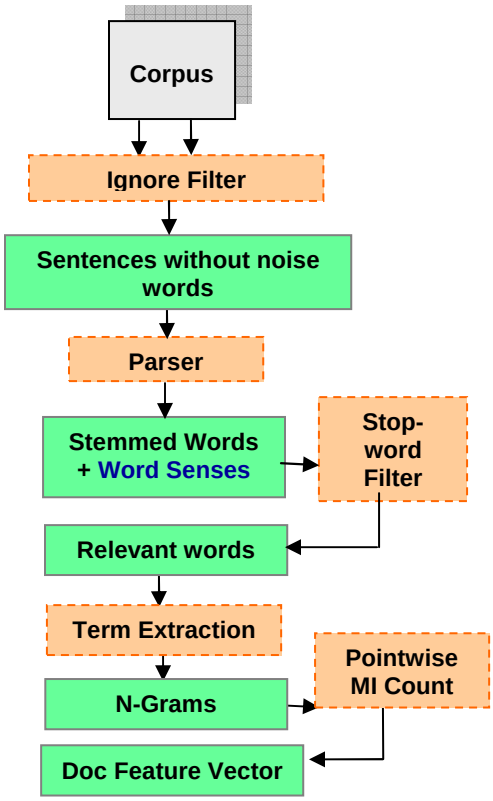

Figure 3: Training Process

4.1 Noise Word Filter, Parsing and Stop-word Filtering

The first step in building the classifier is to remove all noise words from the corpus. Noise words are simply words that are used in spoken language but do not carry any significant meaning. These are words like "*uh*","*um*", "*yeah*" and so on. This list of words is known as the ignore list. The ignore list was put together after a short study on the kind of noise words that can occur in spoken language was done.

The next step involves the parsing of the corpus with noise words removed. A dependency parser known as Minipar [2] was used to parse the corpus. The reason for parsing each document is two folded. First, the parsing of each document will enable the extraction of the root form of each word. Secondly, the **part of speech** of each word can be determined and this is the part that really helps in the word sense disambiguation. For example the word "*train*" in the sentence "*I am going to train the students*" and the

word "*train*" in the sentence "*I have to catch a train in a while*" have totally different meanings. The first sentence refers to "*train*" in its verb sense. The second sentence on the other hand uses the word "*train*" in its noun sense. The ability to tell that these two words have actually very different meanings actually helps the classifier a great deal in returning more accurate answers. This is because one sense of a word may be relevant to one class and another sense of the same word may be relevant to some other answer class. Figure 1 illustrates the output of a sentence parsed using the **Minipar** parser. The words in red are the root form of the actual words. The words in green are the part of speech of a word. This is the information that is needed for the purpose of word sense disambiguation.

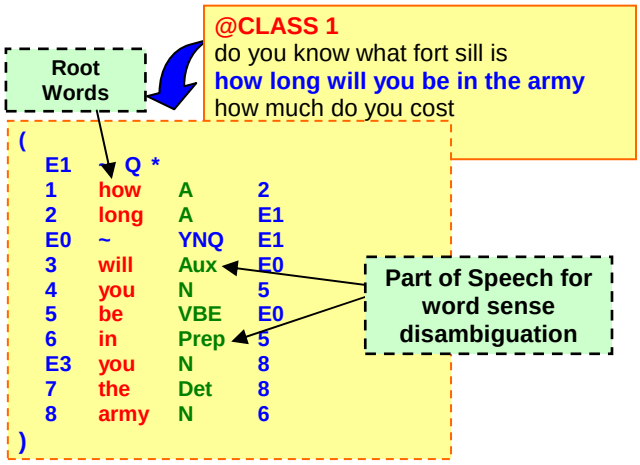

Figure 1: Example of a Minipar parsed sentence

The next part essentially involves the omission of unimportant words from each sentence in the corpus. Certain words like "*is*" and "*a*" are really irrelevant in determining the answer for a particular question. Such words are so frequently used that their existence would only confuse the classifier. The stop-word list used is a subset of the stop list in [5]. Not all words in [5] can be used as stop-words for this work because this list is mainly for written text. Certain words that are considered insignificant in written text may be in fact significant to dialogue utterances. Hence, a brief analysis was done on the stop-word list in [5] and only reasonable words were added into the actual stop list used to build the classifier.

Each document was run through the stop list filter and every occurrence of a stop-word is replaced with a placeholder "***W***". The reason for replacing the word with a placeholder string rather than entirely omitting the word is to prevent the classifier from forming incorrect n-grams. Each term or word that remains after the filter process will have its part of speech attached to it so that if the word sense is different, the classifier treats the word as an entirely different word. Figure 2 illustrates an example of the result of parsing, and stop-word filter process on the sentence "*How long will you be in the army?*"

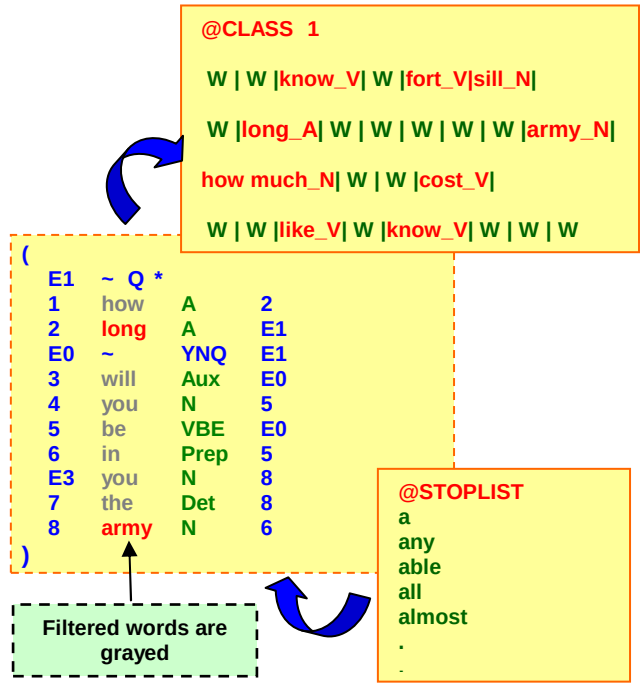

Figure 2: Example of parsing and stop-word filtering

4.2 Term Extraction

The result of the parsing and stop-word filtering process is a set of documents, one for each answer, containing only the root forms of the words with the different **senses** of those words in the context that they appear in. To be able to capture the collocation of words in the documents, n-gram terms and their respective term frequency counts have to be calculated. In this paper we capture tri-grams and bi-grams because after the stop-word filtering was done the highest order gram in the corpus were tri-grams. All highest order grams up to the lowest order grams are listed as features. For example, if you are building a tri-gram model, you will also have bi-grams and unigrams as part of the feature set. However, only the n-grams that occur sufficiently frequently will be eventually used in the training. A minimum feature count of two was initially used during the training process. An example of feature vector extraction is illustrated in figure 3.

4.3 Term Matrix Construction

Once relevant n-gram terms have been extracted, we construct a *P* x Q term-document frequency matrix where *P* represents the rows containing all the n-gram terms and *Q* represents the columns containing the destinations. Each entry $E_{t,d}$ is the frequency with which term *t* occurs in questions to answer *d*

.

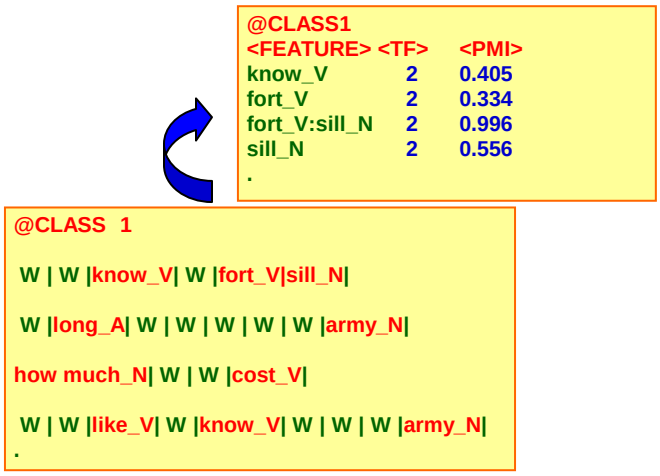

Figure 3: Feature vector extraction example

Using the raw frequency count alone is not sufficient to determine the closeness of the query to the destinations. This is due to the fact that by using only the raw frequency count, the weighting is only based on the number occurrences of terms/words in a particular document. There may be cases where a term does not only frequently occur in one document but rather occurs very frequently across all documents. Such terms obviously are not going to have very strong power of discrimination. Hence, such a term should not be weighted as heavily as compared to terms that occur less frequently in a particular document. Thus, the occurrence of a feature across documents has to be known in order to assign appropriate weights to features within a particular document. The weights assigned to features have to be representative of all features across all documents. For example, the bi-gram "*hard part*" may occur twice in a document and only three times across all documents. The unigram "*hard*" itself may occur ten times in the same document and a hundred times across all documents. Since the term "*hard part*" has better power of discrimination than frequent terms like "*hard*", the infrequent but important term should be given more weight. This can be done using measures like pointwise mutual information [3] or term-frequency x inverse document frequency (tf x idf). This paper mainly uses the pointwise mutual information for the representation of the feature vectors. Figure 3 shows an example of term frequency count and pointwise mutual information weights for features in a particular document.

4.3.1 Pointwise Mutual Information Calculation

Pointwise Mutual Information is roughly a measure of how much one word tells us about the other. Figure 4 illustrates how the pointwise mutual information calculation is done.

$$Mi_{ef} = \text{Log} \left((C_{ef} / N) / (\sum_{(i=1-n)} C_{if})/N \times (\sum_{(i=1-m)} C_{ej})/N \right)$$

Cef *is the frequency count of element e and feature f*
N *is the total frequency count of all features of all elements*

Figure 4: Pointwise Mutual Information calculation

5. ANSWERING QUERY

When an unseen question is posed to the classifier, the classifier treats the query as a new document. This document is called a pseudo document. The pseudo document will first go through the same process as the training process explained in section 4.Once this is done the similarity between the pseudo document and every other document has to be computed. There are many ways of computing document similarity. For this paper, the cosine similarity [4] measure has been used. See section 5.1 for details on how the cosine similarity is computed. Figure 5 illustrates the similarity measure data flow.

5.1 Cosine Similarity Measure

There are many different ways to measure how similar two documents are, or how similar a document is to a query. The cosine measure is a very common similarity measure used in information retrieval. Using a similarity measure, a set of documents can be compared to a query and the most similar document will be returned.

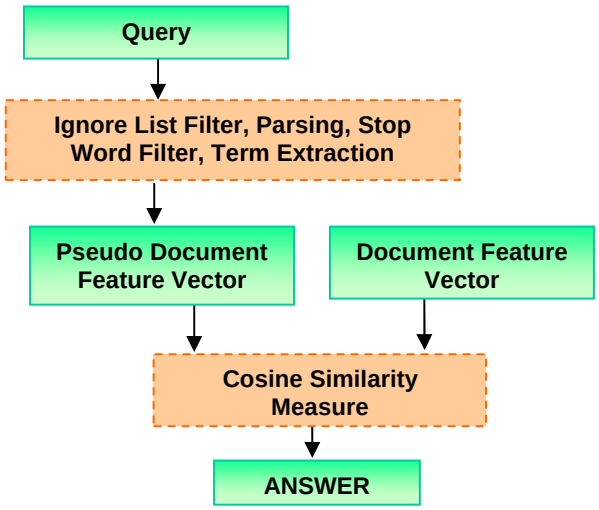

Figure 5: Cosine Similarity Measure Flow

The cosine similarity is calculated by measuring the cosine of the angle between two document vectors. For two vectors d and d' the cosine similarity between d and d' is given by:

$$\frac{d \times d'}{|d||d'|}$$

Here d X d' is the vector product of d and d', calculated by multiplying corresponding values together. Figure 6 illustrates an example of how the feature space is represented before the cosine similarity measure is done.

	F1	F2	F3	F4
@QUERY	0.23	0	0.3	0
@CLASS1	0.34	0.4	0.6	0
@CLASS2	0.67	0	0	0.78
@CLASS3 . .				

cosine similarity between query vector and each document vector.

Figure 6: Example of feature space representation for the cosine similarity measure

6. EVALUATION

The classifier developed was tested for classification error across different models using the 10-folded cross validation. Apart from that, a chi square analysis was also done to be able to see if there is significant association between the different classifier models and the classification accuracy. The next section steps you through the 10-folded cross validation done using the developed classifier. The section following that discusses how the chi squared analysis was done.

6.1 10-Folded Cross Validation

The 10-folded cross validation was done by randomizing the training data and partitioning the training data into 10 almost equal portions. During each training phase, the classifier is trained on nine of the segments and tested on the one segment not used in the training. This process is repeated ten times until all segments have been used for testing. The test segment is used to measure the average classification error rate across different models as shown in table 1.

GRAM	Usage of Word Senses & stop List Filter	Minimum Feature Count
BI-GRAM	o Sense + Stop List o Sense + No Stop List o No Sense + Stop List o No Sense + No Stop List	MINCOUNT 2
TRI-GRAM	o Sense + Stop List o Sense + No Stop List o No Sense + Stop List o No Sense + No Stop List	MINCOUNT 1,2,3 & 4

Table 1: The Different Models Classifier Trained On

Each segment in the training data contains approximately 159 samples. Since the cross validation is done ten times, there are approximately 1590 test samples altogether. The error rate across the different models is calculated using the following formula:

$$\text{Error Rate (E)} = \frac{\text{\# Of Samples} - \text{\# Of Samples Correctly Classified}}{\text{\# Of Samples}}$$

The error rate obtained in the ten rounds of test for each model is averaged out to get an approximate classification error rate. The training was initially done using bi-grams and tri-grams with/without word sense and with/without the usage of stop list filtering and with a minimum feature count of two. The classification error rate obtained for the tri-gram

model and the bi-gram model is as shown in figure 7 and figure 8.

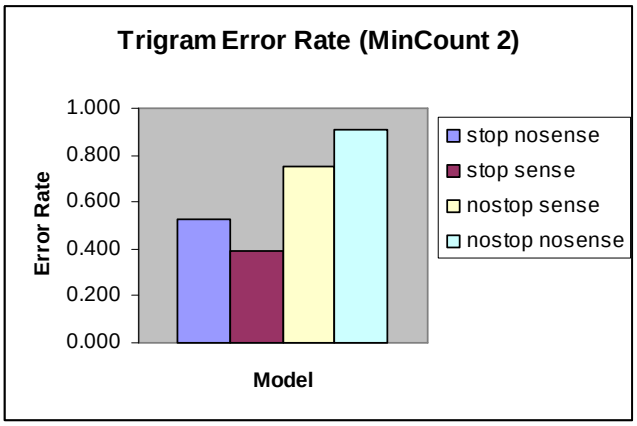

Figure 7: Error Rate across different tri-gram classifier models

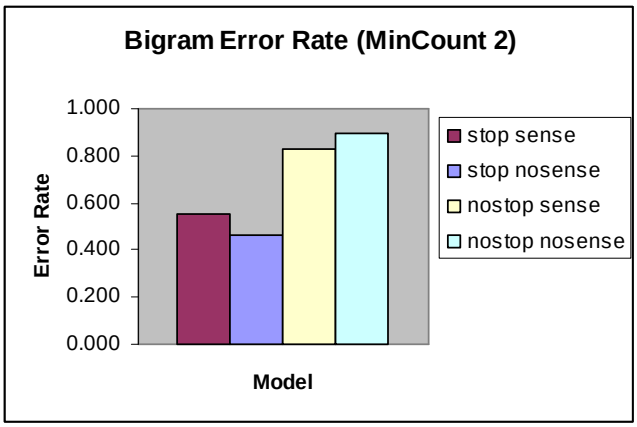

Figure 8: Error Rate across different bi-gram classifier models

As can be seen from figure 7, in the tri-gram model, the usage of word sense and stop-word filter results in an error rate of approximately 0.38 and is the model that has the lowest error rate. If you look at the error rate for the model that does not use word sense but uses only the stop-word filter you get an error rate of approximately 0.45. So, the model that uses the word sense shows an improvement of 0.07 in the classification accuracy.

On the contrary, if you study the bi-gram model's classification error in figure 8, you will find that the error rate is lowest for the model that does not use word sense but uses only the stop-word filter. The model that uses word sense with stop-word filter in fact, performs a little poorer than the afore mentioned model. Possible reasons for this could be that bi-grams are usually more common than tri-grams and hence have a lower power of discrimination. This in turn results in no significant difference in the classification accuracy. One important point to note would be that the lowest classification error rate in the tri-gram model is lower than the lowest classification error rate in the bi-gram model. Hence, best model that could improve classification accuracy would be the trigram model that has word sense disambiguation and uses the stop-word filter. I shall refer to this model as the **tri-sense-stop** model.

Since the classifier models were only trained with a minimum feature count of two, it is not known if the classification accuracy may improve with a different minimum feature count. Hence, the best tri-gram and bi-gram models are retrained with different minimum feature counts. The accuracy obtained for the different feature count cut offs are as shown in figure 9 and figure 10.

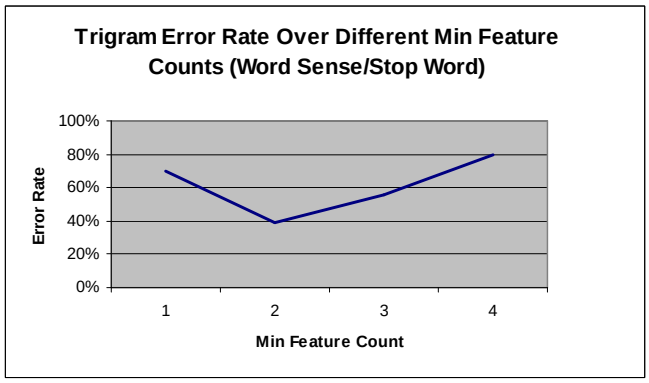

Figure 9: Classification accuracy for best tri-gram model using different minimum feature counts

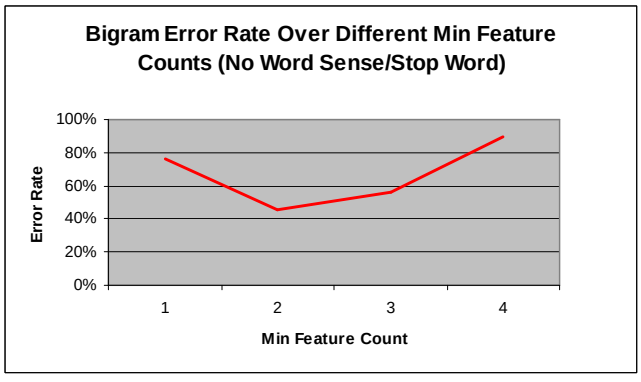

Figure 10: Classification accuracy for best bi-gram model using different minimum feature counts

From figure 9 and 10 you can see that the minimum feature count of two has the best classification accuracy. Classification seems to drop when the minimum feature count is too high or too low. Upon analyzing the raw count of features in the Sergeant Blackwell corpus, detailed observation shows that most of the important features occur at least twice and

not more than three/four times. Also, a lot of the unimportant features occur only once. So, this is the reason why the classification accuracy is poor when the minimum feature count is very low or very high.

6.2 Chi Square Analysis

In probability theory and statistics, the **chi-square distribution** or χ^2 **distribution**, is one of the theoretical probability distributions most widely used in inferential statistics, i.e. in statistical significance tests [6]. It is useful because, under reasonable assumptions, easily calculated quantities can be proved to have distributions that approximate to the chi-square distribution if the null hypothesis is true.

In order to perform a chi squared analysis, a tabular bivariate table is required. This table is also known as a contingency table and the values in this table should be raw frequency counts. In the 10-folded cross validation, in each round of test, a classification error rate was obtained. However, for chi squared analysis the classification error rate cannot be used. So, instead of using the classification error rate, the number of correct & incorrect classification across the ten folds of test was recorded and averaged. Since the training data has approximately 1590 samples, the average sample size is 159 (when divided by 10). The contingency table used for the purpose of this analysis is as shown in table 2.

Effect Of Word Sense on Classification Accuracy			
	Correct	**Incorrect**	**Total**
no sense+stop	75	84	159
sense+stop	96	63	159
Total	171	147	318

Table 2: Classification Contingency Table

The χ^2 value obtained from the data collected is approximately **5.5789**. With the degree of freedom of 1 (number of rows-1 * number of columns-1), and with a significance level of 0.025 (critical value=5.02), it can be said that the distribution is in fact significant (χ^2=5.5789 > 5.02). The chi square analysis shows a 97.5% confidence level that word sense actually has a strong correlation to the classification accuracy.

7. CONCLUSION AND FUTURE WORK

Though work in the area of text classification for question answering has been long explored, including word senses in the development of a classifier has not been delved into.

This paper provides insights into how a classifier that takes word senses into consideration has been developed. This paper also presents evaluation on the classification performance across different classifier models (with word sense, without word sense, with stop list, without stop list). The evaluation shows that the tri-gram classifier that uses word sense and stop-word filter with a minimum feature count of two has the lowest error rate. A Chi Square analysis was performed to confirm that the use of word sense for classification does have a strong correlation with classification accuracy. The result of this analysis was a 97.5% confidence level that the use of word sense during classification indeed has strong association with the classification accuracy.

Since text classification used in this work is mainly for spoken dialogue, the stop-word list compiled for the use with written text cannot be plugged in directly with spoken dialogue. For this project however, only a subset of the stop-words have been used after a short analysis of the stop-word list. Then again, the quality of the stop list used is not known. Hence more work needs to be done in this area.

In written text, words like *"cannot"* are really very insignificant, but this may not be the case in dialogue utterances. Spoken dialogue is more than often not shorter than written text and the amount of important information contained in a sentence in spoken dialogue may be way higher than that of written text. Hence, the use of a stop-word list tailored for written text if used with spoken dialogue may have a significant impact on the classification accuracy as important words in each dialogue utterance may be inadvertently eliminated.

One easy way to get a reasonable stop list suitable for spoken dialogue would be to study the stop list used in written text and train and test the classifier on different subsets of stop-words. The classifier can also be tested for accuracy using just prepositions, determiners and the like as stop-words.

Apart from experimenting with the stop-word list, the work done in this paper can also be extended by testing the classifier on different domains (that have more data) and also by combining domains to see if

the classifier performs any better, or just as well as with the Sergeant Blackwell domain.

8. REFERENCES

[1] http://jot.communication.utexas.edu/flow/?jot=view&id=1336

[2] Dekang Lin, Dependency-based evaluation of minipar, In In Workshop on the Evaluation of Parsing Systems Granada Spain, 1998.

[3] Patrick Pantel , Dekang Lin, Discovering word senses from text, Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, July 23-26, 2002, Edmonton, Alberta, Canada

[4] http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html

[5] http://www-a2k.is.tokushima-u.ac.jp/member/kita/NLP/lex.html

[6] http://www.georgetown.edu/faculty/ballc/webtools/web_chi_tut.html

[7] Carpenter, Bob and Jennifer Chu-Carroll. 1998. Natural language call routing: A robust, self-organizing approach. In Proceedings of the Fifth International Conference on Spoken Language Processing, pages 2,059--2,062.