EXPERIMENTS IN NON-FACTOID QUESTION ANSWERING

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

SAMEER KULKARNI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

August 2013

Dr. Carolyn Crouch

Acknowledgements

I would like to thank my advisor Dr. Carolyn Crouch for her support and constant mentoring. I am grateful to her for providing financial support over the long summers allowing me to concentrate on research. I would also like to thank her for being extremely flexible and accommodating about my defense schedule.

I would like to thank Dr. Ted Pedersen for his support. This work would not have been possible without the papers that he gave me for reading and his time-to-time suggestions.

I am thankful to the team at the International Computer Science Institute in Berkeley, California for giving me access to the FrameNet data. This work would never have started in the first place without this data. I would also like to thank Henrik De Smet for providing the Yahoo! training corpus.

I would also like to thank Dr. Chris Prince for teaching and guiding me through the Operating Systems course. I would also like to thank Dr. Joe Gallian for being an excellent teacher and for agreeing to be on my thesis defense committee.

Last, but not the least, I am grateful to the UMD Graduate School and the Director of Graduate Studies, Dr. Carolyn Crouch, for the financial support provided to me in support of my research through the Quality Metrics Allocation/Chancellor's Graduate Fellowships.

Abstract

Question Answering (QA) is the task of generating or extracting an answer for a user query from a corpus of documents. Factoid QA is the most popular and studied form of QA and has received maximum focus from the scientific community. As is apparent from the name, the information requested from these factoid questions is a bare fact and in most cases is a named entity. In a majority of cases, such information is found in a single document and does not require sentence extraction and sentence reordering.

However, most interesting questions are not factoid questions. Users might request a summary of a recent event from a news article, or they might want to know about a recent remedial cure for some observed symptoms that require text extraction from five different medical documents. All such queries require sentence extraction from a single or (often) multiple documents and require sentence reordering to generate a readable answer. This task is non-trivial, and hence there is more to non-factoid QA than meets the eye. Non-factoid QA has recently drawn attention from both the Information Retrieval (IR) and Natural Language Processing (NLP) communities, but most of the research has focused on developing learning models for re-ranking the answers from a set of question-answer pairs.

This thesis explores the use of different natural language (NL) structures to complement the traditional bag-of-words model to generate answers for non-factoid questions. We find that complex linguistic features like semantic role labels outperform

the traditional bag-of-words model. In fact, we find that the combination of different NL structures with the bag-of-words model performs best in our experiments.

We also use Feature Engineering for extracting different sets of features from a given corpus. We find that using similarity features, translation features and occurrence features produces a higher ranked result as compared to the bag-of-words model and may help bridge the semantic gap between non-factoid questions and answers.

Contents

List of Tables

List of Figures

1 Introduction

Information Retrieval (IR) is an extremely broad field that focuses on the storage, analysis and retrieval of different forms of data. Text Retrieval has been the most widely researched and studied domain in this field. Over a period of years, IR has focused on retrieving documents in response to the users' natural language queries. The advent of the Web and rapid development of search engines has marked a new dimension in document retrieval. With this deluge of information, precision in returning the exact information requested by a user has received a lot of attention from the IR community [11, 30, 4, 16].

In response to this information explosion, some user requests have become more specific, and these users want precise information. A typical response is one or more documents, ranked with respect to correlation with the perceived query. But the user may prefer a direct answer to his query. In the case of factoid questions, such answers are simple facts that are usually some kind of named entities. In the case of non-factoid questions, the answers are more complex and might require some sort of sentence reordering to make it readable. Question Answering (QA) is the name given to the task of providing information that directly answers a question.

QA has received significant attention in past years. However, the majority of

the academic and industrial research in QA has focused on factoid questions. QA tracks at conferences such as the Text REtrieval Conference (TREC) [37] required the participants to focus on factoid QA. Answers to factoid questions are often short answers that demand a factual response, e.g., location, time, distance, monetary amount, etc. Non-factoid questions received less attention in the QA community in earlier years (1998-2001). Although many research groups have recently turned their attention to non-factoid QA, most have focused on re-ranking the answers in a hand-annotated corpus or question-answer pairs collected from public discussion forums and question answering websites like *Yahoo! Answers*. Little research has focused on compiling the training data required to build models which can learn to extract answers, making the use of machine learning techniques difficult.

Supervised Machine Learning can be effectively used in factoid QA as evidenced by systems such as IBM Watson [39]. The primary reason for such success is the availability of the training sets provided by TREC [37] through their QA tasks and other hand-annotated data that can be easily generated because the answers are specific and short. For non-factoid QA, its nearly impossible to extract patterns by learning from examples of question-answer pairs since there are infinitely many different types of questions, and we neither can write rules to handle every possible combination nor extract patterns from training examples because of the quality of the answers. As a consequence, non-factoid QA tends to be a much harder problem, since otherwise helpful machine learning techniques provide little help in this context.

This thesis concentrates on non-factoid QA, with a focus on the answer generation and sentence ordering phases. We aim to exploit the rich contextual information present in the basic sentence structure by considering different natural language representations of the original text. We consider both the syntactic and semantic information inherent in a natural language sentence. In particular, we concentrate upon predicate-argument relationships signified by semantic roles [2] in addition to the conventional bag-of-words model. The predicate-argument relationships help us understand the theme, actor and the instrument in a sentence. Semantic roles go further and help to extract meaning beyond syntactic representations [2].

As is usual in QA systems, we employ IR methods to reduce the search space to a document set of interest. For this purpose, we use Indri [33], a text search engine developed at the University of Massachusetts, Amherst. Indri is based on the language modeling approach and uses a combination of inference networks and language modeling. A primary reason for choosing Indri is its speed; it processes millions of documents within a few hours. (An entire collection containing approximately 4 million Wiki documents was indexed in under 5 hours using Indri. More insight into Indri and its usage in our QA system follows in Chapter 2.)

We also explore a set of features used previously by several state-of-the-art QA systems [11, 30, 4, 16]. We use features that deal with the extraction of different information for each feature set. We consider similarity features (SF) and translation features (TF). We use these feature sets to extract rich information from the bag-of-

words models as well as the semantic role labelled data sets.

This thesis is organized as follows. Background information required for information retrieval, question answering and other natural language processing techniques that we have used is described in Chapter 2. Our approach, other complex representations of the textual content, and different feature sets are covered in Chapter 3. An in-depth view of the types of features used and how each feature set extracts rich syntactic and semantic information from the available text is also presented. Chapter 4 contains a detailed explanation of our experiments along with an evaluation of results. All experiments have been evaluated using the INEX evaluation system [13] and ROUGE [18]. Chapter 5 covers some related work in the area of non-factoid question answering. Chapter 6 consists of the contributions made by this thesis and concludes with recommendations for further research.

2 Background

2.1 Information Retrieval

With the rapid development of the World Wide Web or the Internet, searching for information is now amongst the most widely performed activity in the world. Immensely successful search engines like Google have made information retrieval a very popular field and an area of active research from the last 30 years. Although IR as a field emerged in the early 1960s with the late Gerard Salton's work [28], experimental work was then focused largely on small static collections of documents. However, with the advent and exponential growth of the Internet, new issues arose. As a consequence, IR techniques have evolved to deal with large dynamic document collections. Document retrieval forms the backbone of our QA system, and the performance of the system depends on how effective the document retrieval is.

2.1.1 Vector Space Model

The Vector Space Model [28] is arguably the most popular document retrieval model and is based on the application of vector algebra to documents and queries. Both query and documents are represented as vectors. A similarity measure such as

cosine is used to measure the similarity between them. The result is a rank ordered set of documents. The importance of terms is often indicated by *tf-idf* weights. Thus, the Vector Space Model ranks documents according to their similarity to the query. However, this similarity may or may not correlate with relevance.

2.1.2 Probabilistic Model

Another view of the IR system is given by probabilistic models. These models are based on the Probabilistic Relevance Framework (PRF) developed for document retrieval. BM25, one of the most successful and best performing term weighting formulas, evolved from this view. The classic probabilistic model is based on estimating the answer to this question, as posed by Karen Sparck Jones, *et al.,"What is the probability that a user will judge this document relevant to the query?"* [31, p.6] This view estimates the probability of relevance and then ranks documents in descending order of this probability.

The BM25 formula is given below.

$$\mathbf{BM25(A)} = \sum_{\mathbf{i=0}}^{\mathbf{|Q|}} \frac{\mathbf{tf_i^A(k1+1)}}{\mathbf{k1((1-b)+b(\frac{|A|}{avg\_length}))+tf^A}} \mathbf{log(idf_i)}, \text{where}$$

$\mathbf{b = 0.75}$, $\mathbf{k1 = 1.2}$, $\mathbf{avg\_length}$ is the average length of documents in the collection, $\mathbf{tf}$ is the term frequency, $\mathbf{idf}$ is the inverse document frequency, and $\mathbf{|Q|}$ is the number of query terms in a given query.

BM25 has been implemented by Lemur, Okapi, Terrier, Wumpus and many other groups. Okapi BM25 is the version developed by Robertson, *et al.,* at the City University, London [27]. BM stands for "Best Match" and Okapi is the retrieval system that this group designed. BM25 [26] is only one of many formulas used by the Okapi system (e.g., BM1, BM11 and BM15), but BM25 has proven most successful.

2.1.3 Language Model

Language Modeling is a part of the probabilistic framework and emerged as a result of Claude Shannon's seminal work in Information Theory and usage of n-gram models to predict natural text.

In the recent years, an increasing number of people have applied Language Modeling methods to document retrieval and other text processing tasks. This approach is quite different from probabilistic approaches and has sparked interests in the IR and NLP community. In Language Modeling, we are trying to estimate the probability of a query being generated from the document model. Thus, when we use Language Modeling for document retrieval we are usually trying to answer the following question: *"Given a query and a document, what is the probability of that query being generated from that document?"* We are trying to estimate the value of P(q|d), where q is the query and d is a document belonging to the set of documents, D.

A language model can be thought of as a noisy channel [14] (as used in Speech Recognition and Machine Translation); it establishes the relation between documents

and the queries. To quote Ponte and Croft in the first work that explored the idea of Language Modeling [25, p.2]: *"When designing a statistical model for language processing tasks, often the most natural route is to apply a generative model which builds up the output step-by-step. Yet to be effective, such models need to liberally distribute probability mass over a huge space of possible outcomes. This probability can be difficult to control, making an accurate direct model of the distribution of interest difficult to construct. The source channel perspective suggests a different approach: turn the search problem around to predict the input. Far more than a simple application of Bayes' law, there are compelling reasons why reformulating the problem in this way should be rewarding. In speech recognition, natural language processing, and machine translation, researchers have time and again found that predicting what is already known (i.e., the query) from competing hypotheses can be easier than directly predicting all of the hypotheses."*

The method of estimating document language models or a document-to-query translation model has emerged as the "Language Modeling" approach and has presented a new dimension in Information Retrieval. This approach has been quite successful and is used in the Lemur toolkit [17]. Indri is a part of the Lemur toolkit and we have used it as our document retrieval system.

Chapter 3 gives a more detailed explanation of Indri and related information.

2.2 Question Answering

The task of returning a particular piece of information in response to the user's query is called ***Question Answering***. The task is termed ***factoid question answering*** if the answer is a simple fact such as a location or a date or a person. However, most interesting questions are not factoid questions, and hence we concentrate on ***non-factoid QA***. ***Focused Summarization*** is another term used for this task, where we try to summarize multiple documents for the purpose of generating an answer to a user's query. We use these two terms interchangeably as they are synonymous for the problem at hand.

2.2.1 Non-factoid Question Answering

Since our document retrieval system, Indri, produces the top-k documents, our task is to summarize these k documents to form an answer. As specified in Jurafsky and Martin [14, p.796-798], we perform three basic steps, i.e., ***content selection***, ***information ordering*** and ***sentence realization***.

Content Selection

Content Selection is an important step in non-factoid question answering or multi-document summarization, primarily due to the redundancy introduced when we deal with multiple documents. Sentences in multiple documents can be similar in terms of the overlap of words, phrases, etc. It seems obvious that content selection algorithms

must minimize redundancy when selecting sentences to form the final answer. To ensure minimal overlap between a new sentence and previously extracted sentences, we need a model that measures how similar the new sentence is to those already extracted. (See Chapter 3 for details.)

Information Ordering

Information ordering is the second stage of a multi-document summarization system. It is important to produce a summary with appropriate flow that is readable and coherent. Jurafsky and Martin mention that "*the most important factor for information ordering is* ***coherence***." [14, p.798] This is the stage where we order the extracted sentences to form a coherent answer. The original order cannot be used because of the coherency issue. In this work, we combine *content selection* and *information ordering* into a single step. (See Chapter 3 for details.)

Sentence Realization

Sentence realization is an optional step required only when the sentences resulting from the previous steps still have coherence problems. It is not employed here.

2.3 Machine Learning in IR and QA

Machine Learning refers to systems that predict the output for test data based on characteristic properties learned from training data. However, machine learning techniques are not generally helpful with respect to generating answers in non-factoid QA because they lack a human-annotated corpus. ***Supervised learning***, on the other hand, maybe useful because the training set consists of human-labelled data; the classifier learns from these examples and can then be used to classify test examples. We use supervised learning in the evaluation of generated answers or summaries to sense whether they are better than the human-annotated answers or if they need further improvement. We also use it in our ***Semantic Role Labeling System***, based on a Support Vector Machine package. (See Chapter 3 for technical system details and Chapter 4 for the use of supervised learning to evaluate our answers.) Supervised learning approaches of particular interest in this work are described below.

2.3.1 Support Vector Machines

The definition of a support vector machine (SVM) is specified in [6, p.353] as: *"A support vector machine directly computes the separating hyperplane that maximizes the margin or distance to the nearest example points. Several points will be at the same distance; these points are known as the support vectors and the resulting classifier is a linear combination of these vectors - all other points may be ignored."*

A SVM is a binary linear classifier which takes in a set of training data and

classifies each member of the set into one of the two categories. The training algorithm builds a model that can be used to classify the items from a test data set into one category or another.

Figure 2.1 [34] delineates the hyperplane and the support vectors in an SVM. As seen, the hyperplane tries to maximize the distance to the nearest sample points. Figure 2.2 [35] shows a three dimensional view of the SVM space and the hyperplane that separates different categories.

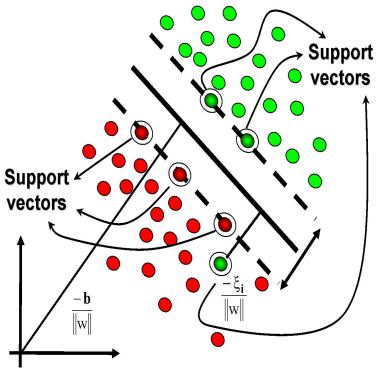

Figure 2.1: Support Vector Machine

2.3.2 Hidden Markov Models

The Hidden Markov model (HMM) is an extension of a Markov chain and both descend from a common parent, *Finite Automaton*. The input sequence in a Markov chain uniquely determines which states the automaton will go through. Every transition is associated with a probability which indicates how likely that path is to be taken. Thus, Markov chains can be used to assign probabilities to a sequence of

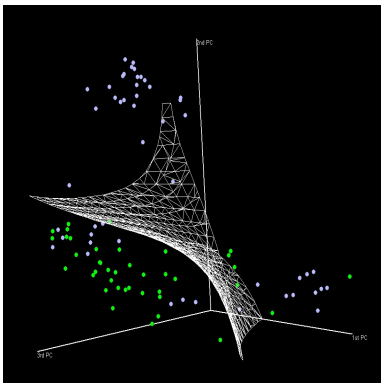

Figure 2.2: SVM Hyperplane Plot

events or to compute the probabilities for a sequence of events that are observable in the surrounding world. But many times the events of interest are not observable and hence are hidden. For example, semantic role labels are not observed in the world; we judge words and reckon what semantic role label, if any, can be applied to that word. To quote Jurafsky and Martin [14, p.177]: *"A* ***hidden Markov model*** *allows us to talk about both observed events and hidden events that we think of as causal factors in our probabilistic model."* Chapter 3 gives more detail on the use of hidden Markov models in our system.

3 Implementation

This thesis seeks to determine if the structural information present in natural language sentences can be used to generate better answers for non-factoid questions. We would like to show that natural language structures like semantic role labels provide more information than is usually available through the bag-of-words model.

The quality of the answers generated by any QA system is dependent on the corpus used. Most QA systems are extractive summarization systems, wherein specific sentences extracted from the text are combined in some order to form the final answer. Our system uses the same approach, i.e., it is an extractive summarization system and generates answers using a corpus of documents that correlate well with the query.

It is important to understand the architecture of the system before we delve deeper into the approach and implementation.

3.1 Architecture

Figure 3.1 portrays the architecture of our non-factoid QA system. All components are explained in the following sections.

The first component of the system is the *Document Retrieval* system. This system

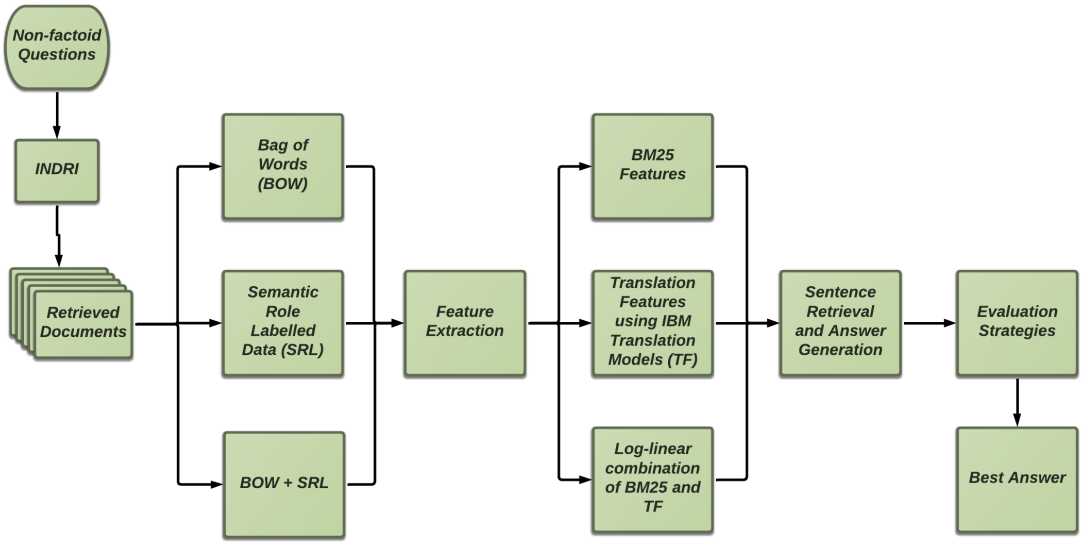

Figure 3.1: System Architecture

does not focus on query processing, and it is assumed that the non-factoid questions are well formed.

The output of our system depends entirely on how well the document retrieval system performs. We use ***Indri*** for document retrieval, and it serves as the foundation of the entire QA system.

3.2 Indri

Indri is a document retrieval system built by the Center for Intelligent Information Retrieval (CIIR) at the University of Massachusetts, Amherst and the Language Technologies Institute (LTI) at the Carnegie Mellon University as a part of a larger project called Lemur [33, 17].

Indri combines inference networks and language modeling. It provides a rich, structured query language for use with large document collections [12]. Its support for massive collections gets more robust and expands when it is installed across different nodes and implemented as a distributed search; it can easily be scaled to handle terabyte-size document collections.

Indri supports a myriad of document types including, but not restricted to, HTML, XML, PDF and UTF-8 encoded text. It also provides additional functionality allowing field retrieval in XML-type documents and passage retrieval that can be used for snippet generation.

Indri's retrieval commands can be invoked by specifying parameters over the command line or via parameter files. The input queries can be specified in a separate file. (Types of queries used here are discussed in Chapter 4.) Indri returns a specified number of documents that correlate with the query in decreasing rank order.

3.3 Semantic Role Labeling System

Semantic role labeling is a technique used to establish a link between word meanings and sentence meanings.Jurafsky and Martin [14, p.670] define semantic role labeling as *"the task of automatically finding the semantic roles for each predicate in a sentence."* Semantic role labeling has many applications in natural language processing and is usually applicable to any task that involves semantic analysis. It has been used for *Question Answering* and *Information Extraction* [8].

Supervised machine learning is a frequent alternative for implementing semantic role labeling and requires sufficient training data. FrameNet and PropBank are the most widely used datasets to this end. FrameNet has been used in this work. (A more precise and in depth explanation can be found in Section 3.3.1.)

Figure 3.2 shows the phases involved in developing a system for automatic semantic role labeling. State-of-the-art implementations of machine learning algorithms are readily available. We use Support Vector Machines by Thorsten Joachims [1, 38]. (See Section 3.3.2 for details.)

3.3.1 FrameNet

FrameNet [2] is a semantic role labeling project maintained by the International Computer Science Institute in Berkeley, California, which is building a lexical database of English by annotating how words are used in actual texts. It currently consists of more than 170,000 sentences that have been annotated with semantic roles, for

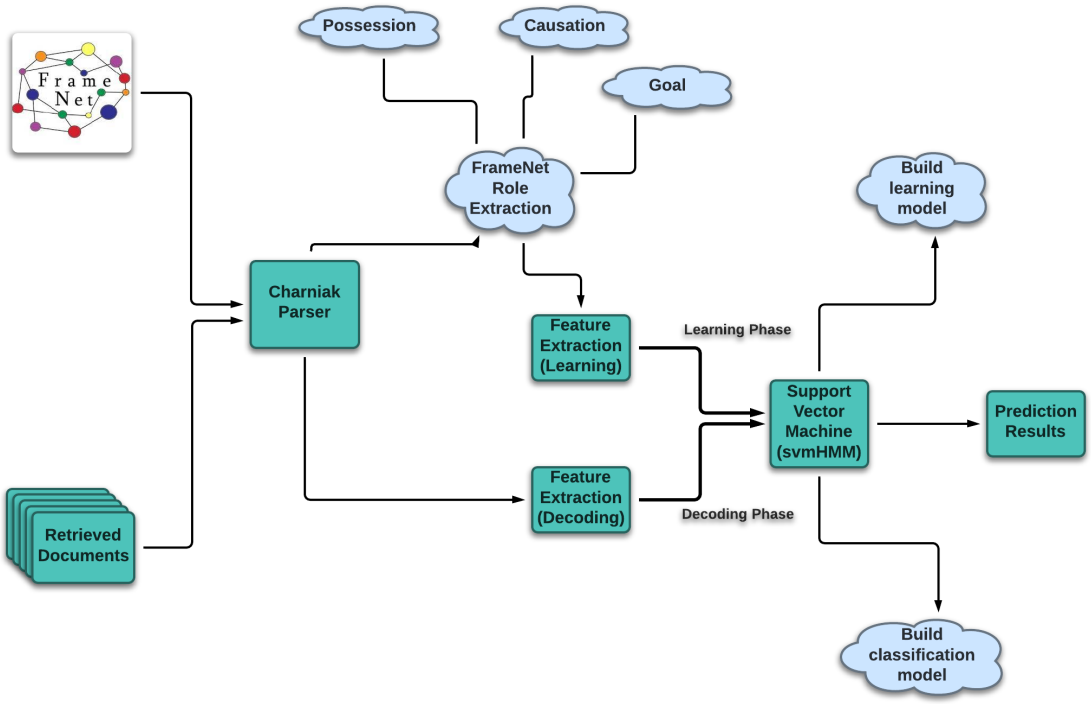

Figure 3.2: Semantic Role Labeling

example, *Activity_pause*, *Causation*, etc., and acts as a training dataset for semantic role labeling.

FrameNet [2] is based on a theory of meaning called *Frame Semantics*, which states that the meaning of a sentence can be better understood if the relationships between individual words can be identified based on a semantic frame. A semantic frame is a description of a type of event, relation, entity and the participants. Consider this example from the FrameNet database [2]:

Russia could [SUSPEND$_{Activity_pause}$]the[$DESTRUCTION_{Destroying}$]

In this example, *Activity_pause* and *Destroying* are the roles that have been assigned to suspend and destruction, respectively. The roles are usually classified into three categories: *Agent* or *Cause*, *Theme* and *Goal*. This is called a frame; the individual words with roles are called *frame elements (FEs)*. Words that evoke this frame are called *lexical units (LUs)*. FrameNet intends to define such frames and annotate sentences with the appropriate roles.

Semantic roles provide more information about the semantic composition of a sentence and can be used to identify sentence meanings from individual word meanings. These roles can be used to identify relationships between words in a sentence, eventually leading to a better semantic understanding of it. This is why semantic role labeling is a critical component in a non-factoid question answering system.

It is not sufficient to annotate sentences with appropriate semantic role labels. It is also important to establish relationships between the role labels of a query and a sentence from a correlated document. Different types of role-role relationships have been annotated in FrameNet [2] and are considered for linking query terms with the sentence terms. We classify the role-role relationships into two classes and a score is assigned for each class depending on how important the relation is. The classes are defined as follows:

Class A: inherits from, is inherited by, uses, is used by, subframe of, has subframes.

Score assigned: 0.75

Class B: perspective on, is perspectivized in, precedes, is preceded by, is inchoative of, is causative of.

Score assigned: 0.50

These scores have been adjusted to scale properly for the model at hand. These relationships allow us to evaluate how good a relation between a query term and a sentence term is and to appropriately weight that term.

3.3.2 Hidden Markov Support Vector Machines

The *Hidden Markov SVM package* (SVM-HMM) [36] uses hidden Markov models in combination with a support vector machine for sequence tagging tasks such as part-of-speech tagging and named-entity recognition. It is actually an implementation of structural SVM. It uses the training algorithm described in [1, 38]. It learns a hidden Markov model from training examples, which indicate the correct assignment of tags. The goal is to predict the correct tags for new sentences that come in as part of the testing data.

Since it is specifically used for sequence tagging, SVM-HMM is a natural choice for semantic role labeling. We use the FrameNet [2] dataset for training this SVM model; the learning model is then built from this training data. This data is used for learning the hidden Markov model. The number of iterations required and other parameters (like the trade-off between the training error and margin) must be set

appropriately before building the learning model. The set of documents retrieved by Indri for a given query acts as testing data, and the SVM learning model is used to predict semantic roles for the new sentences.

3.4 Feature Extraction

Feature extraction is an important step in any machine learning system. Feature extraction is part of *Feature Engineering*, and it is an effective method for filtering. Clarke, Buttcher and Cormack [6, p.338] define Feature Engineering as follows: *"the process of defining and extracting features likely to be useful to the classifier is called Feature Engineering and has a profound effect on overall filter effectiveness."*

The types and the number of features to be extracted depend on the application, and there is a significant difference between the types of features used for document retrieval and question answering. For non-factoid question answering, in particular, features should give more information about the surrounding words and sentence structure. This is important in order to identify relationships between different words, eventually moving us closer to understanding the semantic meaning of that sentence.

Feature extraction is critical because semantic role labeling depends on how the SVM is trained. The features selected for a word should provide information about surrounding words to train the SVM to achieve good precision. (The accuracy of the predicted results depends on how well the features have been selected.)

This work focuses on extracting the following 6 features for use in the SVM-HMM

package [36].

1. ***Semantic role label***: Each word may or may not have a semantic role label associated with it. If it does, the corresponding role label number is included as the feature value. If it does not, any large number can be included as the feature value. There are 1011 role labels in the FrameNet dataset. A word without a role label has 1012 as its feature value.
2. ***Part-of-Speech tag 1***: Charniak's parser [7] has been used to parse sentences and extract a part-of-speech tag for each word. The first word in the sentence has a feature value of 0. Charniak's parser also requires that each sentence begin with a start tag and end with an end tag. These tags have values of 0. All the other words will have the corresponding part-of-speech tag number as the feature value.
3. ***Part-of-Speech tag 2***: The part-of-speech tag for the current word; it uses the same feature values as those used for Part-of-Speech tag 1.
4. ***Part-of-Speech tag 3***: The part-of-speech tag for the next word; it uses the same feature values as those used for Part-of-Speech tag 1.
5. ***Verb***: A feature value indicating whether a given word is a verb (with feature value 1) or not (with feature value 0).
6. ***Noun***: A feature value indicating whether a given word is a noun (with feature value 1) or not (with feature value 0).

For this work, approximately 1.63 million features have been extracted as part of the training set; each feature belongs to one of the six categories listed above.

3.5 Features for Sentence Scoring

Sentence scoring is based on how well a sentence relates to a given query and should be represented as a quantitative value. Different feature sets, modeling the similarity between questions and answers, are required. The feature sets in this work portray two very different aspects of the relationship between a sentence and a query.

3.5.1 BM25 Similarity Features

The similarity between a sentence and a query is measured using the original BM25 formula of Robertson and Walker [6], as given below.

$$BM25(S) = \sum_{i=0}^{|Q|} \frac{tf_i^S(k1+1)}{k1((1-b)+b(\frac{|S|}{avg\_len}))+tf_i^S} \; log\_2(idf_i), \text{where}$$

1. S is a sentence to be scored
2. avg_len is the average length of a sentence in the set of retrieved documents
3. idf_i is the ratio of the total number of sentences in the set of retrieved documents for a given query to the total number of sentences containing that term
4. b and k1 are parameters tuned to the collection (with values 0.75 and 1.2, respectively, for this collection).

This formula is often used in document retrieval.

3.5.2 Translation Features using GIZA++

Sentence retrieval models based solely on the use of term frequency with similarity measures do not perform well in QA because they fail to bridge the semantic gap between the query and the sentence. This was first shown by Berger and Lafferty [3], giving rise to the use of *Statistical Machine Translation* in QA.

Similarity-based models fail largely because query terms are scarcely present in the sentence and similarity-based models are meant to work with large bodies of text, where term frequency is highly significant. However, in non-factoid QA, the textual structure shrinks from a document to a sentence. Even a long sentence usually has no more than 15-20 words. Hence, sentence retrieval models based solely on similarity is problematic for non-factoid QA.

The most intuitive way to address this problem is to use a translation model and learn question-to-answer transformations [3, 8, 20, 21]. IBM Translation Models have been used by many researchers to this end and have been successful in learning these transformations. The basic idea is to compute the probability of a sentence being a translation of the given query. We use IBM Model 4 [5], since it is simple and easy to accommodate within the existing system by including the translation probability.

We use the modified Dirichlet Prior Smoothing in order to compute the translation probability, where the answer is the source and the query is the target.

Modified Dirichlet Prior Smoothing:

$$Final\_Score = (1 - \lambda) Sentence\_Score(q|A)$$

$$Sentence\_Score(q|A) = \sum_{a \epsilon A} (T(q|A), \text{where}$$

1. $\lambda = 1$ - $\frac{|A|}{|A|+m}$
2. m is tuned to the collection (with value 0.75, for this collection)
3. a is a word in that sentence that is a part of the whole answer
4. A is a sentence and part of the answer.
5. T(q | A) is the probability of an answer A being a translation of a query q. This probability is calculated using the GIZA++ machine translation toolkit and IBM Model 4 as the translation model.

Computing such transformations addresses the traits of similarity-based models. But an important limitation of translation models needs to be addressed whenever it is used for sentence retrieval. Translation models do not set the probability of translating a word to itself to be high. However, for non-factoid QA, the presence of a query term in a sentence makes it a good candidate for becoming a part of the final answer. We address this limitation by discarding the original translation probability as assigned by the GIZA++ toolkit and assigning a translation score of 0.50 for that word. All the experiments that we conducted using the translation features indicate that setting a high value for this word score (translating it to itself) is important. (Chapter 4 details these results.)

3.5.3 GIZA++

GIZA++ is a statistical machine translation toolkit that is used to train the IBM Translation Models [1-5] and an HMM word alignment model [22]. This toolkit is used to generate the translation table required for computing the translation probabilities in the above formulas. Answer-query pairs are used for training the GIZA++ model. Every sentence from a highly correlated retrieved document has the same query recorded against it in order to align them and to get the word translation probabilities after the model is trained. We use the *Yahoo-based Contrastive Corpus of Questions and Answers (YCCQA)* [9] for training the GIZA++ toolkit.

Yahoo-based Contrastive Corpus of Questions and Answers (YCCQA) is a contrastive corpus of English, French, German, and Spanish, based on the questions and answers submitted by users of the *Yahoo! Answers* website. De Smet's description says *"It consists of question-answer interactions between internet users, produced*

under almost identical circumstances, for the four languages. The near-identical production contexts allow contrastive analysis, but the sub-corpora can also be used independently for language-specific research. The language represented in the corpus is characteristically informal and unmonitored, illustrating the casual writing style of internet postings." [9].

YCCQA consists of around 90,000 questions and 575,000 answers. Since we deal with English-English translation, we use only English questions and answers. This corpus has played an important part in the machine translation process; all the translation model based results have been based on this corpus.

3.6 Sentence Retrieval and Answer Generation

Answer generation, consisting primarily of sentence retrieval and sentence ordering, is the final step in the process.

3.6.1 Sentence Retrieval

Sentences are represented in two different formats before the features are extracted.

1. **Bag-of-Words:** This is the actual representation of the sentence, as it appears in a document, without the inclusion of additional information about the sentence structure or word relationships.
2. **Semantic Role Labeled Data:** This is the representation of the sentence created using the techniques described above, with words annotated with semantic role labels. This representation provides more information about the natural

structure of a sentence and also describes relationships between different words.

The BM25 similarity and translation features are then extracted for each of these representations. A total of 5 different scoring models are developed using a combination of the textual representations and the features. The models are as follows.

1. **BM25_BOW:** BM25 + bag-of-words text representation
2. **BM25_SRL:** BM25 + semantic role labeled text representation
3. **Translation_BOW:** Translation + bag-of-words text representation
4. **Translation_SRL:** Translation + semantic role labeled text representation
5. **BM25_SRL + Translation_SRL:** (BM25 + semantic role labeled text representation) + (translation + semantic role labeled text representation)

The results and detailed descriptions of the models are given in Chapter 4.

3.6.2 Sentence Ordering

There is little evidence that complex algorithms or deeper linguistic analysis help at the sentence retrieval stage [20, 21]. As a consequence, most algorithms are applied before the sentences are retrieved. A weighting scheme (such as those mentioned in Section 3.6) is then applied to score the sentences and create a sorted list. Sentence ordering is the last step before the final answer is generated.

The final task of sentence ordering requires specifying the number of words in the answer and INEX [13] has set the limit to 500 words. Generating a readable answer is challenging; sentence ordering is claimed to be an ***NP-complete problem*** [14]. Thus, finding a good approximation technique for solving this problem is important. This work generates different combinations of the top sentences to form a set of

answers and then uses ***ROUGE*** [18] for scoring these answers by comparing them with a gold standard answer. The gold standard answer is the reference answer that is part of the INEX evaluation toolkit. The top answer is then selected as the best answer.

We use a *"shuffle"* routine to create ten different representations of the same answer. This routine shuffles the sentences in a random order. Creating ten different answers increases the probability of getting an answer that is readable and closer to the gold standard. This routine has only been applied to the most promising model viz. ProTran as described in Chapter 4.

4 Experiments

This work uses Indri to select the top-ranked N documents that correlate with the query. Clearly, the sentence retrieval models and hence the entire QA system are dependent on these documents. The performance of any QA system that employs extractive summarization for generating answers and uses documents as the primary source is dependent on the document retrieval system that underlies it.

4.1 Evaluation

Most of this work is evaluated through INEX [13] and their evaluation systems. The INEX Tweet Contextualization track is a QA track and uses tweets instead of the normal queries. Wikipedia is used as the document collection and consists of well-structured documents. ROUGE [18] is used for the automatic evaluation of summaries. Participation in conferences like INEX allows researchers to compare their strategies and results with other participants.

INEX Tweet Contextualization evaluations are based on two important aspects of the submitted answer, viz., *1) how much information does the answer provide about the pertinent topic, i.e., informativeness*, and *2) how readable is the generated answer, i.e., readability.*

4.1.1 ROUGE

ROUGE [18] evaluates the extracted answers based on average precision, average recall and average F-measure. These measures are defined as:

1. Average Precision (AP): AP combines precision values at all possible recall levels [6].

$$AP = \frac{1}{|Rel|} * \sum_{i=1}^{|Res|} relevant(i) * P@i, \text{where}$$

Res is the set of retrieved sentences,

Rel is the set of relevant sentences, and

relevant(i) is 1 if the i-th sentence in Res is relevant ; 0 otherwise.

P@i is the precision at i-th sentence.

2. Average Recall (AR): Recall is the fraction of the sentences that are relevant to the query that are successfully retrieved. AR is the average across all the queries.

$$AR = \frac{\sum_{i=1}^{|Q|} \frac{|Res \cap Rel|}{|Rel|}}{|Q|}, \text{where}$$

Res is the set of retrieved sentences,

Rel is the set of relevant sentences, and

Q is the number of queries.

3. Average F-measure (AF): F-measure is the weighted harmonic mean of precision and recall.

$$AF = \frac{\sum_{i=1}^{|Q|} \frac{2 * precision * recall}{precision + recall}}{|Q|}, \text{where}$$

Q is the number of queries.

INEX 2012 experiments include runs that were officially submitted to and evaluated by INEX. 2013 runs were not submitted in time for evaluation by INEX and so were not included in the competition. 2013 runs were evaluated by the author (using the INEX evaluation toolkit and ROUGE).

4.2 INEX 2012 Experiments

This experiment uses cosine similarity for sentence scoring. Indri is used for document retrieval and the top ranked sentences are retrieved after applying cosine similarity. The cosine formula we use is:

$$numerator = \sum_{i=0}^{|Q|} Q_i * S_i$$

$$den_1 = \sqrt{\sum_{i=0}^{|Q|} Q_i * Q_i}$$

$$den_2 = \sqrt{\sum_{i=0}^{|Q|} S_i * S_i}$$

$$Sentence\_Score = \frac{numerator}{den_1 * den_2}, \text{where}$$

Q_i is the frequency of a query term i in a given query, and

S_i is the frequency of a query term i in a given sentence.

The algorithm for this model is as follows.

1. Retrieve the top-ranked N documents using Indri.
2. Run the sentence boundary detection procedure to identify individual sentences.
3. Apply the cosine similarity measure and score every sentence.

4. Sort the sentences in decreasing order.
5. Retrieve the top-ranked k sentences to form the final answer.

4.3 INEX 2013 Experiments

4.3.1 BM25 Model

BM25 is used for term weighting in this work, because it is easy to implement and works well for document retrieval. The BM25 formula is used in its original form, as specified in Chapter 3, but the terms in the formula have been reformulated and tuned to work with sentences (as a body of text) instead of a document. Chapter 3 describes the five models that are used for sentence retrieval. Two of these models are based on the BM25 formula.

4.3.2 BM25_BOW

This model uses a combination of the bag-of-words text representation and the BM25 model. The results produced by this model are moderately good and it is used here as a baseline.

The algorithm for this model is as follows.

1. Retrieve the top-ranked N documents using Indri.
2. Run the sentence boundary detection procedure to identify individual sentences.
3. Apply the BM25 sentence retrieval formula and score every sentence.
4. Sort the sentences in decreasing order.
5. Retrieve the top-ranked k sentences to form the final answer.

4.3.3 BM25_SRL

This model uses a combination of the semantic role-labeled text representation and the BM25 model modified for sentence retrieval. Semantic role labels are learned from the training data (See Chapter 3).

The algorithm for this model is as follows.

1. Retrieve the top-ranked N documents using Indri.
2. Run the sentence boundary detection procedure to identify individual sentences.
3. Using the FrameNet data learn the role labels for all the words.
4. Decode the role labels for sentence constituents from the set of retrieved documents.
5. Decode the role labels for the query as well.
6. Apply the BM25_SRL formula and score every sentence. The scoring formula adjusts itself according to the role label of that word and its relationship with the role label of a query. term. The BM25 formula used in BM25_BOW is now boosted with the role label scores.
7. Sort the sentences in decreasing order.
8. Retrieve the top-ranked k sentences to form the final answer.

4.3.4 Translation Model

We use translation models to bridge the lexical and semantic gap between the queries and answers. IBM Translation Models are the most widely used models [5] and include five models, with more translation and alignment information acquired in every model as we move from model 1 to model 5. The GIZA++ toolkit [22] provides

the most comprehensive package of IBM models 1 through 4 [5] and has been widely used in research communities for machine translation. GIZA++ generates many files when trained with a corpus amongst which the *alignment file*, *translation probabilities file*, and *perplexity file* are most important and useful for learning source word to target word translations. IBM Model 4 is the most advanced model that is available as part of GIZA++, and we use this model for learning source to target translations.

4.3.5 Translation_BOW

This model uses a combination of the bag-of-words text representation and IBM Model 4 to generate a weighting scheme for the sentences from the top-ranked retrieved documents.

The algorithm for this model is as follows.

1. Retrieve the top-ranked N documents using Indri.
2. Run the sentence boundary detection procedure to identify individual sentences.
3. Train the GIZA++ toolkit with the Yahoo! QA corpus and get the alignments, translation. probabilities and the source and target vocabularies.
4. Compute individual word scores using their translation probabilities.
5. Use the modified Dirichlet Prior Smoothing formula to compute sentence scores.
6. Sort the sentences in decreasing order.
7. Retrieve the top-ranked k sentences to form the final answer.

4.3.6 Translation_SRL

This model uses a combination of the semantic role-labelled text representation and IBM translation model 4 to score sentences and then retrieve the top-ranked k sentences to form an answer.

The algorithm for this model is as follows.

1. Retrieve the top-ranked N documents using Indri.
2. Run the sentence boundary detection procedure to identify individual sentences.
3. Using the FrameNet data, learn the role labels for all the words.
4. Decode the role labels for sentence constituents from the set of retrieved documents.
5. Decode the role labels for the query.
6. Train the GIZA++ toolkit with the Yahoo! QA corpus and get the alignments, translation probabilities and the source and target vocabularies.
7. Compute individual word scores using their translation probabilities.
8. Apply the Translation_SRL formula and score every sentence. The scoring formula adjusts itself according to the role label of that word and its relationship with the role label of the query term. The modified Dirichlet Prior Smoothing formula used in Translation_BOW is now boosted with the role label scores.
9. Sort the sentences in decreasing order.
10. Retrieve the top-ranked k sentences to form the final answer.

4.3.7 ProTran Model

This model is called ProTran because it combines a probabilistic model (BM25) and a translation model (IBM Model 4). This model uses the semantic role-labelled text representation and a linear combination of the scores generated using the BM25 model and IBM translation model 4. The top-ranked k sentences are then retrieved to form an answer. Thus, this model uses a combination of BM25_SRL and Translation_SRL. The algorithm is similar to the BM25_SRL and Translation_SRL models. This model uses the *"shuffle"* routine specified in Chapter 3 to create ten different representations of the same answer.

4.4 Results and Analysis

Results and analyses of these experiments are described in this chapter.

4.4.1 INEX 2012 Results

We submitted only one run at INEX 2012 for the Tweet Contextualization track.

- **Informativeness**

Table 4.1 shows the scores of participants at INEX 2012. Our model ranks 27 and for informativeness, the lower the score the higher the rank. This run did not involve any part-of-speech tagging or usage of any other natural language processing techniques.

- **Readability**

Table 4.2 shows the scores of top 10 participants at INEX 2012. We did not use any special technique for arranging sentences to form the final answer in the cosine model.

Rank	**Uni**	**Bi**	**Skip**
1	0.7734	0.8616	0.8623
2	0.7827	0.8713	0.8748
3	0.7901	0.8825	0.8848
4	0.7864	0.8868	0.8887
5	0.7959	0.8881	0.8904
6	0.7972	0.8917	0.8930
7	0.7909	0.8920	0.8938
8	0.8265	0.9129	0.9135
9	0.8380	0.9168	0.9187
10	0.8347	0.9210	0.9208
27 (UMD)	**0.9366**	**0.9913**	**0.9916**

Table 4.1: Informativeness Scores for Cosine Similarity Model at INEX 2012

Sentences were assigned scores and sorted to put the best sentences at the top of the list. The top-ranked k sentences were then chosen to form an answer whose length does not exceed 500 words.

The results confirm that using term frequency-based weighting schemes designed for large bodies of text are not useful in the context of single sentences.

Rank	Relevance	Syntax	Structure
1	0.6336	0.6087	0.5289
2	0.5966	0.5793	0.5433
3	0.6760	0.6529	0.5611
4	0.6975	0.6342	0.5703
5	0.7008	0.6676	0.5636
6	0.6760	0.6529	0.5611
7	0.5936	0.6049	0.5442
8	0.5966	0.5793	0.5433
9	0.6968	0.6161	0.5315
10	0.6336	0.6087	0.5289
27 (UMD)	**0.2933**	**0.2716**	**0.2278**

Table 4.2: Readability Scores for Cosine Similarity Model at INEX 2012

4.4.2 INEX 2013 Results

Table 4.3 shows the informativeness scores for BM25_BOW and BM25_SRL in comparison with the INEX 2013 Baseline. Table 4.4 shows the readability scores of the top 10 participants at INEX 2013, along with the scores for BM25_BOW and BM25_SRL. BM25_SRL answers are on average more readable than BM25_BOW.

Model	Average Precision	Average Recall	Average F-measure
BM25_BOW	**0.22003**	**0.15424**	**0.16583**
BM25_SRL	**0.18369**	**0.16558**	**0.15452**
INEX2013_BASELINE	0.12158	0.25383	0.15169

Table 4.3: ROUGE Scores for BM25 Models

Participant / Model	Mean Average (%)
1	72.44
2	72.13
3	71.71
4	71.35
5	69.54
BM25_SRL	**69.54**
6	67.46
7	65.97
BM25_BOW	**60.30**
8	49.72
9	46.72
10	44.17

Table 4.4: Readability Scores for BM25 Models at INEX 2013

The following conclusions can be drawn from Table 4.3 and Table 4.4.

1. ***Average Recall:*** Both BM25_BOW and BM25_SRL perform poorly in terms of average recall as compared with the INEX 2013 baseline, but BM25_SRL produces a higher value than BM25_BOW.
2. ***Average Precision:*** Both BM25_BOW and BM25_SRL perform better than the INEX 2013 baseline when average precision is considered. BM25_SRL ranks low compared to BM25_BOW.
3. ***Average F-measure:*** BM25_SRL and the INEX 2013 baseline have very close scores. BM25_BOW produces a higher value than these two models.
4. ***Readability:*** In terms of readability, BM25_SRL outperforms BM25_BOW.

Table 4.5 shows the informativeness scores for IBM_Model4_BOW and IBM_Model4_SRL in comparison with the INEX 2013 Baseline. Table 4.6 shows the readability scores of the top 10 participants at INEX 2013, along with the scores for IBM_Model4_BOW and IBM_Model4_SRL.

Model	**Average Precision**	**Average Recall**	**Average F-measure**
IBM_MODEL4_SRL	**0.15458**	**0.27165**	**0.16709**
IBM_MODEL4_BOW	**0.13474**	**0.28773**	**0.17197**
INEX2013_BASELINE	0.12158	0.25383	0.15169

Table 4.5: ROUGE Scores for IBM Model 4

Participant / Model	**Mean Average (%)**
1	72.44
2	72.13
3	71.71
4	71.35
5	69.54
6	67.46
7	65.97
IBM_Model4_SRL	**58.66**
IBM_Model4_BOW	**55.44**
8	49.72
9	46.72
10	44.17

Table 4.6: Readability Scores for IBM Model 4 at INEX 2013

The following conclusions can be drawn from Table 4.5 and Table 4.6.

1. ***Average Recall:*** Both IBM_Model4_SRL and IBM_Model4_BOW perform better than the INEX 2013 baseline in terms of average recall, with IBM_Model4_SRL slightly outperforming IBM_Model4_BOW.
2. ***Average Precision:*** Both IBM_Model4_SRL and IBM_Model4_BOW perform better than the INEX 2013 baseline when average precision is considered, with IBM_Model4_SRL performing better than IBM_Model4_BOW.

3. ***Average F-measure:*** IBM_Model4_BOW outperforms IBM_Model4_SRL and both exceed the INEX 2013 baseline.
4. ***Readability:*** IBM_Model4_SRL outperforms IBM_Model4_BOW.

Table 4.7 shows the readability scores of the top 10 participants at INEX 2013, along with the scores for the ProTran model.

Participant / Model	**Mean Average (%)**
1	72.44
2	72.13
3	71.71
ProTran_Model0	**71.57**
4	71.35
5	69.54
6	67.46
7	65.97
8	49.72
9	46.72
10	44.17

Table 4.7: Readability Scores for ProTran Model at INEX 2013

Table 4.8 shows the informativeness scores for ProTran in comparison with the INEX 2013 Baseline.

Model	**Average Precision**	**Average Recall**	**Average F-measure**
ProTran_Original	0.18385	0.26945	0.19888
ProTran_Model0	**0.17129**	**0.29648**	**0.20331**
ProTran_Model1	0.16883	0.29131	0.20012
ProTran_Model2	0.17009	0.29266	0.20139
ProTran_Model3	0.16875	0.29028	0.19999
ProTran_Model4	0.16910	0.29177	0.20042
ProTran_Model5	0.17091	0.29313	0.20205
ProTran_Model6	0.16909	0.29055	0.20006
ProTran_Model7	0.16991	0.29297	0.20142
ProTran_Model8	0.16864	0.29065	0.19980
ProTran_Model9	0.16718	0.28760	0.19792
ProTran_Model10	0.16766	0.28792	0.19832
INEX2013_BASELINE	0.12158	0.25383	0.15169

Table 4.8: ROUGE Scores for ProTran Model

The following conclusions can be drawn from Table 4.7 and Table 4.8.

1. ***Average Recall:*** ProTran_Model0 is the best performing model in terms of average recall, reporting a score in the range of 0.30, followed by ProTran_Model5. The INEX 2013 baseline gave the poorest performance with a value of 0.25383. Every ProTran model on an average gives a score of 0.28+ on recall.
2. ***Average Precision:*** ProTran_Model0 and ProTran_Model5 were the best performing models in terms of average precision. The INEX 2013 baseline gave the poorest performance with a value of 0.12158.
3. ***Average F-measure:*** Again, ProTran_Model0 and ProTran_Model5 were the best performing models among the 13 models in terms of average F-measure. The INEX 2013 baseline gave the poorest performance with a value of 0.15169.
4. ***Readability:*** Protran_Model0 produces a value of 71.57 and beats both the translation models and the BM25 models.

5 RelatedWork

Question Answering has received much attention from the scientific community in the last decade or so. There has been considerably lesser emphasis on non-factoid QA and this area has received attention only of late [4, 11, 30].

This thesis uses some of the ideas mentioned in [20]. Using statistical machine translation in QA dates back to 2001 and was first proposed in the work by Berger and Lafferty [3]. Using machine translation models for sentence retrieval originated in 2005 [21] and was used successfully for TREC tasks [37] in [20]. This work uses IBM Model 4 and the GIZA++ toolkit.

We were influenced by ideas in Ciaramita, *et al.* [8], especially exploring the inherent natural language information present in a sentence by using structures like semantic role labels. However, Ciaramita, *et al.* have worked on the aspect of answer re-ranking as opposed to answer generation. They emphasize learning ranking models for non-factoid questions from web collections. They use PropBank [23] based predicate-argument relationships, contrasting with our use of FrameNet [2] for the same.

Other related work includes Girju [10], which concentrates on automatically identifying causal relations for question answering. Bilotti, *et al.* [4], proved that predicate-argument frames constructed from the question and the expected answer types improve answer ranking.

6 Conclusions

6.1 Comparison of Models

The experiments and the results for each of the models described in Ch. 4 allow us to compare all the models that have been implemented. Consider the models that we have implemented as part of INEX 2013 experiments along with the INEX 2013 Baseline (see Table 6.1). The comparison is as follows.

1. ***Average Recall:*** ProTran_Model0 is the best performing model in terms of average recall. BM25_BOW is the worst model.

2. ***Average Precision:*** BM25_BOW is the best performing model in terms of average precision, whereas the INEX 2013 baseline is the worst.

3. ***Average F-measure:*** Again, ProTran_Model0 is the best performing model in terms of average F-measure, and the INEX 2013 baseline is the worst.

The following points can be concluded from the experiments and their results.

1. ***Combination of models leads to better scores:*** The linear combination of a probabilistic retrieval model and a translation model i.e., ProTran, produces higher scores in comparison with each working individually.

2. ***Use of Translation models boosts performance:*** Translation models like IBM Model 4 are very helpful in determining what source or query term

Model	Average Precision	Average Recall	Average F-measure
BM25_BOW	0.22003	0.15424	0.16583
BM25_SRL	0.18369	0.16558	0.15452
IBM_Model4_BOW	0.13474	0.28773	0.17197
IBM_Model4_SRL	0.15458	0.27165	0.16709
ProTran_Model0	0.17129	0.29648	0.20331
INEX2013_Baseline	0.12158	0.25383	0.15169

Table 6.1: Model Comparison

translates to what answer term. When used in combination with semantic role labels, these models produce better scores in terms of average precision and readability (see Table 6.1 and Table 4.6). Hence, semantic role labels can be useful for generating good quality answers.

3. ***Semantic role labels can be useful for raising the quality of an answer:*** Natural language structures like semantic role labels improve the quality of the retrieved sentences and hence the final answer because they make topic-related relations available. For example, consider

She [**child**] *was born about AD 460* [**Time**] [**Being_Born**]

Frames like *Being_Born* can be used for identifying relationships with the query frames. Thus, the inherent natural language information present within a sentence can be explored to improve the quality of the generated answer.

It is important to note that there is no data available for calculating statistics with respect to the significance testing and hence we have no claims relative to the significance of results.

6.2 Recommendations for Future Work

It is clear that the approach in this work is constrained by the recall of Indri. Use of a thesaurus as a query expansion method is a good solution for this problem.

Syntactic dependency chains can also be used as another representation of the textual content for creating one more set of natural language structures. Dependency chains capture the syntactic relationships between different words in a sentence and can complement the semantic role-labelled text representation, thus, further improving the quality of an answer [8].

PropBank [23] provides more lexical coverage than FrameNet [2] and hence can be used whenever the semantic frames cannot be recognized [24]. That way we can propagate back to the predicate-argument structure identified in the queries and documents [15]. For example, consider the lemma *fabric*. This lemma has two unique senses- a cloth and a framework (e.g. the fabric of society). Appropriate frames for this lemma cannot be determined using FrameNet. In such a case, PropBank's simple ARG0, ARG1 predicate-argument relationships can be used.

Another approach towards widening the scope could be to integrate a web crawler with the current system in order to fetch documents from over the web. Since we are limited by the types of semantic role labels in FrameNet, the quality of answers cannot be guaranteed to be superior, but as mentioned above, PropBank and FrameNet can be used as a combination to address that limitation. SemLink is a project that aims

to link together different lexical sources like PropBank, FrameNet, etc., via a set of mappings, which can be used to this end [29].

Sentence realization or sentence boundary detection is an issue while dealing with non-factoid QA and this is recommended for future work. For some queries with which we experimented, some retrieved sentences were not at all readable. Analysis of the retrieved sentences led us to conclude that these sentences were ill-formed because the sentence realization module that the system used was too weak and naive. Our current work focuses only on segmenting sentences using punctuation marks. However, it is important to consider exceptions such as abbreviations, usage of decimal points while specifying numbers, and salutations. Here we concentrate on using the inherent information in a sentence to improve sentence retrieval and hence the answer quality. Also, in our experiments, almost 70% of the time the punctuation mark approach leads to success, giving good readable sentences as output.

References

[1] Altun, Y., Tsochandiritis, I., Hoffman, T. Hidden Markov Support Vector Machines. *International Conference on Machine Learning (ICML)*, 2003.

[2] Baker, C., Fillmore, C., Lowe, J. The Berkeley FrameNet project. *COLING-ACL'98: Proceedings of the Conference*, 86-90, 1998.

[3] Berger, A., Lafferty, J. Information Retrieval as Statistical Translation. *SIGIR'99: Proceedings of the 22nd Annual International ACM SIGIR Conference*, New York, 222-229, 1999.

[4] Bilotti, M., Ogilvie, P., Callan, J., Nyberg, E. Structured retrieval for question answering. *Proceedings of the 30th Annual International ACM SIGIR Conference*, 351-358, 2007.

[5] Brown, P., Della Pietra, S., Della Pietra, V., Mercer, R. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2), 263-311, 1993.

[6] Buttcher, S., Clarke, C., Cormack, G., ed. *Information Retrieval: Implementing and Evaluating Search Engines*. MIT Press, 2010.

[7] Charniak, E. and Johnson, M. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. *Proceedings of the 43rd Annual Meeting of the ACL*, 173-180, 2005.

[8] Ciaramita, M., Zaragoza, H., Surdeanu, M. Learning to Rank Answers in Non-Factoid Questions from Web Collections. *Computational Linguistics*, 37(2), 2011.

[9] De Smet H. Constrastive Corpus of Questions and Answers. Department of Linguistics,University of Leuven, 2009.

[10] Girju R. Automatic detection of causal relations for question answering. *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, 76-83, 2003.

[11] Harabagiu, S., Moldovan, D., Pasca, M., Mihalcea, R., Surdeanu, M., Burnescu, R., Girju, R., Rus, V., Morarescu, P. Falcon: Boosting knowledge for answer engines. *Proceedings of the Text REtrieval Conference (TREC)*, 479-487, 2000.

[12] Basic use of the Indri Query Language [Internet]. Available from: http://sourceforge.net/p/lemur/wiki/Basic/

[13] INEX: Initiative for the Evaluation of XML Retrieval [Internet]. Available from: https://inex.mmci.uni-saarland.de

[14] Jurafsky, D., Martin, J, ed. *Speech and Language Processing*, Prentice Hall. 2009. 163-168, 765-811.

[15] Kaisser, M., Webber, B. Question Answering based on Semantic Roles. *DeepLP '07 Proceedings of the Workshop on Deep Linguistic Processing*, ACL, 41-48, 2005.

[16] Ko J., Mitamura, T., Nyberg, E. Language-independent probabilistic answer ranking for question answering. *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, 784-791, 2007.

[17] The Lemur Project [Internet]. Available from: www.lemurproject.org

[18] Lin C. ROUGE: a package for Automatic Evaluation of Summaries. *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, ACL, 2004.

[19] Metzler, D. and Croft, W. Combining the Language Model and Inference Network Approaches to Retrieval. *Information Processing and Management Special Issue on Bayesian Networks and Information Retrieval*, 40(5), 735-750, 2004.

[20] Murdock V. *Aspects of Sentence Retrieval* [dissertation].[Amherst,(MA)]: University of Massachusetts, 2006.

[21] Murdock V., Croft, W. A Translation Model for Sentence Retrieval. *HLT '05 Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, ACL, 684-691, 2005.

[22] Och, F. and Ney, H. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1), 19-51, 2004.

[23] Palmer, M., Gildea, D., Kingsbury, P. The Proposition Bank: A Corpus Annotated with Semantic Roles. *Computational Linguistics Journal*, 31(1), 2005.

[24] PropBank vs. FrameNet: Semantic role labeling training datasets [Internet]. Available from: http://en.wikipedia.org/wiki/PropBank

[25] Ponte, J. and Croft, W. A Language Modeling approach to Information Retrieval. *Proceedings of the Association of Computing Machinery Special Interest Group on Information Retrieval*, 275-281, 1998.

[26] Robertson, S. and Zaragoza, H. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3(4), 333-389, 2009.

[27] Robertson, S. and Walker, S. Okapi/keenbow at TREC-8. *Proceedings of the 8th Text REtrieval Conference*, 151-163, 1999.

[28] Salton G., Wong A., Yang, C.S. A vector space model for automatic indexing. *Communications of the ACM*, 18(11), 613-620, 1975.

[29] SemLink: Connecting PropBank, VerbNet, FrameNet, WordNet [Internet]. Available from: http://verbs.colorado.edu/semlink/

[30] Soricut, R. and Brill, E. Automatic question answering using the Web: Beyond the factoid. *Journal of Information Retrieval - Special Issue on Web Information Retrieval*, 9(2), 191-206, 2006.

[31] Sparck Jones, K., Walker, S., Robertson, S. A probabilistic model of information retrieval: Development and comparative experiments - Part 1. *Information Processing and Management*, 36(6), 779-808, 2000.

[32] Sparck Jones, K., Walker, S., Robertson, S. A probabilistic model of information retrieval: Development and comparative experiments - Part 2. *Information Processing and Management*, 36(6), 809-940, 2000.

[33] Strohman, T., Metzler, D., Turtle, H., Croft, W. Indri: A language model based search engine for complex queries. *Center for Intelligent Information Retrieval, Technical Report*, 2005.

[34] Support Vector Machine: Minimizing hyperplane [Internet]. Available from: http://www.cac.science.ru.nl/people/ustun/SVM.JPG

[35] Three dimensional Support Vector Machine: Minimizing hyperplane [Internet]. Available from: http://www.epicentersoftware.com/img/features/SVM_plot.jpg

[36] Joachims, T. Sequence Tagging with Structural Support Vector Machines: Hidden Markov Model Support Vector Machines [Internet]. Available from:http://www.cs.cornell.edu/People/tj/svm_light/svm_hmm.html

[37] TREC: Text REtrieval Conference [Internet]. To encourage research in information retrieval from large text collections. Available from: http://trec.nist.gov/

[38] Tsochandiritis, I., Joachims, T., Hoffman, T., Altun, Y. Large Margin Methods for Structured and Interdependent Output Variables. *Journal of Machine Learning Research (JMLR)*, 1453-1484, 2005.

[39] IBM Watson: Ushering in a new era of computing [Internet]. Available from: http://www-03.ibm.com/innovation/us/watson/