2011

Improvements to the complex question answering models

Imam, Md. Kaisar

Lethbridge, Alta. : University of Lethbridge, c2011

Improvements to the Complex Question Answering Models

Md. Kaisar Imam
Bachelor of Science in Computer Science and Information Technology, Islamic University of Technology (Bangladesh), 2005

A Thesis
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

I dedicate this thesis to my beloved parents whose endless support and inspiration has always been with me at each and every step of my life.

Abstract

In recent years the amount of information on the web has increased dramatically. As a result, it has become a challenge for the researchers to find effective ways that can help us query and extract meaning from these large repositories. Standard document search engines try to address the problem by presenting the users a ranked list of relevant documents. In most cases, this is not enough as the end-user has to go through the entire document to find out the answer he is looking for. Question answering, which is the retrieving of answers to natural language questions from a document collection, tries to remove the onus on the end-user by providing direct access to relevant information.

This thesis is concerned with open-domain complex question answering. Unlike simple questions, complex questions cannot be answered easily as they often require inferencing and synthesizing information from multiple documents. Hence, we considered the task of complex question answering as query-focused multi-document summarization. In this thesis, to improve complex question answering we experimented with both empirical and machine learning approaches. We extracted several features of different types (i.e. lexical, lexical semantic, syntactic and semantic) for each of the sentences in the document collection in order to measure its relevancy to the user query.

We have formulated the task of complex question answering using reinforcement framework, which to our best knowledge has not been applied for this task before and has the potential to improve itself by fine-tuning the feature weights from user feedback. We have also used unsupervised machine learning techniques (*random walk*, *manifold ranking*) and augmented semantic and syntactic information to improve them. Finally we experimented with question decomposition where instead of trying to find the answer of the complex question directly, we decomposed the complex question into a set of simple questions and synthesized the answers to get our final result.

Acknowledgments

I would like to express my profound gratitude to my supervisor Dr. Yllias Chali for his persistent and inspiring supervision and helping me learn the ABC of Natural Language Processing. It would not have been possible to complete this work without his encouragement, patience, suggestions and support.

I also thank my M.Sc. supervisory committee members Dr. John Zhang and Dr. Sajjad Zahir for their valuable suggestions and guidance. I thank Dr. Hadi Kharaghani for his kind consent to chair my thesis defense.

I am also thankful to all my fellow researchers and faculty members in the Department of Mathematics and Computer Science for their spontaneous cooperation and encouragement. Furthermore, I would like to express thanks to my friends Shafiq Rayhan Joty, Sadid Hasan for their friendship and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

In recent years we have witnessed an explosion of on-line unstructured information in multiple languages. The size of the publicly indexable World Wide Web (WWW) is at least 25.21 billion pages[1] as of March 2009 and as yet growth shows no sign of leveling off. This vast increase of both the amount of online data and the demand for access to different types of information has led researchers to a renewed interest in a broad range of Information Retrieval (IR) related areas, such as question answering, topic detection and tracking, summarization, multimedia retrieval, chemical and biological informatics, text structuring, text mining, etc.

Traditionally we use search engines like Google[2], Yahoo![3] or Bing[4] to retrieve information from this astronomical set of documents. These search engines offer access to billions of web documents covering virtually every topic of interest. But search engines have some limitations, which include uselessness of a sizable part of the output, limited support for specific information needs, restrictions on input syntax for query formulation, and coarse output granularity.

In response to a user query search engines frequently return hyperlinks to thousands of documents. As the majority of the web search engine users view 10 documents or less, more than 99% of the output is useless. Output granularity is another disadvantage of the search engines. Search engines return hyperlinks to full-length web documents, possibly

[1]http://www.worldwidewebsize.com/.
[2]http://www.google.ca/.
[3]http://ca.yahoo.com/.
[4]http://www.bing.com/.

accompanied by a document fragment in which the keywords are highlighted. To find relevant information, a user must read a few documents. Even if the relevant information is inside the first hit returned, finding it can still be time-consuming if the document is very long.

Question Answering (QA) systems try to deal with these problems. They present advanced user interface where users can write their query in natural language. In response, users are presented directly with a small set of short answers, which are easier to read and evaluate than a set of full-length web documents.

1.2 Text Summarization

Search engines provide a means to access huge volumes of information by retrieving the documents considered relevant to a user's query. But, the user still has to go through the entire document content to judge its relevance. This contributes towards a serious information overload problem. On the other hand, factoid question answering systems provide a short factoid answer phrase. But sometimes the user needs something in between these two extremes. Text summarization tries to fulfill these user needs by providing a short overview of a document or a set of documents.

There is no fixed definition of text summarization. According to Mani and Maybury (1999), "Text summarization is the process of distilling the most important information from a text to produce an abridged version for a particular task and user". According to Mani (2001), "The goal of summarization system is to take an information source, extract content from it, and present the most important content to the user in a condensed form and in a manner sensitive to the user's or application's need". Different summaries can be generated for the same input source depending on their functionality and usage (Mani, 2001). For example, one of the important factors is the compression rate which is the ratio of the summary length to the source length.

1.2.1 Types of Summary

Depending on the user need, there can be different kinds of summary. The goals of summarization can often be characterized by their position on two dimensions (Jurafsky and Martin, 2008):

- single-document verses multiple-document summarization
- generic summarization versus query-focused summarization

In single-document summarization systems, the input is a single document and the output summary can be used in situations like producing a headline or an outline. For multi-document summarization, the input is a group of documents. Multi-document summarization can be used for summarizing a series of news stories on the same event or when we have web contents on the same topic that we would like to synthesize and condense.

A generic summary is the one in which we do not consider a particular user or a particular information need; the summary simply gives the important information in the document(s). In contrast, in query-focused summarization a summary is produced in response to a user query. Query-focused summarization is also known as focused summarization, topic-based summarization or user-focused summarization (Mani, 2001).

The summarizing operations can be applied on elements such as words, phrases, clauses, sentences or discourse. Elements can be analyzed at various linguistic levels: morphological, syntactic, semantic and discourse/pragmatic. Based on the level of linguistic analysis of the source, summarization methods can be broadly classified into two approaches (Mani, 2001):

Shallow Approaches: These methods conduct surface level analysis of the document. They consider features, such as word count, presence of cue phrases, position of sentence etc., to extract the salient sentences and re-arrange them to form a coherent summary.

Deep Approaches: These methods perform deeper syntactic and semantic analysis of the document content to identify the salient portions. They require highly domain-specific information to be able to perform deeper analysis.

Also summaries can be generated by just copying and pasting the text from the source (extracts), or can be generated in abstractor's own words (abstracts).

1.3 Complex Question Answering

Question answering (QA) is the task of automatically answering a question posed in natural language. The answer produced by a QA system can be specific phrases, sentences, or short passages. QA is regarded as requiring more complex natural language processing (NLP) techniques than other types of information retrieval such as document retrieval, thus natural language search engines are sometimes regarded as the next step beyond current search engines (Kotov and Zhai, 2010).

QA research attempts to deal with a wide range of question types including: fact, list, definition, how, why, hypothetical, semantically-constrained, and cross-lingual questions. Some questions, which we call simple questions, are easier to answer. For example, the question "Who is the president of Bangladesh?" asks for a person's name. This type of question (i.e. factoid) requires small snippets of text as the answer. Again, the question "Which countries has Pope John Paul II visited?" is a sample of a list question asking only for a list of small snippets of text.

Complex questions often seek multiple different types of information simultaneously and do not presuppose that one single answer can meet all of its information needs. For example, with a factoid question like “What is the magnitude of the earthquake in Haiti?”, it can be safely assumed that the submitter of the question is looking for a number. However, with complex question like “How is Haiti affected by the earthquake?”, the wider focus of this question suggests that the submitter may not have a single or well-defined information need and therefore may be amenable to receiving additional supporting information that is relevant to some (as yet) undefined informational goal (Harabagiu et al., 2006). Complex question answering tasks require multi-document summarization through an aggregated search, or a faceted search, that represents an information need which cannot be answered by a single document. For example, if we look for the comparison of the average number of years between marriage and first birth for women in the USA, Asia, and Europe, the answer is likely contained in multiple documents. Complex question answering systems are useful for this type of query.

1.4 The State-of-the-Art Complex Question Answering Systems

1.4.1 Knowledge-Based Systems

Knowledge-based systems maintain a database of questions and corresponding answers. As answers are obtained beforehand and are used later, these systems are comparatively faster. Knowledge-based systems can be divided into two categories depending on how the answers are acquired.

Some of the knowledge-based systems act as information-sharing tools. These systems are a community-generated social knowledge Q&A platform which allows users to ask just about any question, such as “How to cook ramen?” or “How to subscribe to international magazines via the Internet?”, and gets answers from other users. One such system is *Naver’s Knowledge iN* which has been launched in 2002. As of January 2008 the Knowledge Search database included more than 80 million pages of user-generated information. After the success of Naver, Yahoo also lunched such system called *Yahoo! Answers* which is based on Naver. *Yahoo! Answers* is available in 12 languages and according to *Yahoo!* currently it has 200 million users worldwide and 15 million daily visitors[5].

In other types of knowledge-based systems answers are obtained automatically. One such system is *Watson*, a QA system built by IBM. Watson had access to 200 million pages of structured and unstructured content consuming four terabytes of disk storage, including encyclopedias, dictionaries, thesauri, newswire articles, and literary works. In 2011, as a test of its abilities, Watson competed on a quiz show called “Jeopardy!”, in the show’s only human-versus-machine match up-to-date. In a two-game, combined-point match, Watson bested Brad Rutter, the biggest all-time money winner on Jeopardy!, and Ken Jennings, the record holder for the longest championship streak.

Another such system is MAYA (Kim et al., 2001). It creates a database of answers before any questions are asked. There are only fifteen types of entities this system considers as answers. Each passage that contains a possible answer (i.e. any of the fifteen entities) is kept, and when a question is asked, the answer that is contained in the passages most closely related to the question is given as the answer. Katz et al. (2003) developed a similar method of question answering that uses knowledge bases to compile facts about every subject before any definition questions are asked. Clifton and Teahan (2004) built a knowledge base of questions from a document set. They used knowledgeable agents (Teahan, 2003) that are based on the knowledge grids proposed by (Cannataro and Talia, 2003). These

[5]http://en.wikipedia.org/wiki/Yahoo!_Answers.

knowledgeable agents go through the documents and form questions around entities they find. For instance, from the phrase, “John Lennon died on December 8th, 1980 during a public dramatic interpretation of J.D. Salingers Catcher in the Rye.” the system forms the question-answer pair, “When did John Lennon die?” and “December 8th, 1980”. When a question is asked, the system will check whether it has the knowledge to answer the question by determining which questions they have identified match the incoming question.

1.4.2 Complex Question Decomposition

One way to answer a complex question is to decompose the question into simple factoid questions. Harabagiu et al. (2006) have used this approach successfully in their complex question answering system (Harabagiu et al., 2006). The process they have used had three components:

- question decompositions (of the complex question)
- factoid question answering (QA) techniques (to process decomposed questions)
- multi-document summarization techniques (to fuse together the answers provided for each decomposed question).

According to them, question decomposition depends on the successive recognition (and exploitation) of the relations that exist between words and concepts extracted from topic-relevant sentences. For example, if a topic-relation r_1 between develop and drugs is recognized in question Q_0 , we assume that this sentence (and all other sentences containing this particular relation) will contain relevant information that can be used to decompose of Q_0. So they created a bipartite graph of relations established between concepts related to the topic of a complex question and subquestions. Complex questions are then decomposed by a procedure that operates on a Markov chain, by following a random walk on that bipartite graph. Decomposed questions discovered during this random walk are then submitted to a state-of-the-art Question Answering (QA) system in order to retrieve a set of passages that can later be merged into a comprehensive answer by a Multi-Document Summarization (MDS) system (Harabagiu et al., 2006).

1.4.3 Topic Focused Summarization

We can consider the task of complex question answering as topic focused summarization, where we consider the topic as a complex question. Most of the techniques that have been used for generic summarization can also be used for topic focused summarization. For example, the LexRank method discussed in (Erkan and Radev, 2004) was very successful in generic multi-document summarization. An extended version of the original LexRank method was introduced by (Otterbacher et al., 2005) for topic focused summarization. The following section gives a short description of different approaches for topic focused summarization.

Graph Based Methods

In recent years, a variety of graph-based methods have been proposed for topic-focused multi-document summarization (Wan et al., 2007a). The graph-based methods first construct a graph representing the sentence relationships at different granularities and then evaluate the topic-biased saliency of the sentences based on the graph.

For example, In the extended version of LexRank proposed by (Otterbacher et al., 2005), the set of sentences in a document cluster is represented as a graph, where nodes are sentences and links between the nodes are induced by a similarity relation between the sentences. Then the system ranked the sentences according to a random walk model defined in terms of both the inter-sentence similarities and the similarities of the sentences to the topic description or question. This idea is captured by the following mixture model:

$$p(s|q) = d \times \frac{rel(s|q)}{\sum_{z\in C} rel(z|q)} + (1-d) \times \sum_{v\in C} \frac{sim(s,v)}{\sum_{z\in C} sim(z,v)} \times p(v|q), \tag{1.1}$$

where, $p(s|q)$ is the score of a sentence s given a question q, is determined as the sum of its relevance to the question and the similarity to the other sentences in the collection. C is the set of all sentences in the collection. Parameter d, called bias, is a trade-off between two terms in the equation and is set empirically. Higher values of d, prefer the relevance to the question to the similarity to the other sentences.

The relevance of a sentence s to the question q is computed by:

$$rel(s|q) = \sum_{w\in q} \log tf_{w,s} + 1 \times \log tf_{w,q} + 1 \times idf_w, \tag{1.2}$$

where, $tf_{w,s}$ and $tf_{w,q}$ are the number of times word w appears in s and q, respectively and idf_w is the Inverse Document Frequency (IDF) of word w.

The system measures the cosine similarity weighted by word IDFs as the similarity between two sentences in a cluster:

$$sim(x,y) = \frac{\sum_{w\in x,y} tf_{w,x} \times tf_{w,y} \times (idf_w)^2}{\sqrt{\sum_{x_i\in x}(tf_{x_i,x} \times idf_{x_i})^2} \times \sqrt{\sum_{y_i\in y}(tf_{y_i,y} \times idf_{y_i})^2}} \tag{1.3}$$

Another universal graph based ranking method is the manifold-ranking method (Zhou et al., 2003a;b), which has been used for topic-focused document summarization (Wan et al., 2007a). The prior assumption of manifold-ranking is: (1) nearby points are likely to have the same ranking scores; (2) points on the same structure (typically referred to as a cluster or a manifold) are likely to have the same ranking scores.

In generic summarization, The sentence relationships are treated as a single modality. For topic focused summarization the sentence relationships are classified into within-document relationships and cross-document relationships, and each kind of relationships is considered as a separate modality (graph). They have used multi-modality learning algorithm for fusing the two modalities.

Summarization Based on Statistical Models

In the 1990s, with the advent of machine learning techniques in NLP, a series of seminal publications appeared that employed statistical techniques to produce document extracts. Both supervised and unsupervised models have been used to improve extractive summary.

Naive-Bayes Methods Kupiec et al. (1995) described a method derived from (Edmundson, 1969) that is able to learn from data. The classification function categorizes each sentence as worthy of extraction or not, using a naive-Bayes classifier. Let s be a particular sentence, S the set of sentences that make up the summary, and $F_1, F_2, ..., F_k$ the features. Assuming independence of the features:

$$P(s \in S | F_1, F_2, ..., F_k) = \frac{\prod_{i=1}^{k} P(F_i | s \in S) \times P(s \in S)}{\prod_{i=1}^{k} P(F_i)} \tag{1.4}$$

The features were compliant to (Edmundson, 1969), but additionally included the sentence length and the presence of uppercase words. Each sentence was given a score according to Equation 1.4, and only n top sentences were extracted.

Aone et al. (1999) also incorporated a naive-Bayes classifier, but with richer features. They described a system called DimSum that made use of features like term frequency (tf) and inverse document frequency (idf) to derive signature words.

Hidden Markov Models Conroy and Oleary (2001) modeled the problem of extracting a sentence from a document using a hidden Markov model (HMM). The basic motivation for using a sequential model is to account for local dependencies between sentences. Only three features were used: the position of a sentence in the document (built into the state structure of the HMM), the number of terms in the sentence, and the likeliness of the sentence terms given the document terms.

Log-Linear Models Osborne (2002) claimed that existing approaches to summarization have always assumed feature independence. The author used log-linear models to obviate this assumption and showed empirically that the system produced better extracts than a naive-Bayes model, with a prior appended to both models. Let c be a label and s be the item, we are interested in labeling, f_i the i^{th} feature, and λ_i the corresponding feature weight. The conditional log-linear model used by (Osborne, 2002) can be stated as follows:

$$P(c|s) = \frac{1}{Z(s)} \exp(\sum_i \lambda_i f_i(c,s)) \tag{1.5}$$

1.5 Related Works

Researchers all over the world working on multi-document summarization are trying different directions to see which methods provide the best results. In recent years, researchers have become more interested in topic-focused summarization and hence, different methods have been proposed ranging from heuristic extensions of generic summarization schemes[6] (by incorporating topic-biased information) to novel ones. For instance Nastase (2008) expands the query by using encyclopedic knowledge in Wikipedia and use the topic expanded words with activated nodes in the graph to produce an extractive summary.

The graph-based methods, such as LexRank (Erkan and Radev, 2004) and TextRank (Mihalcea and Tarau, 2004), are applied successfully to generic, multi-document summarization. The *LexRank* method addressed in (Erkan and Radev, 2004) is very successful in this problem domain. A topic-sensitive LexRank is proposed in (Otterbacher et al., 2005). In LexRank, the set of sentences in a document cluster is represented as a graph, where nodes are sentences and links between the nodes are induced by a similarity relation between the sentences. Then the system ranks the sentences according to a random walk model defined in terms of both the inter-sentence similarities and the similarities of the sentences to the topic description or question. A summarization method based on lexical chains is described in (Li et al., 2007) that utilizes the extraction of nouns, compound nouns and named entities as candidate words to represent lexical cohesion by incorporating their semantic similarities.

Another graph-based method proposed by Wan et al. (2007b) is called manifold-ranking. It make uniform use of sentence-to-sentence and sentence-to-topic relationships whereas the use of multi-modality manifold-ranking algorithm is shown in (Wan and Xiao, 2009). However, these methods use the standard cosine similarity measure to compute the relatedness between the sentences ignoring the syntactic and semantic information. The importance of syntactic and semantic features in finding textual similarity is described by Zhang and Lee (2003b), Moschitti et al. (2007), and Moschitti and Basili (2006). An effective way to integrate syntactic and semantic structures in machine learning algorithms is the

[6]Related work can be found in recent DUC workshop proceedings.

use of *tree kernel* functions (Collins and Duffy, 2001) which has been successfully applied to question classification (Zhang and Lee, 2003b, Moschitti and Basili, 2006). we use the tree kernel functions and to the best of our knowledge, no study has used tree kernel functions to encode syntactic/semantic information for more complex tasks such as computing the relatedness between the sentences in the multi-modality manifold ranking algorithm for topic-focused multi-document summarization.

Machine learning methods have also been employed to extract sentences. Single document summarization systems using Support Vector Machines (SVMs) demonstrated good performance for both Japanese (Hirao et al., 2002a) and English documents (Hirao et al., 2002b). Hirao et al. (2003) showed the effectiveness of their multiple document summarization system employing SVMs for sentence extraction. The motivation of applying Conditional Random Field (CRF) in text summarization came from observations on how humans summarize a document by posing the problem as a sequence labeling problem (Shen et al., 2007).

A new paradigm has been introduced in (Harabagiu et al., 2006) for producing summary-length answers to complex questions that relies on a combination of (a) question decompositions; (b) factoid QA techniques; and (c) Multi-Document Summarization (MDS) techniques. Their method operates on a Markov chain, by a random walk with mixture model on a bipartite graph of relations established between concepts related to the topic of a complex question and subquestions derived from topic-relevant passages that manifest these relations. Decomposed questions are then submitted to a state-of-the-art QA system in order to retrieve a set of passages that can later be merged into a comprehensive answer by a MDS system. They show that question decompositions using this method can significantly enhance the relevance and comprehensiveness of summary-length answers to complex questions. Inspired by this, we propose to augment the TAC ontologies (that provides a better coverage of the topic on the entire document collection) into a random walk framework that no study has used before to the best of our knowledge.

New features such as topic signature are used in NeATS system by (Lin and Hovy, 2002) to select important content from a set of documents about some topic to present them in coherent order. An enhanced discourse-based summarization framework by rhetorical parsing tuning is proposed by (Marcu, 1998). we exploit topic signature and rhetorical structure theory (Mann and Thompson, 1988) to weight the sentences.

In the natural language processing area, reinforcement learning has been extensively applied previously to the problem of dialogue management where the systems converse with a human user by taking actions that emit natural language utterances (Scheffler and Young, 2002, Roy et al., 2000, Litman et al., 2000, Singh et al., 1999). The state space defined in these systems encodes information about the goals of the user and what they say at each time step. There, the learning problem is to find an optimal policy that maps states to actions, through a trial-and-error process of repeated interaction with the user. An iterative reinforcement approach was proposed by (Wan et al., 2007a) to simultaneously extract summary and keywords from single document under the assumption that the summary and keywords of a document can be mutually boosted. Branavan et al. (2009) presented a reinforcement learning approach for mapping natural language instructions to sequences of executable actions.

1.6 Our Approaches

In our research, we have experimented with four different methods for answering complex questions. In the first approach we have formulated the task of complex question answering using Reinforcement Learning. We considered the main task of Document Understanding Conferences (DUC) 2007[7] for this experiment and we used DUC 2006 corpus as training

[7]http://duc.nist.gov/guidelines/2007.html.

data and DUC 2007 corpus as test data. The task was formulated in both *markov decision processes (MDP)* and *partially observable markov decision process (POMDP)*. We have used function approximation and used gradient descent to learn the parameters.

In the next approach we considered the Text Analysis Conference (TAC) 2010[8] guided summarization task. The guided summarization task is to write a 100-word summary of a set of 10 newswire articles for a given topic, where the topic falls into a predefined category. In this task as the category of the topics are predifined we used this information to score document sentences. Finally we used graph based *random walk* model for selecting sentences.

In the next experiment we used *manifold ranking* which has been used successfully for multi-document summarization. We extensively studied the impact of syntactic and semantic information in computing the similarity between the sentences in the multi-modality manifold learning framework for complex question answering.

In our final approach we used question decomposition framework for answering complex question. This framework has five main components

- Sentence simplifier
- Question generator
- Question ranking
- Simple question answering system
- Answer synthesizer module

1.7 Thesis Outline

The remaining chapters of this thesis are organized as follows:

[8]http://www.nist.gov/tac/2010/.

Chapter 2 is a review of the mathematical tools we have used for complex question answering in this thesis. In section 2.1 we discussed about reinforcement learning, MDP and POMDP models and their formulation. Section 2.2 is a discussion of Supervised Models (SVM, MaxEnt).

Chapter 3 is a discussion of various methods for tagging useful information in the documents. In this chapter we provided a detailed description of the taggers and parsers we have used for processing the documents.

Chapter 4 includes different summary evaluation techniques that we have used to evaluate our complex question answering system and to compare them with other existing systems.

Chapter 5 includes the implementation details of our approaches to answer complex questions. In section 5.3 we described how the complex question answering can be formulated using reinforcement learning. In section 5.4 we discussed guided complex question answering technique using random walk. Section 5.5 gives detailed description of a graph based approach for complex question answering. Section 5.6 shows our approach for answering complex question by question decomposition.

Chapter 6 consists of concluding remarks about our findings and our views on the future of QA systems.

1.8 Published Work

Some of the material presented in this thesis has been previously published. Section 5.3, 5.4 and 5.5 expands on the materials published in (Chali, Hasan, and Imam, 2011a) and (Chali, Hasan, and Imam, 2011b).

Chapter 2

Machine Learning Techniques for Text Summarization

2.1 Reinforcement Learning

Learning form interaction is a fundamental idea underlying nearly all theories of learning and intelligence. An infant learns by interacting with its environment and observing the feedback it gets from the environment. This kind of feedback is called a reward or reinforcement. The idea has mainly arisen from the animal psychologists who have been carefully studying reinforcement learning for over 70 years. The term reinforcement learning seems to have come into use in AI through early works of Minsky (Minsky and Selfridge, 1961). Reinforcement learning has been studied as function optimization (McMurtry, 1970, Holland, 1992), which includes the study of hill climbing algorithms (Howland et al., 1960), as learning automata theory (Narendra and Thathachar, 1974) and in regard to the two armed bandit problem (Cover and Hellman, 1970).

2.1.1 Reinforcement Learning Framework

In reinforcement learning, an agent is placed in an environment and must learn to behave successfully by interacting with the environment. Here the agent is the learner or decision maker and the environment is comprising everything outside the agent with which it interacts. The learning process goes through interaction cycles as shown in Figure 2.1.

Elements of Reinforcement Learning A policy specifies what the agent should do for any state that the agent might reach. It is a mapping from perceived states of the environment to actions to be taken when the agent is in those states. In simple cases the policy

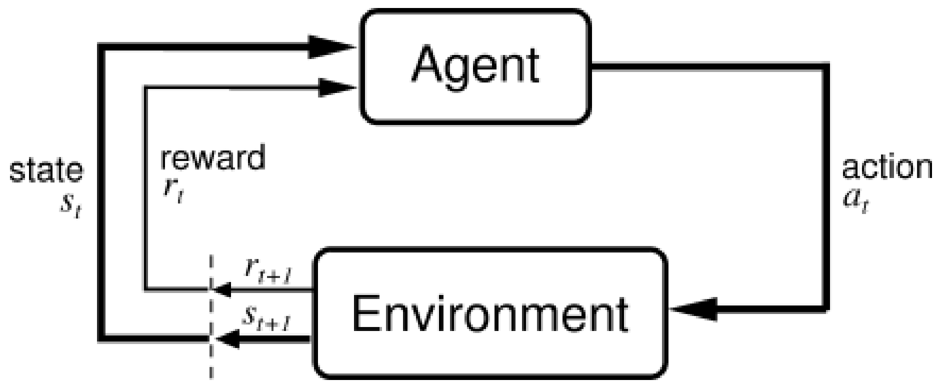

Figure 2.1: Reinforcement learning framework (Sutton and Barto, 1998)

may be represented by a simple function or a lookup table but it may also involve extensive computation such as search process. In general, policies may be stochastic and is sufficient to determine the behavior of the agent (Sutton and Barto, 1998).

A reward function defines the goal in a reinforcement learning problem. Roughly speaking, it maps each perceived state (or state-action pair) of the environment to a single number, a reward, indicating the intrinsic desirability of that state (Sutton and Barto, 1998). A reinforcement learning agent's sole objective is to maximize the total reward it receives in the long run. The reward function defines what are the good and bad events for the agent. Rewards are basically given directly by the environment so the reward function is unalterable by the agent. It may, however, serve as a basis for altering the policy. For example, if an action selected by the policy is followed by a low reward, then the policy may be changed to select some other action in that situation in the future. In general, reward functions may be stochastic.

The value/utility of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state. Whereas a reward function determine the immediate, intrinsic desirability of environmental states, a value function indicate the long-term desirability of states after taking into account the states that are likely to follow, and the rewards available in those states. For example, a state might always yield a low immediate reward but still have a high value because it is regularly followed by other states that yield high rewards. Or the reverse could be true.

Utility is estimated based on rewards. Utility plays a central role in choosing actions. The agent tries to choose its actions that bring about states of highest utility. Values must be estimated and reestimated from the sequences of observations the agent makes over its entire lifetime. In fact, the most important component of almost all reinforcement learning algorithms is a method for efficiently estimating values. If the agent's preferences between state sequences are stationary then there are just two ways to assign utilities to sequences (Russel and Norvig, 2003):

1. Additive rewards: the utility of a state sequence is

$$U_h([s_0,\ s_1,\ s_2,\ \ldots]) = R(s_0) + R(s_1) + R(s_2) + \ \ldots \tag{2.1}$$

where $[s_0,\ s_1,\ s_2,\ \ldots]$ is the state sequences visited by the agent and $R(s)$ is the reward at state s.

2. discounted rewards: the utility of a state sequence is

$$U_h([s_0,\ s_1,\ s_2,\ \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ \ldots \tag{2.2}$$

where the discount factor γ is a number between 0 and 1. The discount factor describes the preference of an agent for current reward over future rewards. When γ is close to 0, rewards in the distant future are viewed as insignificant. When γ is 1, discounted rewards are exactly equivalent to additive rewards, so additive rewards are a special case of discounted rewards.

The fourth and final element of some reinforcement learning systems is a model of the environment. This is something that mimics the behavior of the environment. For example, given a state and action, the model might predict the resultant next state and next reward.

Models are used for planning, by which we mean any way of deciding on a course of action by considering possible future situations before they are actually experienced. The incorporation of models and planning into reinforcement learning systems is a relatively new development. Early reinforcement learning systems were explicitly trial-and-error learners; what they did was viewed as almost the opposite of planning. Nevertheless, it gradually became clear that reinforcement learning methods are closely related to dynamic programming methods, which do use models, and that they in turn are closely related to state-space planning methods (Sutton and Barto, 1998).

Balancing exploration and exploitation The need to balance exploration and exploitation is a distinctive challenge that arises in reinforcement learning. If the agent chooses action that is best according to current knowledge it has, then we say that the agent is exploiting its knowledge. For example, If the agent maintains estimates of the action values, then at any time there is at least one action whose estimated value is greatest. The action with the highest estimate is called a greedy action.

If instead the agent selects one of the non-greedy actions, then we say it is exploring. Exploration is important as it enables the agent to improve its knowledge or in other words to improve its estimate of the non-greedy action's value. Exploitation is the right thing to do to maximize the expected reward on the one play, but exploration may produce the greater total reward in the long run (Sutton and Barto, 1998). Reward may be lower in the short run, during exploration, but higher in the long run because after the agent has discovered the better actions, it can exploit them. As it is not possible both to explore and to exploit with any single action selection, the agent must always do a trade-off between exploration and exploitation.

In any specific case, whether it is better to explore or exploit depends in a complex way on the precise values of the estimates, uncertainties, and the number of remaining plays.

There are many sophisticated methods for balancing exploration and exploitation. We have used ε-greedy method (Sutton and Barto, 1998).

ε-greedy method In ε-greedy methods the agent behaves greedily most of the time, but with small probability ε, instead it selects an action at random, uniformly, independently of the action-value estimates. An advantage of these methods is that, in the limit as the number of plays increases, every action will be sampled an infinite number of times.

Generalization and Function Approximation If the number of states and actions is large it is not possible to represent the agent's estimates of utility functions as a table with one entry for each state or for each state-action pair. The same problem occurs when the state or action spaces include continuous variables or complex sensations, such as a visual image. The problem is not just the memory needed for large tables, but the time and data needed to fill them accurately. In other words, the key issue is that of generalization. How can experience with a limited subset of the state space be usefully generalized to produce a good approximation over a much larger subset? Function approximation takes examples from a desired function (e.g., an utility function) and attempts to generalize from them to construct an approximation of the entire function (Sutton and Barto, 1998). Function approximation is an instance of supervised learning which has been extensively studied in machine learning, artificial neural networks, pattern recognition, statistical curve fitting, etc. So, to a large extent we only need to combine reinforcement learning methods with existing generalization methods.

2.1.2 Markov Decision Process

The specification of a sequential decision problem for a fully observable environment with Markovian transition model and additive rewards is called a Markov Decision Process (MDP) (Russel and Norvig, 2003). An MDP is defined by the following three components:

- Initial State: S_0
- Transition Model: $T(s, a, s^{'})$
- Reward Function: $R(s)$

Here the transition model is a specification of the outcome probabilities for each action in each possible state. In the case of MDP the transition model is Markovian. So the probability of reaching $s^{'}$ from s depends only on s and not on the history of earlier states.

To solve a MDP problem we have to find out an optimal policy that gives the maximum expected utility of the possible environment histories generated by the policy.

A careful balancing of risk and reward is a characteristic of MDPs that does not arise in deterministic search problems. Many real-world decision problems also share the same characteristic. That is why MDPs have been studied in several fields, including AI, operations research, economics, and control theory.

2.1.3 Partially Observable MDP (POMDP)

In the case of MDP the environment is fully observable but in the case of POMDP the environment is only partially observable. The agent does not necessarily know which state it is in. So the utility of a state s and the optimal action in s depend not on s, but also on how much the agent knows when it is in s. The elements of POMDP are:

- Initial State: S_0

- Transition Model: $T(s, a, s')$
- Reward Function: $R(s)$
- Observation Model: $O(s, o)$

The observation model specifies the probability of perceiving the observation o in a state s. In POMDP the agent always has to maintain its belief state. A belief state b is a probability distribution over all possible states. Equation 2.3 shows how new belief state $b'(s')$ is calculated given the previous belief state $b(s)$, action taken a and perceived observation o.

$$b'(s') = \alpha\, O(s', o) \sum_{s} T(s, a, s')\, b(s), \tag{2.3}$$

where α is a normalizing constant that makes the belief state sum to 1.

In POMDP the optimal action depends only on the agent's current belief state as the agent does not know its actual state; all it knows is the belief state. The decision cycle of a POMDP agent is:

- Given the current belief state b, execute the optimal action a.
- Receive observation o.
- Update the current belief state.

It can be shown that solving a POMDP on a physical state space can be reduced to solving an MDP on the corresponding belief state space. Although POMDPs can be reduced to MDPs, the resulted MDP has a continuous (and usually high-dimensional) state space. As a result the algorithms used to solve MDPs can not be applied directly to such MDPs.

2.2 Supervised Models

2.2.1 Support Vector Machines (SVM)

SVM is a powerful methodology for solving machine learning problems introduced by Vapnik (Cortes and Vapnik, 1995) based on the Structural Risk Minimization principle. In the classification problem, the SVM classifier typically follows from the solution to a quadratic problem. SVM finds the separating hyperplane that has the maximum margin between the two classes in the case of binary classification.

Figure 2.2 shows the conceptual structure of SVM. Training samples each of which belongs either to positive or negative class can be denoted by:

$$(x_1, y_1), \ldots, (x_u, y_u), \; x_j \in R^n, \; y_j \in \{+1, -1\}. \tag{2.4}$$

Here, x_j is a feature vector of the j-th sample represented by an n dimensional vector; y_j is its class label and u is the number of the given training samples. SVM separates positive and negative examples by a hyperplane defined by:

$$w \cdot x + b = 0, \; w \in R^n, \; b \in R, \tag{2.5}$$

where "$\cdot$" stands for the inner product. In general, a hyperplane is not unique (Cortes and Vapnik, 1995). The SVM determines the optimal hyperplane by maximizing the margin. The margin is the distance between negative examples and positive examples; the distance between $w \cdot x + b = 1$ and $w \cdot x + b = -1$.

From Figure 2.2, we clearly see that the SVM on the left will generalize far better than that of the right as it has an optimally maximized margin between two classes of samples. The examples on $w \cdot x + b = \pm 1$ are called the Support Vectors which represent both

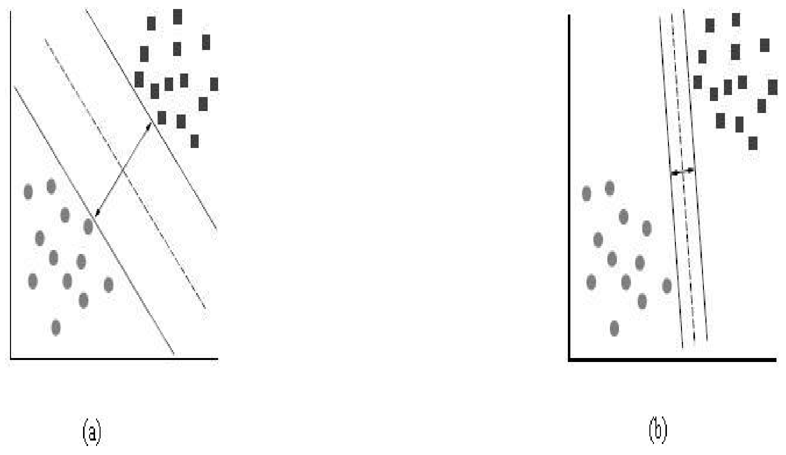
Figure 2.2: Support Vector Machines (Hasan, 2009)

positive or negative examples. The hyperplane must satisfy the following constraints:

$$y_i \left(w \cdot x_j \, + \, b \right) \, - \, 1 \geq 0. \tag{2.6}$$

Hence, the size of the margin is $2/||w||$. In order to maximize the margin, we assume the following objective function:

$$\begin{aligned} &Minimize_{w,b} \, J(w) = \frac{1}{2} \, ||w||^2 \\ &s.t. \; y_j \left(w \cdot x_j \, + \, b \right) \, - \, 1 \geq 0. \end{aligned} \tag{2.7}$$

By solving a quadratic programming problem, the decision function $f(x) = sgn(g(x))$ is derived, where

$$g(x) = \sum_{i=1}^{u} \lambda_i y_i x_i \cdot x \, + \, b. \tag{2.8}$$

When examples are not linearly separable, the SVM algorithm allows for the use of slack variables $\left(\xi_j\right)$ for all x_j to allow classification errors and the possibility to map examples to a (high-dimensional) feature space. These ξ_j give a misclassification error and should satisfy the following inequalities (Kudo and Matsumoto, 2001):

$$y_i\left(w \cdot x_j + b\right) - \left(1 - \xi_j\right) \geq 0. \tag{2.9}$$

Hence, we assume the following objective function to maximize the margin:

$$\begin{gathered} Minimize_{w,b,\xi}\, J(w, \xi) = \frac{1}{2}\,||w||^2 + C\sum_{j=1}^{u} \xi_j \\ s.t.\; y_j\left(w \cdot x_j + b\right) - \left(1 - \xi_j\right) \geq 0. \end{gathered} \tag{2.10}$$

Here, $||w||/2$ indicates the size of the margin, $\sum_{j=1}^{u} \xi_j$ indicates the penalty for misclassification, and C is the cost parameter that determines the trade-off for these two arguments. The decision function depends only on support vectors $(\lambda_i \neq 0)$. Training examples, except for support vectors $(\lambda_i = 0)$, have no influence on the decision function.

SVMs can handle non-linear decision surfaces with kernel function $K(x_i, x)$. Therefore, the decision function can be rewritten as follows:

$$g(x) = \sum_{i=1}^{u} \lambda_i y_i K(x_i, x) + b. \tag{2.11}$$

In this work, we use polynomial kernel functions, which have been found to be very effective in the study of other tasks in natural language processing (Joachims, 1998b, Kudo and Matsumoto, 2001):

$$K(x, y) = (x \cdot y + 1)^d. \tag{2.12}$$

2.2.2 *Maximum Entropy (MaxEnt)*

The maximum entropy approach is a novel method for the task of sentence extraction. The main principle of the MaxEnt method is to model all that is known and assumes nothing about that which is unknown. In other words, given a collection of facts, the model must be consistent with all the facts, but otherwise act as uniformly as possible (Berger et al., 1996). One advantage of this form of statistical inference is that we only constrain the model of our data by the information that we do know about the task, i.e. we do not assume anything about information of which we have no knowledge. Another advantage is that the information we use to constrain the model is in no way restricted so we can encode whatever linguistic information we want via the features. However, a disadvantage of this approach is that, although the maximum entropy approach may make good predictions, we cannot interpret the individual elements that cause the behavior of the system as a large number of features tend to be used in the approach and hence the output cannot be used to interpret all of these separately (Ferrier, 2011).

MaxEnt models can be termed as multinomial logistic regression if they are to classify the observations into more than two classes (Jurafsky and Martin, 2008). However, in this research, we used the MaxEnt model to classify the sentences into two classes: summary or non-summary. The parametric form for the maximum entropy model is as follows (Nigam et al., 1999):

$$P(c|s) = \frac{1}{Z(s)}\, exp\left(\sum_i \lambda_i f_i\right) \tag{2.13}$$

$$Z(s) = \sum_c exp\left(\sum_i \lambda_i f_i\right). \tag{2.14}$$

Here, c is the class label and s is the sentence we are interested in labeling. Z is the normalization factor that is just used to make the exponential into a true probability. Each f_i is a feature with the associated weight λ_i which can be determined by numerical optimization techniques in the absence of a closed form solution.

Chapter 3

Document Processing and Feature Extraction

Our summarization systems require different kinds of preprocessing of the documents. The raw data provided by DUC-2006 and DUC-2007 is just a string of characters. We had to preprocess the documents, including sentence tokenization, parts-of-speech tagging, syntactic parsing of the sentences etc. Some of the existing tools we have utilized to do tagging and parsing are:

- WordNet (http://wordnet.princeton.edu/)
- OAK System (http://nlp.cs.nyu.edu/oak/)
- Basic Element (BE) extractor (http://www.isi.edu/cyl/BE)
- Charniak Parser (http://www.cs.brown.edu/people/ec/software)
- ASSERT semantic role labeling system (http://cemantix.org/assert)

In this chapter, we give a detailed description of the different tags and parses that were done and the tools that were used to do the tagging and parsing.

3.1 Overview of Selected Tools

3.1.1 OAK System

OAK system (Sekine, 2002) is an English analyzer. It uses explicit rules that are extracted based on transformation or decision list learning methods. OAK system can be used as:

- Sentence Splitter
- Tokenizer

- POS tagger and Stemmer
- Chunker
- Named Entity tagger

3.1.2 WORDNET

WordNet is a lexical database of English. It groups English words (i.e. nouns, verbs, adjectives and adverbs) into sets of cognitive synonyms (synsets), each expressing a distinct concept. Each of these sets is called synsets. WordNet provides a general definition (i.e. gloss definition) for each synset and also provides various semantic relations between these synsets. Different senses of a word are in different synsets. For example, the noun *computer* has two senses and each sense belongs to a different synset as shown in the following table:

Synset ID	Words in the synset	Gloss definition
03082979	computer, computing machine, computing device, data processor, electronic computer, information processing system	a machine for performing calculations automatically
09887034	calculator, reckoner, figurer, estimator, computer	an expert at calculation (or at operating calculating machines)

Two kinds of relations are represented between the synsets: lexical and semantic. Lexical relations hold between semantically related word forms; semantic relations hold between word meanings. These relations include (but are not limited to) hypernymy/hyponymy (superordinate/subordinate), antonymy, entailment, and meronymy/holonymy.

Nouns and verbs are organized into hierarchies based on the hypernymy/hyponymy relation between synsets. Adjectives are arranged in clusters containing head synsets and satellite synsets.

3.1.3 ASSERT

ASSERT (Automatic Statistical SEmantic Role Tagger) is an automatic statistical semantic role tagger, that can annotate naturally occurring text with semantic arguments. ASSERT parser identifies all the predicates in a sentence, and then identifies and classifies sets of word sequences, that represent the arguments (i.e. semantic roles) of each of these predicates. In this process, it performs a full syntactic analysis of the sentence, automatically identifies all the verb predicates in that sentence, extracts features for all constituents in the parse tree relative to the predicate, and identifies and tags the constituents with the appropriate semantic arguments.

3.1.4 Lemur Toolkit 4.12

Lemur toolkit provides a set of NLP tools to support research and development of information retrieval and text mining software. For our research we have used Indri search engine that provides state-of-the-art text search and a rich structured query language. It supports text collections of up to 50 million documents (single machine) or 500 million documents (distributed search).

3.1.5 ROUGE

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation (Lin and Hovy, 2003) (Lin, 2004) is an automatic summarization evaluation metrics. ROUGE automatically determines the quality of a summary by measuring the number of N-gram overlapping units between the candidate and human generated summaries (Jurafsky and Martin, 2008). ROUGE toolkit calculates precision, recall and F-measure for evaluation. Precision (P) is the percentage of overlapping units in the candidate summary that are correct. Recall (R) is the percentage of over lapping units actually present in the input that were correctly identified by the candidate summary. The F-measure (Van Rijsbergen, 1979) provides a way to combine these two measures into a single metric. The F-measure is defined as:

$$F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

The parameter β differentially weights the importance of recall and precision. Values of $\beta > 1$ favor recall, while values of $\beta < 1$ favor precision.

Depending on how we choose the overlapping units there are different variants of ROUGE measures: ROUGE-N (N = 1, 2, 3, 4), ROUGE-L, ROUGE-W, ROUGE-S and ROUGE-SU.

ROUGE-N (N = 1, 2, 3, 4) is an n-gram recall between the candidate summary and the set of human reference summaries. ROUGE-N is computed as follows:

$$\text{ROUGE-N} = \frac{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} Count_{match}\,(gram_n)}{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} Count\,(gram_n)}$$

The function $Count_{match}(N - gram)$ returns the maximum number of N-grams that co-occur in the candidate summary and the set of reference summaries.

ROUGE-L measures the longest common subsequence between the reference and the candidate summaries. ROUGE-S measures the number of skip bigrams between the reference and candidate summaries. A skip bigram is a pair of words in their sentence order, but allowing for any number of other words to appear between the pair.

All the ROUGE measures in this thesis were calculated by running ROUGE-1.5.5 with stemming but no removal of stopwords. ROUGE run-time parameters were set as the same as DUC 2007 evaluation setup.

3.2 Corpus

For training and testing of our automatic summarization systems we used DUC-2006, DUC-2007 and TAC-2010 document sets. These documents come from the AQUAINT and AQUAINT-2 collections of news articles. The AQUAINT corpus of English News Text consists of documents taken from the New York Times, the Associated Press, and the Xinhua News Agency newswires. The collection spans the years 1999-2000 (1996-2000 for Xinhua documents). The AQUAINT-2 collection spans the time period of October 2004 - March 2006; articles are in English and come from a variety of sources including Agence France Presse, Central News Agency (Taiwan), Xinhua News Agency, Los Angeles Times-Washington Post News Service, New York Times, and the Associated Press. The articles are categorized into several topics. Each topic has a topic description and a set of related articles. The topic description includes topic ID, title and narrative. The following is an example of topic description from DUC-2007

```
<topic>
<num>D0703A</num>
<title> steps toward introduction of the Euro </title>
<narr>
```

```
Describe steps taken and worldwide
reaction prior to the introduction
of the Euro on January 1, 1999.
Include predictions and expectations
reported in the press.
</narr>
</topic>
```

3.3 Document Processing

3.3.1 Sentence Tokenization

The raw data provided by DUC-2006 and DUC-2007 is just a string of characters. Tokenization is a process of splitting a string of characters into lexical elements such as words, punctuation or sentences. The first step of document processing is to split the raw data into a set of sentences. We have used OAK System (Sekine, 2002) for this. The OAK system command for sentence splitting is:

./oak -i TEXT -s TEXT -o SENTENCE -O PLAIN -r source-document -w destination-document

3.3.2 Word Stemming

Stemming is the process for reducing inflected (or sometimes derived) words to their stem or root form. So, a stem of a word is the part left after the affixes have been taken off. Affixes can take different forms and are letters that are used to modify the base form of a word. Examples of these are the suffixes; “e”, “ly”, “er”, “ess”, “ed”, and “ing”. A stemming algorithm (for example, Porter stemmer (Porter, 1997)) reduces the words "fishing", "fished", "fish", and "fisher" to the root word, "fish". We used the OAK system for stemming. The OAK system command for word stemmming is:

./oak -i SENTENCE -o POSTAG -O STEM -r source-document -w destination-document

An example of word stemming is:

Input Passage

Angelina Jolie lives on the edge. Jolie, 25, delights in making waves. She is the woman who literally jumped in a swimming pool in her ballgown after winning a Golden Globe Award for Gia (1998).

Stemmed Passage

Angelina Jolie life on the edge. Jolie, 25, delight in make wave. She be the woman who literally jump in a swimming pool in her ballgown after win a Golden Globe Award for Gia (1998).

3.3.3 Part of Speech Tagging

In English, the words that function similarly with respect to what can occur nearby (syntactic distributional properties) or with respect to the affixes they take (morphological properties) are grouped into classes, which is known as Part of Speech (POS). Part-of-speech tagging is the process of assigning a part-of-speech or other syntactic class marker to each word in a corpus. The significance of parts-of-speech for language processing is the large amount of information they give about a word and its neighbors. The number of tagsets for parts-of-speech can vary. For example the number of tagsets for Penn Treebank is 45 (Marcus et al., 1994), for Brown corpus is 87 (Greenbaum et al., 1979) (Francis and Kucera, 1997) and for C7 it is 146 (Garside et al., 1997). We used the Penn Treebank POS tag set which is the most popular one and used to train the OAK system. For example:

Input Passage

Angelina Jolie lives on the edge. Jolie, 25, delights in making waves. She is the woman who literally jumped in a swimming pool in her ballgown after winning a Golden Globe Award for Gia (1998).

Stemmed POS tagged Passage

Angelina/NNP Jolie/NNP life/NN on/IN the/DT edge/NN. Jolie/NNP, 25/CD, delight/NN in/IN make/VB wave/NN. She/PRP be/VB the/DT woman/NN who/WP literally/RB jump/VBP in/IN a/DT swimming/NN pool/NN in/NN her/PRP ballgown/NN after/IN win/NN a/DT Golden/NNP Globe/NNP Award/NNP for/IN / Gia/NNP / (/-LRB- 1998/CD)/-RRB- ./.

Most of the current systems available for part-of-speech tagging has fairly high accuracy (almost 95%). Most tagging algorithms fall into one of two classes: rule-based taggers and probabilistic taggers. Rule-based taggers generally involve a large database of hand-written disambiguation rules. Example of rule-based tagger is EngCG which is based on the Constraint Grammar architecture of (Karlsson et al., 1995). Probabilistic taggers generally learn the probability of a word having a given tag in a given context through supervised machine learning techniques from manually created training corpus. Example of probabilistic tagger is HMM tagger, which is based on Hidden Markov Model.

There is another approach to tag called the transformation based tagger, or the brill tagger (Brill, 1994). The brill tagger shares features of both tagging architectures. Like the rule-based tagger, it is based on rules that determine when an ambiguous word should have a given tag. Like the probabilistic taggers, it has a machine-learning component: the rules are automatically induced from a previously tagged training corpus. For POS tagging, we were using the OAK System which uses a method similar to the Brill tagger, but has 13% fewer errors (Sekine, 2002).

3.3.4 Syntactic Parsing

Syntactic parsing is the process of recognizing a sentence and assigning a syntactic structure to it. Syntactic parsing is used to discover word dependencies. It analyzes a sentence using the grammar rules. We used Charniak parser[1] to get syntactic parse of the passages. Charniak parser is a stochastic syntactic parser which learns the probabilities through supervised machine learning techniques. The probability is the chance that two words are dependent, given certain features like part of speech and distance.

The following is an example of a sentence parsed with the Charniak parser:

```
(S1 (S (NP (NNP Angelina) (NNP Jolie))
(VP (AUX is)
(NP (NP (DT the) (NN woman))
(SBAR (WHNP (WP who))
(S (VP (ADVP (RB literally))
(VBD jumped)
(PP (IN in)
(NP (NP (DT a) (VBG swimming) (NN pool))
(PP (IN in) (NP (PRP$ her) (NN ballgown)))))
(PP (IN after)
(S (VP (VBG winning)
(NP (DT a) (NNP Golden) (NNP Globe) (NN Award))
(PP (IN for)
(NP (NP ( ) (NNP Gia\) ( ) (NNP \) (POS ))
(PRN (-LRB- -LRB-) (NP (CD 1998)) (-RRB- -RRB-)))))
)))))))
```

[1]Available at ftp://ftp.cs.brown.edu/pub/nlparser/.

```
(. .)))
```

Pasca and Harabagiu (2001) demonstrated that with a syntactic form one can see which words depend on other words. There should be a similarity between the words that are dependent in the question and the dependency between words of the passage containing the answer. The importance of syntactic feature in question answering was described by (Zhang and Lee, 2003a), (Moschitti et al., 2007) and (Moschitti and Basili, 2006).

3.3.5 Semantic Parsing

Shallow semantic representations, bearing more compact information, can elude the sparseness of deep structural approaches and the weakness of bag of words models (Moschitti et al., 2007). Initiatives such as PropBank (PB) (Kingsbury and Palmer, 2002) made it possible to design accurate automatic Semantic Role Labeling (SRL) systems (Hacioglu et al., 2004). So, attempting an application of SRL to automatic annotation seems natural, as similarity of an abstract sentence with a document sentence relies on a deep understanding of the semantics of both. For example, let us consider the PB annotation:

```
[ARG0 all] [TARGET use]
[ARG1 the french franc]
[ARG2 as their currency]
```

Such annotation can be used to design a shallow semantic representation that can be matched against other semantically similar sentences, e.g.

```
[ARG0 the Vatican] [TARGET uses]
[ARG1 the Italian lira]
[ARG2 as their currency]
```

To experiment with semantic structures, we parsed the corresponding sentences semantically using a Semantic Role Labeling (SRL) system like ASSERT[2]. ASSERT is an automatic statistical semantic role tagger that can annotate naturally occurring text with semantic arguments. When presented with a sentence, it performs a full syntactic analysis of the sentence, automatically identifies all the verb predicates in that sentence, extracts features for all constituents in the parse tree relative to the predicate, and identifies and tags the constituents with the appropriate semantic arguments.

3.3.6 Topic Signature

Topic signatures that can play a central role in automated text summarization and information retrieval are typically used to identify the presence of a complex concept–a concept that consists of several related components in fixed relationships (Lin and Hovy, 2000). Inspired by the idea presented in (Lin and Hovy, 2000), for each topic present in the data set, we calculate its topic signature defined as below:

$$\begin{aligned} TS &= \{topic, \, signature\} \\ &= \{topic, \langle (t_1, \, w_1), \cdots, (t_n, \, w_n) \rangle \} \end{aligned} \tag{3.1}$$

where $topic$ is the target concept and *signature* is a vector of related terms. Each t_i is a term highly correlated to the $topic$ with association weight, w_i. We use the following log-likelihood ratio to calculate the weights associated with each term (i.e. word) of a sentence:

[2]Available at http://cemantix.org/assert.

$$w_i = log \frac{occurrences\ of\ t_i\ in\ the\ topic}{occurrences\ of\ t_i\ in\ all\ topics} \tag{3.2}$$

To calculate the topic signature weight for each sentence, we sum up the weights of the words in that sentence and then, normalized the weights. Thus, a sentence gets a high score if it has a set of terms that are highly correlated with a target concept (topic).

3.3.7 Rhetorical Structure Theory (RST)

A well-written text is often supported by a hierarchically structured set of coherence relations that reflect the author's intent (Mann and Thompson, 1988). Discourse parsing focuses on a higher-level view of text (called *rhetorical structure*), allowing some flexibility in the choice of formal representation (Duverle and Prendinger, 2009). Rhetorical Structure Theory (Mann and Thompson, 1988) provides a framework to analyze text coherence by defining a set of structural relations to composing units ("spans") of text. The most frequent structural pattern in RST is that two spans of text are related such that one of them has a specific role relative to the other. A paradigm case is a claim followed by evidence for the claim. RST posits an "Evidence" relation between the two spans that is represented by calling the claim span a *nucleus* and the evidence span a *satellite*[3]. In this thesis, we parse each document sentence within the framework of Rhetorical Structure Theory (RST) using a Support Vector Machine (SVM)-based discourse parser described in (Duverle and Prendinger, 2009) that was shown 5% to 12% more accurate than the current state-of-the-art parsers. We observe that in a relation the nucleus often contains the main information while the satellite provides some additional information. Therefore, we assign a weight to

[3]http://www.sfu.ca/rst/01intro/intro.html.

each sentence that is a nucleus of a relation and normalize the weights at the end.

Chapter 4

Evaluation Techniques

4.1 Introduction

In any speech and language processing area a systematic and standard evaluation is important to assess the quality of the system and to compare the performance against other systems. Many NLP tasks, such as parsing, named entity recognition, chunking and semantic role labeling etc., can be automatically evaluated using the standard precision and recall measures. However, the evaluation of a summary is a very difficult task as there is no unique gold standard. For the same source documents there can be multiple summaries and it is always difficult to identify which qualities make a summary a good one since this fact largely depends on the evaluator. Still there are a wide variety of evaluation metrics for summarization, metrics requiring human annotation as well a completely automatic metrics. In this chapter, we discussed the widely available summary evaluation techniques.

Methods for evaluating text summarization can be broadly classified into two categories, *extrinsic* evaluation or task-based evaluation and *intrinsic* evaluation or task-independent evaluation (Sparck-Jones and Galliers, 1996).

4.2 Extrinsic (task-based)

Ideally, summarization results need to be assessed in a task-based setting, determining their usefulness as part of an information browsing and access interface (Mani et al., 2002) (Mckeown et al., 2005) (Koumpis and Renals, 2005). An extrinsic evaluation tests the summarization based on how it affects the completion of some other tasks. It tests the impact of summarization on tasks like relevance assessment, reading comprehension, finding documents from a large collection, routing documents, producing an effective report or presentation using a summary etc. It is also possible to judge the impact of a summarizer on the system in which it is embedded, for example, in a question answering system. The amount of work required to post-edit a summary output to make it more readable can be thought of as another measure to evaluate it. But such extrinsic evaluations are time consuming, expensive and require a considerable amount of careful planning. They are thus not very suitable for system comparisons and evaluation during development (Nenkova, 2006).

4.3 Intrinsic (task-independent)

In intrinsic evaluation, summarization quality is assessed based on the analyses of the summaries directly. This type of evaluations mainly measure the coherence and informativeness of summaries. Intrinsic evaluations are normally employed in such cases, either by soliciting human judgments on the goodness and utility of a given summary, or by a comparison of the summary with a human-authored gold-standard (Nenkova, 2006). This type of evaluation might involve user judgment of fluency of the summary. Measures of fluency can address language complexity, redundancy, coherence, preservation of different structured environments such as lists or tables, grammatical features, etc.

4.3.1 Manual Evaluation

In early 1960s, the evaluation of summaries was mainly done by humans (Edmundson, 1969). In this type of evaluation, human judges rank the summaries mainly based on the coherence and informativeness of the summary.

DUC Manual Evaluation

The Document Understanding Conference (DUC)[1] has been carrying out large-scale evaluations of summarization systems on a common dataset since 2001. In DUC-2007, National Institute of Standards and Technology (NIST) manually evaluated the linguistic features of each submitted summary using a set of quality questions[2]. These linguistic quality questions are targeted to assess how readable and fluent the summaries are, and they measure qualities of the summary that DO NOT involve comparison with a model summary or DUC topic. These questions require a certain readability property to be assessed on a five-point scale from "1" to "5", where "5" indicates that the summary is good with the respect to the quality under question, "1" indicates that the summary is bad with respect to the quality stated in the question, and "2" to "4" show the gradation in between. The quality of the summary is assessed only with respect to the property that is described in the specific category. The information content and responsiveness of the summary are measured separately in the "responsiveness" part of the evaluation.

Grammaticality The summary should have no datelines, system-internal formatting, capitalization errors or obviously ungrammatical sentences (e.g., fragments, missing components) that make the text difficult to read.

Non-redundancy There should be no unnecessary repetition in the summary. Unnecessary repetition might take the form of whole sentences that are repeated, or repeated facts, or the repeated use of a noun or noun phrase (e.g., "Morris Dees") when a pronoun ("he") would suffice.

[1]http://duc.nist.gov.
[2]http://www.nlpir.nist.gov/projects/duc/duc2007/quality-questions.txt.

Referential clarity It should be easy to identify who or what the pronouns and noun phrases in the summary are referring to. If a person or other entity is mentioned, it should be clear what their role in the story is. So, a reference would be unclear if an entity is referenced but its identity or relation to the story remains unclear.

Focus The summary should have a focus. Sentences should only contain information that is related to the rest of the summary.

Structure and Coherence The summary should be well-structured and well-organized. The summary should not just be a heap of related information, but should build from sentence to sentence to a coherent body of information about a topic.

Responsiveness This is measured primarily in terms of the amount of information present in the summary that actually helps to satisfy the information need expressed in the topic statement. The linguistic quality of the summary might play only an indirect role in this judgment, insofar as poor linguistic quality interferes with the expression of information and reduces the amount of information that is conveyed.

Pyramid Evaluation

The pyramid method (Nenkova and Passonneau, 2004) is another manual evaluation technique for summarization evaluation which was concerned with analysis of a key problem in summarization, variation in human summaries. The pyramid method addresses the problem by using multiple human summaries to create a gold-standard and by exploiting the frequency of information in the human summaries in order to assign importance to different facts. This method involves semantic matching of content units to which differential weights are assigned based on their frequencies in a corpus of summaries. This way it is possible to assign more stable, more informative scores, and hence to a meaningful content evaluation.

The Pyramid method tries to reduce the dependency of the evaluation results on the model used for evaluation. It performs semantic analysis of the model summaries and the target summary (a peer). For each topic, a weighted inventory of Summary Content Units (SCUs) is created. Each summary content unit (SCU) represents the same information meaning, even when expressed using different wording in different summaries. An SCU is similar to a collection of paraphrases in that it groups together words and phrases from distinct summaries into a single set, based on shared content. Each SCU is assigned a weight equal to the number of human summarizers who expressed the SCU in their summaries. The distribution of SCU weights is Zipffian, with few SCUs being included by many summarizers and a heavy tail of low-weight SCUs. SCU analysis shows that summaries that differ in content can be equally good and assign a score that is stable with respect of the models when 4 or 5 human summaries are used. The actual pyramid score is equal to the ratio between the weight of content expressed in a summary and the wight of an ideally informative summary with the same number of SCUs (Nenkova, 2006).

The drawback of this approach is that users have to analyze different summaries to manually identify the SCUs which can be tedious. We have used a special annotation tool DUCView[3] to facilitate the process. In this process the annotator must assign a label to the SCU that expresses the shared content. The label is a concise English sentence that states what the annotator views as the meaning of the content unit. Coincidentally, the SCU will have a weight corresponding to the number of model summaries that expresses the designated content. The SCU weight is automatically computed, based the number of summaries that contribute to it, so the annotator is not responsible for assigning weights.

[3]http://www1.cs.columbia.edu/ becky/DUC2006/2006-pyramid-guidelines.html.

4.3.2 Automatic Evaluation

The comparison between different summarization systems is best carried out by humans. But including human judgment for summary evaluation makes the process time consuming and costly. Especially during system development, as the evaluation has to be performed frequently, it is impractical to elicit human judgments for evaluation. Due to this, researchers seek methods for evaluating system output automatically.

The main problem for automatic evaluation methods is the unavailability of *gold standard* for a direct comparison with the system generated summary. Research as early as (Resnick et al.) reported that extracts selected by six different human judges for 10 articles from Scientific American had only 8% overlap on average. The same summary can obtain a recall score that is between 25% and 50% different depending on which of two available human extracts are used for evaluation (Drummey et al., 2000). It is thus unclear how to define a gold-standard.

Precision and recall In the case of extractive summarization, where the output summary consists entirely of material copied from the input, precision and recall can be used as evaluation measure. Recall is the fraction of sentences chosen by the human summarizer that were also correctly identified by the system

$$Recall = \frac{number\ of\ sentences\ chosen\ by\ both\ human\ and\ system}{number\ of\ sentences\ chosen\ by\ human|} \tag{4.1}$$

and precision is the fraction of system sentences that were correct

$$Precision = \frac{number\ of\ sentences\ chosen\ by\ both\ human\ and\ system}{number\ of\ sentences\ chosen\ by\ system|} \tag{4.2}$$

ROUGE In DUC-2007, each topic and its document cluster were given to 4 different NIST assessors, including the developer of the topic. The assessor created a 250-word summary of the document cluster that satisfies the information need expressed in the topic statement. These multiple "reference summaries" were used in the evaluation of our summary content. We considered the widely used evaluation measures Precision (P), Recall (R) and F-measure for our evaluation task.

We evaluate our system generated summaries using the automatic evaluation toolkit ROUGE (Lin, 2004) which has been widely adopted by DUC. We discussed the ROUGE similarity measures in section 3.1.4. We report the two widely adopted ROUGE metrics in the results: ROUGE-2 (bigram) and ROUGE-SU (skip bigram) because these have never been shown not to correlate with the human judgment.

We show 95% confidence interval of the evaluation metric ROUGE-2 for all systems to report significance for doing meaningful comparison. ROUGE uses a randomized method named bootstrap re-sampling to compute the confidence intervals. Bootstrap re-sampling has a long tradition in the field of statistics (Efron and Tibshirani, 1994). The assumption here is that, estimating the confidence interval from a large number of test sets with n test samples drawn from a set of n test samples with replacement is as good as estimating the confidence interval for the test sets of size n from a large number of test sets with n test samples drawn from an infinite set of test samples. The benefit of this assumption is that we only need to consider n samples. We use 1000 sampling points in the bootstrap re-sampling.

4.4 Our Approach

We have used both manual and automatic methods to evaluate our system generated summaries. For manual evaluation, we have used DUC manual evaluation techniques. For automatic evaluation, we have used ROUGE.

Chapter 5

Implementation

5.1 Introduction

The complex question answering problem is a general one. One instance of it was the problem defined in the DUC-2007 main task. In this thesis, we focus on a query-based extractive approach of summarization where a subset of the sentences from the original documents are chosen. Complex questions often seek multiple different types of information simultaneously and do not presuppose that one single answer can meet all of its information needs. For example, the wider focus of the complex questions like: "How is Haiti affected by the earthquake?", suggests that the submitter may not have a single or well-defined information need and therefore may be amenable to receiving additional supporting information that is relevant to some (as yet) undefined informational goal. Multi-document summarization is an intelligent way to handle this type of query.

We have formulated complex question answering problem using different Machine learning approaches. We have also experimented the impact of syntactic and semantic information on complex question answering. In section 5.3 we discuss how our task can be formulated using reinforcement learning technique. In section 5.4 we presented a graph based random walk model for guided summarization task. Section 5.5 details Manifold ranking model for topic-focused multi-document summarization and the impact of syntactic and semantic information on it. In the last section we discuss a complex question answering system where first we decompose the complex question into factoid questions and then used a simple question answering system to generate summary.

5.2 Task Description

The DUC conference series is run by the National Institute of Standards and Technology (NIST) to further progress in summarization and enable researchers to participate in large-scale experiments. This experiment deals with the topic-focused (i.e. query-based) multi-document summarization task as defined in the Document Understanding Conference, DUC-2007. The task is defined as follows:

> *Given a complex question (topic description) and a collection of relevant documents, the task is to synthesize a fluent, well-organized 250-word summary of the documents that answers the question(s) in the topic*

For example, given the topic description (from DUC-2007):

```
<topic>
<num>D0703A</num>
<title> steps toward introduction of the Euro </title>
<narr>
        Describe steps taken and worldwide reaction prior to
        the introduction of the Euro on January 1, 1999. Include
        predictions and expectations reported in the press.
</narr>
</topic>
```

and a collection of relevant documents, the task of the summarizer is to build a summary that answers the question(s) in the topic description. we consider this task to generate topic-oriented 250-word extract summaries for all the topics of DUC-2007 (See sample summary in Appendix).

5.3 Reinforcement Learning

5.3.1 Introduction

Judging the importance of a sentence is the most essential aspect of extractive summary generation. Given a collection of document sentences and their abstract summaries (created by human), we can think of a learner that tries to find the most important sentences that can be extracted as system generated automatic summaries. The importance of a sentence can be verified by measuring its similarity with the abstract summary sentences using a reward function. The more similar a sentence is the better reward it receives. This is reinforcement learning where the task is to learn what to do — how to map situations to actions — so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them (Sutton and Barto, 1998).

Search engines are proved to be adequate as a tool for finding documents on the web. Although there is no limitation in the expressiveness of the user in terms of query formulation, certain limitation exists in what the search engine does with the query. Moreover, search engines provide no way of measuring whether a user is satisfied with the answer or not, and hence, it cannot improve its policy dynamically in real time. This becomes a main motivation of applying the reinforcement approach to our domain. We can treat complex question answering as an interactive problem since we consider that, if real-time user feedback can be provided in terms of reward, the answering systems might evolve substantially by improving automatically as time passes.

Supervised learning techniques are not adequate for learning from interaction. Moreover, it requires a huge amount of human-annotated training data. In interactive problems, it is often impractical to obtain training examples of desired behavior that are both correct and representative of all the situations in which the agent has to act. So, the strategy here is to use a reinforcement approach that can sense the state of the environment to some extent and is able to take actions that affect the state. We assume that a little amount of supervision is provided in the form of a reward function that defines the quality of executed actions. During training, the learner repeatedly constructs action sequences for a set of given documents, executes those actions, and observes the resulting reward. The learner's goal is to estimate a policy that maximizes future expected reward (Branavan et al., 2009).

In this section, we present a reinforcement learning framework for answering complex questions. We simplify our formulation by assuming no real time user interaction. We treat the fact that the human generated abstract summaries are the gold-standards and users (if they were involved) are satisfied with these summaries. Thus, our approach tries to produce automatic summaries that are as close as the abstract summaries. Then, the correspondence between these two types of summaries is learned and the final weights are used to output machine generated summaries from the unseen data.

We formulate the topic-focused multi-document summarization task as a sequential decision problem and used both Markov Decision Process models (MDPs) and Partially Observable MDP models (POMDPs) to solve it. The specification of a sequential decision problem for a fully observable environment with a Markovian transition model is called a Markov Decision Process (MDP) (Russel and Norvig, 2003). With this assumption, the agent always knows which state it is in. MDPs are proved to be useful in a variety of sequential decision problems (Puterman, 1994). The Partially Observable MDP model (POMDP) generalizes the MDP model to allow for even more forms of uncertainty to be accounted for in the process. So, an increasing number of researchers in many areas are becoming interested in the application of POMDPs to model different problems that involve uncertainty (Cassandra, 1998). We explain how Partially Observable Markov Decision Processes (POMDPs) can be used for modeling the inherent uncertainty in the multi-document summarization task.

5.3.2 MDP Model

Problem Formulation

Almost all reinforcement learning algorithms are based on estimating value functions — functions of states (or of state-action pairs) that estimate how good it is for the agent to be in a given state (Sutton and Barto, 1998). We formulate the complex question answering problem by estimating an action-value function. We define the value of taking action a in state s under a policy π , denoted $Q^{\pi}(s,a)$, as the expected return starting from s, taking the action a, and thereafter following policy π:

$$\begin{aligned} Q^{\pi}(s,\, a) &= E_{\pi}\{R_t | s_t = s,\, a_t = a\} \\ &= E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma_k r_{t+k+1} | s_t = s,\, a_t = a\right\} \end{aligned} \tag{5.1}$$

Here, E_{π} denotes the expected value given that the agent follows policy π and and t is any time step. We call Q^{π} the action-value function for policy π. γ stands for the discount factor that determines the importance of future rewards. We try to find out the optimal policy through policy iteration. Once we get the optimal policy (π^*) the agent chooses the actions using the Maximum Expected Utility Principle (Russel and Norvig, 2003).

Environment, State & Actions For complex question answering problem we are given a complex question q and a collection of documents $D = \{d_1,\, d_2,\, d_3,\, \ldots,\, d_n\}$, we have to find out an answer (extract summary). The state is defined by the current status of the answer space. Initially, there is no sentence in the answer pool. So, the initial state s_0 is empty. In each iteration, we add a sentence from the document to the answer pool that in turn changes the state. In each state we have a set of actions. Actions are defined by selecting a sentence from the remaining document sentences that are not included so far in the extract summary.

Reward Function During training for each complex question, we had abstract summaries generated by human as answers. After taking each action a, we computed the immediate reward, r using the following function:

$$r = relevance(a) \; - \; 0.5 * redundancy(a) \tag{5.2}$$

Here, $relevance(a)$ is the textual similarity measure between the selected sentence and the abstract summaries. $redundancy(a)$ is the similarity measure between the selected sentence and the current state (set of sentences already chosen). By including redundancy in the immediate reward calculation we discourage redundancy in the final extract summary. We measure the textual similarity using ROUGE (Recall-Oriented Understudy for Gisting Evaluation) 3.1.4. The ROUGE measures considered are: ROUGE-N (N = 1, 2, 3, 4), ROUGE-L, ROUGE-W and ROUGE-S.

Function Approximation In many tasks such as the one to which we apply reinforcement learning, most states encountered will never have been experienced exactly before. This will almost always be the case when the state or action spaces include continuous variables or complex sensations. As in our case the number of states and actions are infinite, the approximate action-value function is represented as a parameterized functional form with parameter vector, $\vec{\theta}_t$. Our approximate action-value function is a linear function of the parameter vector, $\vec{\theta}_t$. Corresponding to every state-action pair $(s,\, a)$, there is a column vector of features, $\vec{\varphi}_s = [\varphi_s(1),\ \varphi_s(2),\ \ldots,\ \varphi_s(n)]^T$ with the same number of components as $\vec{\theta}_t$. The approximate action-value function is given by:

$$Q_t(s,\ a) = \vec{\theta}_t^T \vec{\varphi}_s = \sum_{i=1}^{n} \theta_t(i)\varphi_s(i) \tag{5.3}$$

Markov Decision Process (MDP) Our environment has the Markov property. That is, given the current state and action we can predict the next state and expected next reward. For our problem formulation, given the current state s if we take an action a, the next state will be $s^{'} = s + a$ since our action is to choose a sentence from the document collection and adding it into the extract summary pool. Given any state and action, s and a, the transition model is defined by:

$$\rho^a_{ss^{'}} = P_r\left\{s_{t+1} = s^{'}|s_t = s,\ a_t = a\right\} \tag{5.4}$$

$\rho^a_{ss^{'}}$ will be 1 when $s^{'} = s + a$. For all other states, the transition probability will be 0. Similarly, given any current state and action, s and a, together with any next state, $s^{'}$, the expected value of the next reward is:

$$R^a_{ss^{'}} = E\left\{r_{t+1}|s_t = s,\ a_t = a,\ s_{t+1} = s^{'}\right\} \tag{5.5}$$

We viewed our problem as an infinite horizon sequential decision making problem. For calculating the reward of a state-action pair, we used the discount factor γ. We kept the initial value of γ as 0.1. The value of γ decreases with the increase of iteration counts.

Reinforcement Learning

Our reinforcement learning problem finds the parameter vector $\vec{\theta}$ that maximize $Q(s,a)$ from Equation 5.3. Policy gradient algorithms tend to estimate the parameters θ by performing a stochastic gradient ascent. The gradient is approximated by interacting with the environment, and the resulting reward is used to update the estimate of θ. Policy gradient algorithms optimize a non-convex objective and are only guaranteed to find a local optimum (Branavan et al., 2009). We use a modified linear, gradient-descent version of Watkins' $Q(\lambda)$ algorithm with ε-greedy policy to determine the best possible action i.e. to select the most important sentences. We have used eligibility[1] traces to keep trace of the parameters that should go under learning changes in each cycle of learning. The eligibility traces are updated in two steps. If an exploratory action is taken, they are set to zero for all state-action pairs. Otherwise, the eligibility traces for all state-action pairs are decayed by $\gamma\lambda$. In the second step, the eligibility trace value for the current state-action pair is incremented by 1 while accumulating traces. The original version of the Watkins' $Q(\lambda)$ algorithm uses a linear, gradient-descent function approximation with binary features. However, since we deal with a mixture of real-valued and boolean features, we modified the algorithm to induce a different update for the eligibility traces. In the second step of eligibility trace update, we increment the value by the corresponding feature score. The addition of a random jump step avoids the local maximums in our algorithm. We reduced the step size α of the gradient method by 0.99 as the learning converges towards the goal. Algorithm 1 shows all the steps in detail.

Our formulation with its modified version is unique in how it represents the complex

[1]Eligibility traces are one of the basic mechanisms of reinforcement learning. In mechanistic point of view, an eligibility trace is a temporary record of the occurrence of an event, such as the visiting of a state or the taking of an action. The trace marks the memory parameters associated with the event as eligible for undergoing learning changes. Thus, eligibility traces help bridge the gap between events and training information (Sutton and Barto, 1998).

```
Input: $\alpha$, $\vec{\theta}$, $\vec{e}$, $\lambda$, $\gamma$, $\delta$, $\varphi$, $\varepsilon$, number of sentences/actions available $T$
Output: A vector $\vec{\theta}$ of learned weights
Initialize: $\vec{\theta}$ to $\vec{0}$, $\alpha$ to 0.01, $\gamma$ to 0.1, $\lambda$ to 0.9
$\vec{e} = \vec{0}$
$s, a \leftarrow$ initial state and action of episode
$\varphi \leftarrow$ set of features present in $s, a$
for each $i = 1 \dots T$ do
    if $s$ is not terminal then
        for $i \in \varphi$ do
            $e(i) \leftarrow e(i) + \varphi(i)$
        end
        Take action $a$, observe reward $r$, and next state, $s$
        $\delta \leftarrow r - \sum_i \varphi(i)\theta(i)$
        for $a \in A(s)$ do
            $\varphi \leftarrow$ set of features present in $s, a$
            $Q_a \leftarrow \sum_i \varphi(i)\theta(i)$
        end
        $\delta \leftarrow \delta + \gamma max_a Q_a$
        $\theta \leftarrow \theta + \alpha\delta\vec{e}$
        $\alpha \leftarrow 0.99 * \alpha$
        $probability \leftarrow get\ the\ probability\ of\ choosing\ a\ random\ action$
        if $probability \leq 1 - \varepsilon$ then
            for $a \in A(s)$ do
                $Q_a \leftarrow \sum_i \varphi(i)\theta(i)$
            end
            $a \leftarrow argmax_a Q_a$
            $\vec{e} \leftarrow \gamma\lambda\vec{e}$
        else
            $a \leftarrow$ a random action $\in A(s)$
            $\vec{e} \leftarrow 0$
        end
    end
end
return $\vec{\theta}$
```

Algorithm 1: Modified Watkins' $Q(\lambda)$ algorithm

question answering task in the reinforcement learning framework.

5.3.3 POMDP Model

In contrast to Markov Decision Processes (MDPs) that provide a good statistical framework for allowing forward planning in a fully observable environment, POMDPs provide a mathematical model for the sequential decision-making problems in partially observable environments. The key advantage of the POMDP formalism is that it provides a complete and principled framework for modeling the inherent uncertainty of the problem under consideration (Young, 2006).

Formal Definition

A POMDP is a tuple $\left\langle S, A, T\left(s, a, s'\right), O(s, o), R(s)\right\rangle$, where:

- S is the *state space* defined as a set of mutually exclusive states.
- A is the *action space* i.e. a set of possible actions that an agent can perform in one state.
- $T\left(s, a, s'\right)$ is the *transition model* that denotes the probability of reaching state s' if action a is done in state s.
- $O(s, o)$ is the *observation model* that specifies the probability of perceiving the observation o in state s.
- $R(s)$ is the *reward function* that encodes the utility for the agent to perform the action a while in state s.

Belief State The key idea of POMDP is the assumption that the state of the world is not directly accessible and can only be inferred via observation. Such uncertainty is expressed in the belief state b, which is a probability distribution over all possible states. We denote $b(s)$ for the probability assigned to the actual state s by belief state b. We can calculate a current belief state as the conditional probability distribution over the actual states given the sequence of observations and actions so far. In the current belief state $b(s)$, if action a is performed and observation o is perceived, the new belief state will be given by:

$$b^{'}(s^{'}) = \alpha O(s^{'}, o) \sum_{s} T(s, a, s^{'}) b(s), \tag{5.6}$$

where α is a normalizing constant that makes the belief state sum to 1.

Policies The solution to the POMDP problem must specify what the agent should do for any state that it might reach. A solution of this kind is termed as policy, denoted by π (Russel and Norvig, 2003). The fundamental concept of POMDP is that the optimal action depends only on the agent's current belief state since it has limited access to the actual current state. Therefore, the optimal policy (that yields the highest expected utility), denoted by $\pi^{*}(b)$ is a mapping from belief states to actions. We define the value of taking action a in belief state b under a policy π , denoted $Q^{\pi}(b, a)$, as the expected return starting from b, taking the action a, and thereafter following policy π:

$$Q^{\pi}(b, a) = E_{\pi}\{R_t | b_t = b, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma_k r_{t+k+1} | b_t = b, a_t = a\right\}.$$

Here, E_{π} denotes the expected value given that the agent follows policy π and t is any time step. We call Q^{π} the action-value function for policy π. γ stands for the discount factor that determines the importance of future rewards. We try to find out the optimal policy through policy iteration. Once we get the optimal policy (π^{*}) the agent chooses the actions using the Maximum Expected Utility Principle.

POMDP-based Summarization

For the topic-focused multi-document summarization task, we are given a query (i.e. topic), q and a collection of related documents $D = \{d_1, d_2, d_3, \ldots, d_n\}$, we have to find out an extract summary. We define the state space as a summary state and a non-summary state. Once we get the highest probability of the belief that we reached the summary state, a reward 1 is assigned, otherwise 0 is returned. When we select a sentence as important, we calculate its observation probability by measuring its relatedness with the given abstract summaries. We measure this similarity using ROUGE 3.1.4. The ROUGE measures considered are: ROUGE-N (N = 1, 2, 3, 4), ROUGE-L, ROUGE-W and ROUGE-S. The action space is defined as selecting a sentence from the remaining document sentences that are not included so far in the extract summary.

Solving POMDP

In our formulation, the number of actual states is two (summary state, non-summary state). Since the environment is partially accessible, we modeled the problem as a POMDP by introducing an infinite number of belief states that are observable to the agent. To solve this problem, we consider a functional approximation approach to the problem. We represent the approximated action-value function as a parameterized functional form with parameter vector, $\vec{\theta}_t$. Our approximate action-value function is a linear function of the parameter vector, $\vec{\theta}_t$. Corresponding to every belief state–action pair (b, a) , there is a column vector of features, $\vec{\varphi}_b = (\varphi_b(1), \varphi_b(2), \ldots, \varphi_b(n))^T$ with the same number of components as $\vec{\theta}_t$. The approximate action-value function is given by:

$$Q_t(b,\, a) = \vec{\theta}_t^T \vec{\varphi}_b = \sum_{i=1}^{n} \theta_t(i)\varphi_b(i). \tag{5.7}$$

We use a policy gradient algorithm to solve our POMDP problem. Policy gradient algorithms tend to estimate the parameters θ by performing a stochastic gradient ascent. The gradient is approximated by interacting with the environment, and the resulting reward is used to update the estimate of θ. Policy gradient algorithms optimize a non-convex objective and are only guaranteed to find a local optimum (Branavan et al., 2009). We use a modified linear, gradient-descent version of Watkins' $Q(\lambda)$ algorithm with ε-greedy policy to determine the best possible action i.e. to select the most important sentences. As long as the initial policy selects greedy actions, the algorithm keeps learning the action-value function for the greedy policy. But when an exploratory action is selected by the behavior policy, the eligibility traces[2] for all state-action pairs are set to zero. The eligibility traces are updated in two steps. If an exploratory action is taken, they are set to zero for all state-action pairs. Otherwise, the eligibility traces for all state-action pairs are decayed by $\gamma\lambda$[3]. In the second step, the eligibility trace value for the current state-action pair is incremented by 1 while accumulating traces. We modified the algorithm to induce a different update for the eligibility traces. In the second step of eligibility trace update, we increment the value by the corresponding feature score (Features are discussed in the next section). The addition of a random jump step avoids the local maximums in our algorithm. We reduced the step size, α by 0.99 as learning converges toward the goal.

[2]Eligibility traces are one of the basic mechanisms of reinforcement learning. In mechanistic point of view, an eligibility trace is a temporary record of the occurrence of an event, such as the visiting of a state or the taking of an action. The trace marks the memory parameters associated with the event as eligible for undergoing learning changes. Thus, eligibility traces help bridge the gap between events and training information. Sutton and Barto (1998)

[3]For calculating the reward of a state-action pair, we used the discount factor γ. We use the ε-greedy policy (meaning that most of the time this policy chooses an action that has maximal estimated action value, but with probability ε they instead select an action at random) to balance between exploration and exploitation during the training phase. We set $\varepsilon = 0.1$. So, our algorithm chooses an action with the best action-value 90% times and for 10% time it chooses an action randomly. We kept the initial value of γ as 0.1. The value of γ decreases with the increase of iteration counts.

5.3.4 Feature Space

We represent each sentence of a document as a vector of feature-values (φ in Algorithm 1). We divide the features into two major categories: static and dynamic. Static features include two types of features, where one declares the importance of a sentence in a document and the other measures the similarity between each sentence and the user query (Chali et al., 2009, Edmundson, 1969, Sekine and Nobata, 2001). We use the following eighteen features as static features. We use one dynamic feature that measures the similarity of already selected candidate with each remaining sentences. The dynamic feature is used to ensure that there is no redundant information present in the final extract summary.

Static Features: Importance

Position of Sentences We give the score 1 to those sentences found within the first and the last 3 sentences of a document and assign score 0 to the rest, as the early and late sentences are considered important intuitively (Hasan, 2009).

Length of Sentences If a sentence is longer, we can heuristically claim that it has a better chance of inclusion in the summary. We give the score 1 to a longer sentence and assign the score 0 otherwise. In this research, we considered a sentence as long if it has more than 11 words.

Title Match If we find a match such as exact word overlap, synonym overlap or hyponym overlap between the title and a sentence, we give it the score 1, otherwise 0.

Named Entity The score 1 is given to a sentence, which contains a certain Named Entity class among: PERSON, LOCATION, ORGANIZATION, GPE (Geo-Political Entity), FACILITY, DATE, MONEY, PERCENT, TIME. We believe that a certain Named Entity increases the importance of a sentence. We use *OAK* System (Sekine, 2002), from New York University for Named Entity recognition.

Cue Word Match The probable relevance of a sentence is affected by the presence of pragmatic words such as “significant”, “impossible”, “in conclusion”, “finally” etc. We use a cue word list of 228 words. We give the score 1 to a sentence having any of the cue words and 0 otherwise.

Static Features: Query-related

n–gram Overlap This is the recall between the query and the candidate sentence where n stands for the length of the n–gram $(n = 1, 2, 3, 4)$.

LCS Given two sequences S_1 *and* S_2, the longest common subsequence (LCS) of S_1 *and* S_2 is a common subsequence with maximum length.

WLCS Weighted Longest Common Subsequence (WLCS) improves the basic LCS method to remember the length of consecutive matches encountered so far (Lin, 2004). We compute the WLCS-based F-measure between a query and a sentence.

Skip-Bigram Skip-bigram measures the overlap of skip-bigrams between a candidate sentence and a query sentence. Skip-bigram counts all in-order matching word pairs while LCS only counts one longest common subsequence.

Exact-word Overlap This is a measure that counts the number of words matching exactly between the candidate sentence and the query sentence.

Synonym Overlap This is the overlap between the list of synonyms of the important words extracted from the candidate sentence and the query related words. We use WordNet (Fellbaum, 1998) database for this purpose.

Hypernym/Hyponym Overlap It is the overlap between the list of hypernyms and hyponyms (up to level 2 in WordNet) of the nouns extracted from the sentence and the query related words.

Gloss Overlap Our systems extract the glosses for the proper nouns from WordNet. Gloss overlap is the overlap between the list of important words that are extracted from the glossary definition of the nouns in the candidate sentence and the query related words.

Syntactic Feature The syntactic similarity between the *query* and the *sentence* is calculated after parsing them into syntactic trees using a parser such as (Charniak, 1999) and finding the similarity between the two trees using the *tree kernel* (Collins and Duffy, 2001).

Basic Element (BE) Overlap We extract BEs for the sentences in the document collection. Then we filter those BEs by checking whether they contain any word which is a *query word* or a *query related word* and get the BE overlap score (Hovy et al., 2005).

Dynamic Feature To lower redundancy in the extract summary, for each sentence that is selected we measure its similarity with the remaining non-selected sentences using ROUGE. We use the Maximal Marginal Relevance (MMR)[4] method (Carbonell and Goldstein, 1998) to balance this feature with query relevance.

[4]A sentence has the high marginal relevance if it is both relevant to the query and contains minimal similarity to previously selected sentences.

5.3.5 Evaluation Framework

Corpus

We use 50 topics of DUC-2006 data to learn the weights respective to each feature and then use these weights to produce extract summaries for the first 25 topics (subset of the given 45 topics) of the DUC-2007 data.

Baseline System

We report the evaluation scores of one baseline system (used in DUC-2007) in each of the tables in order to show the level of improvement our system achieved. The baseline system generates summaries by returning all the leading sentences (up to 250 words) in the $\langle TEXT \rangle$ field of the most recent document(s).

SVM Settings

We compare the performance of our reinforcement learning approach with a SVM-based technique to answer complex questions. A Support Vector based approach requires huge amount of training data during the learning stage. Here, typically, the training data includes a collection of sentences where each sentence is represented as a combination of a feature vector and corresponding class label ($+1$ or -1). We obtain a training data set by automatically annotating (using only ROUGE similarity measures) 50% sentences of each document set as positive and the rest as negative.

During training step, we used the third-order polynomial kernel keeping the value of the trade-off parameter C (equation 2.8) as default. We used the SVM^{light}[5] (Joachims, 1998a) package. We performed the SVM training experiments in the WestGrid[6] for faster computation. We used the *Cortex* cluster which comprises some shared-memory computers for large serial jobs or demanding parallel jobs. Forcing summaries to obey a certain length constraint is a common set-up in summarization as in the multi-document summarization task at DUC-2007, the word limit was 250 words. In SVM systems, we used $g(x)$, the normalized distance from the hyperplane to x to rank the sentences. Then, we chose the top N sentences until the summary length is reached.

5.3.6 Results and Analysis

MDP

Table 5.1 and Table 5.2 show the ROUGE scores of the reinforcement system and the SVM system, respectively. In Table 5.3, we compare the ROUGE-F scores of the baseline system, SVM system and reinforcement system. From here, we find that the reinforcement system improves the ROUGE-2 and ROUGE-SU scores over the baseline system by 32.9% and 21.1%, respectively. On the other hand, the reinforcement system advances the SVM system improving the ROUGE-2 and ROUGE-SU scores by 28.4% and 2.7%, respectively. In table 5.4 and table 5.5, we report the 95% confidence intervals for ROUGE-2 and ROUGE-SU to show significance for meaningful comparison.

Most Effective Features After the training phase, we get the final updated weights corresponding to each feature. A weight value close to zero indicates that the associated feature

[5]http://svmlight.joachims.org/
[6]http://westgrid.ca/

Measures	ROUGE-2	ROUGE-SU
Precision	0.0878	0.1417
Recal	0.0849	0.1319
F-score	0.0863	0.1365

Table 5.1: ROUGE measures for MDP system

Measures	ROUGE-2	ROUGE-SU
Precision	0.0707	0.1477
Recall	0.0641	0.1209
F-score	0.0672	0.1329

Table 5.2: ROUGE measures for SVM system

Systems	ROUGE-2	ROUGE-SU
Baseline	0.0649	0.1127
SVM	0.0672	0.1329
MDP	0.0863	0.1365

Table 5.3: Performance comparison between Baseline, SVM and MDP systems: F-Score

Systems	ROUGE-2
Baseline	0.060870 - 0.068840
SVM	0.057032 - 0.078794
MDP	0.074092 - 0.096803

Table 5.4: 95% confidence intervals for Baseline, SVM and MDP systems: ROUGE-2

Systems	ROUGE-SU
Baseline	0.108470 - 0.116720
SVM	0.121819 - 0.144470
MDP	0.123609 - 0.147870

Table 5.5: 95% confidence intervals for Baseline, SVM and MDP systems: ROUGE-SU

to this weight can be eliminated because it does not contribute any relevant information for action selection. So, from this viewpoint we can infer that — weights reflect the effectiveness of a certain feature. Table 5.6 shows the top ten final feature weights (ranked by higher effectiveness) for this problem domain that we find after the training experiment.

Final Weight	Associated Feature
0.012837	Basic Element Overlap
0.007994	Syntactic Feature
0.007572	Length of Sentences
0.006483	Cue Word Match
0.005235	Named Entity Match
0.002201	2–gram Overlap
0.002182	Title Match
0.001867	Skip–Bigram
0.001354	WLCS
0.001282	1–gram Overlap

Table 5.6: Effective features indicated by MDP system

POMDP

Table 5.7 to Table 5.9 show the ROUGE-1, ROUGE-2, and ROUGE-W scores of the MaxEnt system and the POMDP-based system.

Systems	Recall	Precision	F-score
MaxEnt	0.3633	0.3769	0.3699
POMDP	0.4172	0.3484	0.3796

Table 5.7: ROUGE-1 measures

Table 5.10 reports the 95% confidence intervals of the ROUGE F-measures for the two systems.

Systems	Recall	Precision	F-score
MaxEnt	0.0799	0.0831	0.0814
POMDP	0.0966	0.0805	0.0878

Table 5.8: ROUGE-2 measures

Systems	Recall	Precision	F-score
MaxEnt	0.0876	0.1684	0.1152
POMDP	0.0999	0.1548	0.1214

Table 5.9: ROUGE-W measures

From these tables we find that the POMDP system improves the MaxEnt system by a significant margin in terms of almost all the ROUGE measures proving the effectiveness of our POMDP model. In Table 5.11, our proposed POMDP model is also compared with the NIST baseline system. The NIST baseline is the official baseline system established by NIST in DUC-2007. We also list the average ROUGE scores of all the participating systems for DUC-2007 (i.e. AverageDUC). From these results, we can see that the proposed POMDP model mostly outperforms the NIST baseline system. We also find that our system achieved higher ROUGE scores as comparable to the average scores of all the participating systems of DUC-2007.

Systems	R-1	R-2
MaxEnt	0.3537 - 0.3860	0.0694 - 0.0935
POMDP	0.3663 - 0.3926	0.0769 - 0.0974

Table 5.10: 95% confidence intervals for different systems

Systems	ROUGE-1	ROUGE-2	ROUGE-W
POMDP	0.37964	0.08783	0.12143
Baseline	0.33434	0.06479	0.11360
AverageDUC	0.40059	0.09550	0.13726

Table 5.11: System comparison (F-scores)

5.4 Augmenting TAC Ontologies with Random Walk Models

5.4.1 Introduction

Recently, there has been increased interest in topic-focused multi-document summarization where the task is to produce automatic summaries in response to a given topic or specific information request stated by the user. Given a topic and a set of related newswire articles, the guided summarization task in Text Analysis Conference (TAC 2010) aims to encourage a deeper semantic analysis of the source documents to select important concepts. They put each topic in a predefined category and associate with it a list of aspects that act as a guide for selecting the most relevant sentences into the summaries. We term these "list of aspects" as "TAC ontologies" and use them to build a novel methodology for topic-focused multi-document summarization that operates on a Markov chain tuned to extract the most important sentences by following a random walk paradigm. Our evaluations suggest that augmentation of TAC ontologies with the random walk model can considerably improve the summary quality in comparison with the random walk model alone.

The recent 2005, 2006, and 2007 Document Understanding Conferences (DUC[7]) have modeled real-world complex question answering as a form of multi-document summa-

[7]http://duc.nist.gov/.

rization and hence, tasked their systems requiring participants to provide summaries from multiple documents as answers to complex questions. The 2010 Text Analysis Conference (TAC) presents a new direction in focused summarization research with a novel task termed *guided summarization*[8]. The goal of guided summarization is to encourage a deeper linguistic (semantic) analysis of the source documents instead of relying only on document word frequencies to select important concepts. The guided summarization task is to write a 100-word summary of a set of 10 newswire articles for a given topic, where the topic falls into a predefined category. Participants are given a list of aspects for each category, and a summary must include all aspects found for its category. We call these lists of aspects as "TAC ontology[9]" and use those to focus our search to include the most relevant sentences into the summary. We propose a novel topic-focused multi-document summarization framework that operates on a Markov chain model (Lafferty and Zhai, 2001) and follows a random walk paradigm (inspired from (Harabagiu et al., 2006)) in order to generate possible summary sentences. We build three alternative systems for summary generation that are based on TAC ontology, random walk model, and a combination of both. We run our experiments on the TAC-2010, and DUC-2006 data and based on the evaluation results we argue that augmenting TAC ontologies with a random walk model often outperforms the other two alternatives.

5.4.2 Our Approaches

In this section, we give a detailed description of three approaches that we used for the task of topic-focused multi-document summarization. At the end of each method applied, we get a candidate summary from it. Therefore, three different techniques give us three can-

[8] *http://www.nist.gov/tac/2010/Summarization/Guided-Summ.2010.guidelines.html.*

[9] Since we plan to build a large knowledge base having all the aspects to satisfy the topic-focused information need, for now, we term it as this.

didate summaries for the same given topic (See example summaries in Appendix). Figure 5.1 presents the overall architecture of our systems.

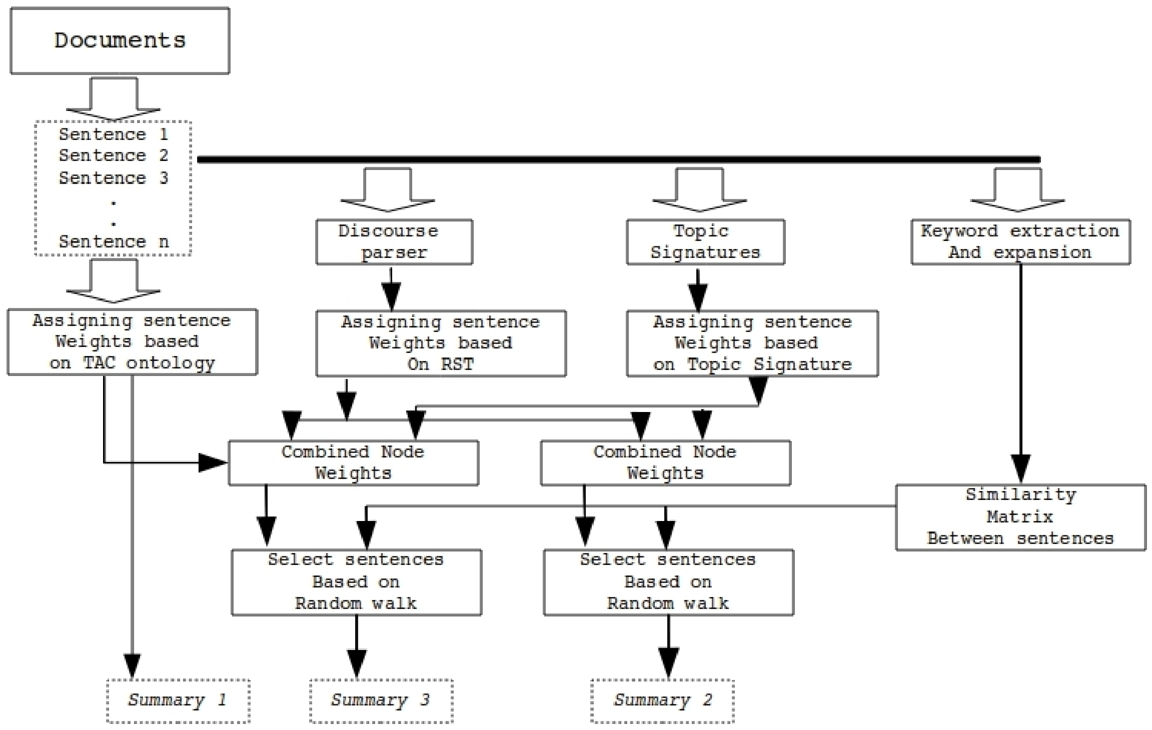
Documents
Sentence 1
Sentence 2
Sentence 3
.
.
Sentence n
Discourse parser
Topic Signatures
Keyword extraction And expansion
Assigning sentence Weights based on TAC ontology
Assigning sentence Weights based On RST
Assigning sentence Weights based on Topic Signature
Combined Node Weights
Combined Node Weights
Similarity Matrix Between sentences
Select sentences Based on Random walk
Select sentences Based on Random walk
Summary 1
Summary 3
Summary 2

Figure 5.1: The overall architecture of our approaches

TAC Ontology-based System

The Guided Summarization task at TAC-2010 aims to encourage summarization systems to make a deeper linguistic (semantic) analysis of the source documents instead of relying only on document word frequencies to select important concepts. The task is to write a 100-word summary of a set of 10 newswire articles for a given topic, where the topic falls into a predefined category. There are five topic categories:

1. Accidents and Natural Disasters

2. Attacks
3. Health and Safety
4. Endangered Resources
5. Investigations and Trials

Participants are given a list of important aspects for each category, and a summary must cover all these aspects (if the information can be found in the documents) including any other information relevant to the topic. The list of aspects that we call "TAC ontology" are shown in Appendix A.

Our first approach simply uses the list of aspects i.e. TAC ontologies in order to find the most relevant sentences from the document collection to create 250-word topic-focused summaries. For each question (i.e. aspect) in a category, we did keyword expansion using WordNet[10]. For example, the word "happen" being a keyword in the question "What happened?" returns the words: *occur, pass, fall out, come about, take place* from WordNet. On the other hand, for each document sentence in the collection we perform Named Entity (NE) tagging using *OAK* system (Sekine, 2002). Named Entities (NE) are defined as terms that refer to a certain entity. For instance, *Canada* refers to a certain *country*, and *$200* refers to a certain quantity of money. OAK system has 150 named entities (such as PERSON, LOCATION, ORGANIZATION, GPE (Geo-Political Entity), FACILITY, DATE, MONEY, PERCENT, TIME etc.) that can be tagged. They are included in a hierarchy. We weigh each sentence based on the presence of one or more Named Entity classes. We rank[11] the document sentences based on the following two criteria:

1. Similarity of each sentence with the expanded question, and

[10]WordNet (http://wordnet.princeton.edu/.) is a widely used semantic lexicon for the English language. It groups English words (i.e. nouns, verbs, adjectives and adverbs) into sets of synonyms called synsets, provides short, general definitions (i.e. gloss definition), and records the various semantic relations between these synonym sets.

[11]To satisfy the length limit of 250-words, we give a score to each sentence.

2. weight assigned to each sentence by the NE tagging procedure[12].

Finally, we select the top-ranked sentences to be included in the candidate summary (Summary 1 in Figure 5.1).

Random Walk Model

For our second approach, we select the most relevant sentences by following a random walk on a graph where each node is a document sentence and the edges represent similarity between sentences. The whole procedure operates on a Markov chain (MC). A Markov chain is a process that consists of a finite number of states and some known probabilities p_{ij}, where p_{ij} is the probability of moving from state j to state i. For each node (i.e. sentence) and each edge in the graph, we calculate *"node weight"* and *"edge weight"*, respectively. Once we find all the node weights and edge weights, we perform a random walk on the graph following a Markov chain model in order to select the most important sentences. The initial sentence is chosen simply based on the node (sentence) weights using the following formula:

$$\text{Initial Sentence} = \arg\max_{i=1}^{N} \left(weight\left(S_i\right)\right), \tag{5.8}$$

where N is the total number of nodes in the graph. After finding the initial best sentence, in each step of the random walk we calculate the probability (transition probability) of choosing the next relevant sentence based on the following equation:

[12]For example, for a question (aspect) like *"when did the accident happened?"*, we search for ⟨Time⟩ tag in the NE tagged sentences and give them higher weights if found.

$$P(S_j|S_i) = \frac{1}{\alpha}\ \arg\max_{j=1}^{Z} \left(weight\left(S_j\right)\ \times\ similarity\left(S_i,\ S_j\right)\right), \tag{5.9}$$

where S_i is the sentence chosen earlier, S_j is the next sentence to be chosen, Z is the set of sentence indexes that does not contain i, the $similarity(S_i,\ S_j)$ function returns a similarity score between the already selected sentence and a new sentence under consideration, and α is the normalization factor that is determined as follows:

$$\alpha = \sum_{j=1}^{Z} \left(weight\left(S_j\right)\ \times\ similarity\left(S_i,\ S_j\right)\right). \tag{5.10}$$

Node Weight We associate each node (sentence) in the graph a weight that indicates the importance of the node with respect to the document collection. Node weights are calculated based on a Topic Signature (TS) model proposed in (Lin and Hovy, 2000) and Rhetorical Structure Theory (RST) (Mann and Thompson, 1988). We combined the weights of TS and RST, and normalized it to get the final weights of the sentences/nodes.

Topic Signature Topic signatures that can play a central role in automated text summarization and information retrieval are typically used to identify the presence of a complex concept–a concept that consists of several related components in fixed relationships (Lin and Hovy, 2000). Inspired by the idea presented in (Lin and Hovy, 2000), for each topic present in the data set, we calculate its topic signature defined as below:

$$\begin{aligned} TS &= \{topic,\ signature\} \\ &= \{topic,\ \langle(t_1,\ w_1),\ \cdots,\ (t_n,\ w_n)\rangle\}, \end{aligned} \tag{5.11}$$

where $topic$ is the target concept and signature is a vector of related terms. Each t_i is a term highly correlated to the $topic$ with association weight, w_i. We use the following log-likelihood ratio to calculate the weights associated with each term (i.e. word) of a sentence:

$$w_i = log \frac{occurrences\ of\ t_i\ in\ topic\ j\ sentences}{occurrences\ of\ t_i\ in\ all\ topics'\ sentences}. \tag{5.12}$$

To calculate the topic signature weight for each sentence, we sum up the weights of the words in that sentence and then, normalized the weights. Thus, a sentence gets a high score if it has a set of terms that are highly correlated with a target concept (topic).

Rhetorical Structure Theory (RST) A well-written text is often supported by a hierarchically structured set of coherence relations that reflect the authors intent (Mann and Thompson, 1988). Discourse parsing focuses on a higher-level view of text (called *rhetorical structure*), allowing some flexibility in the choice of formal representation (Duverle and Prendinger, 2009). Rhetorical Structure Theory (Mann and Thompson, 1988) provides a framework to analyze text coherence by defining a set of structural relations to composing units (“spans”) of text. The most frequent structural pattern in RST is that two spans of text are related such that one of them has a specific role relative to the other. A paradigm case is a claim followed by evidence for the claim. RST posits an “Evidence” relation between the two spans that is represented by calling the claim span a *nucleus* and the evidence span a *satellite*[13]. we parse each document sentence within the framework of Rhetorical Structure Theory (RST) using a Support Vector Machine (SVM)-based discourse parser described in (Duverle and Prendinger, 2009) that was shown 5% to 12% more accurate than current state-of-the-art parsers. We observe that in a relation the nucleus often contains the main

[13]http://www.sfu.ca/rst/01intro/intro.html

information while the satellite provides some additional information. Therefore, we assign a weight to each sentence that is a nucleus of a relation and normalize the weights at the end.

Edge Weight Edge weight is determined by measuring similarity between the sentences. We use the *OAK* system (Sekine, 2002) to get the stemmed word of a sentence. Then, we remove the stopwords from the sentence using a stopword list. We expand the remaining keywords of the sentence using WordNet. Finally, we find out the similar words between each pair of sentences that denotes the edge weight between the two sentences. We build a similarity matrix by populating into it the edge weights between sentences.

Augmenting TAC ontology with Random Walk Model

The third approach we follow to generate a candidate summary is the augmentation of TAC ontologies into the random walk framework. Motivated by (Harabagiu et al., 2006), where they described how a random walk could be used to populate a network with potential decompositions of a complex question, we propose to use the list of aspects (given in TAC-2010) in the random walk model as a guided way to provide a better coverage to satisfy a wide range of information need on a given topic. We term this model as a *Combined Model* since it combines TAC ontologies with the random walk paradigm. The whole procedure can be again formulated according to a Markov Chain principle described in Section 5.4.2 except the fact that the node(sentence) weights will also include the weights obtained by using the list of aspects' information as defined in Section 5.4.2. Figure 5.2 shows a part of the graph with node and edge weights (after applying the combined model) for the top ranking sentences that were chosen by random walk. This is an example of a DUC-2006 topic outlined below.

```
<topic id = "D0626H" category = "2">
<title> bombing of US embassies in Africa  </title>
S1: Among them is Saudi dissident Osama
bin Laden, who allegedly runs al Qaida,
a radical Islamic network accused of
planning the bombings.
S2: In an interview Tuesday, Home Affairs
Minister Ali Ameir Mohamed likened Ahmed
to a chameleon.
S3: It said Khalid, who can not speak English
or Kiswahili but only Arabic, was identified
by a guard and a civilian worker at the embassy and
a third witness.
S4: Although no details were released in court,
local media said traces of chemicals that could
have been used to make the bomb had been
found in Saleh's home and car.
S5: The action contrasted markedly to a
decision by Kenya, where the American
Embassy was bombed on the same day.
```

From Figure 5.2, we get to the fact that initially, sentence S_1 is chosen into the candidate summary as it has the highest node (sentence) weight, then, by performing a random walk based on the transition probabilities of the Markov chain model, we find S_2 as the next candidate sentence, then, S_5, S_4, S_3 and so on. The random walk stops after the k steps which is related to reaching the summary-length of 250 words.

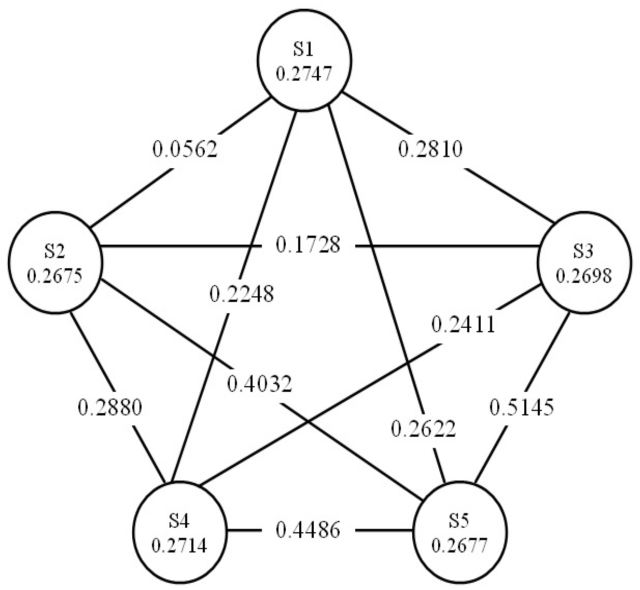

Figure 5.2: TAC ontologies with random walk model

5.4.3 Evaluation Results and Analysis

In this research, we run our experiments using a subset of TAC-2010 and DUC-2006 data applying three different models to generate three candidate summaries for each topic.

Manual Evaluation

Some university graduate students judged the summaries for linguistic quality and overall responsiveness. The given score is an integer between 1 (very poor) and 5 (very good) and is guided by consideration of the following factors: 1. Grammaticality, 2. Non-redundancy, 3. Referential clarity, 4. Focus and 5. Structure and Coherence. They also assigned a content responsiveness score to each of the automatic summaries. The content score is an integer between 1 (very poor) and 5 (very good) and is based on the amount of information in the summary that helps to satisfy the information need expressed in the topic. These measures were used at DUC 2007. Table 5.12 and Table 5.13 presents the average linguistic quality and overall responsive scores of all the systems on TAC-2010 data and DUC-2006 data, respectively. While presenting the results, we highlight the top scores to indicate significance at a glance.

Systems	Linguistic Quality	Overall Responsiveness
TAC ontology	**4.00**	**4.00**
Random walk	3.6	3.00
Combined	**4.00**	3.00

Table 5.12: Linguistic quality and responsiveness scores (TAC-2010 data)

Systems	Linguistic Quality	Overall Responsiveness
TAC ontology	3.72	3.00
Random walk	3.52	3.00
Combined	**3.76**	**3.20**

Table 5.13: Linguistic quality and responsiveness scores (DUC-2006 data)

Analyzing the results of the manual evaluation yields the fact that augmenting TAC ontologies with the random walk model often outperforms the random walk model alone in terms of linguistic quality and responsiveness scores. This is because, the use of TAC ontologies enhances the coverage of the information that is necessary to satisfy the quest of the users.

Automatic Evaluation

For the DUC-2006 data subset, we carried out automatic evaluation of our candidate summaries using ROUGE 3.1.4 toolkit. For all our systems, we report the widely accepted important metrics: ROUGE-1, ROUGE-2 and ROUGE-SU. We also show 95% confidence interval of the important evaluation metrics for our systems to report significance for doing meaningful comparison. Table 5.14 to Table 5.16 show the ROUGE-1, ROUGE-2, and ROUGE-SU scores of our three different summary generation models.

Scores	TAC ontology	Random Walk	Combined
Recall	0.347148	0.334269	**0.361347**
Precision	0.342975	**0.360458**	0.359846
F-score	0.346982	0.339435	**0.358654**

Table 5.14: ROUGE-1 measures

Scores	TAC ontology	Random Walk	Combined
Recall	**0.072359**	0.051764	0.064682
Precision	**0.068935**	0.055743	0.061967
F-score	**0.070964**	0.054753	0.061863

Table 5.15: ROUGE-2 measures

Scores	TAC ontology	Random Walk	Combined
Recall	0.116964	0.101865	**0.126246**
Precision	0.117532	**0.119176**	0.118645
F-score	0.114752	0.109165	**0.116937**

Table 5.16: ROUGE-SU measures

For all the three systems, Table 5.17 shows the F-scores of the reported ROUGE measures while Table 5.18 reports the 95% confidence intervals of the important ROUGE measures.

Table 5.17 clearly shows that the **Combined** system improves the ROUGE-1, ROUGE-2, and ROUGE-SU scores over the **Random walk** system by 3.67%, 19.22%, and 8.21%, respectively, whereas, it could not beat the ROUGE-2 score of **TAC ontology-based** system but improves the ROUGE-1, and ROUGE-SU scores by 4.15%, and 4.97%, respectively. These results suggest that augmenting TAC ontologies with the random walk model

Systems	ROUGE-1	ROUGE-2	ROUGE-SU
TAC ontology	0.346982	**0.070964**	0.114752
Random walk	0.339435	0.054753	0.109165
Combined	**0.358654**	0.061863	**0.116937**

Table 5.17: ROUGE-F scores for different systems

Systems	ROUGE-2	ROUGE-SU
TAC ontology	0.055273 - 0.084321	0.106424 - 0.119059
Random walk	0.036383 - 0.068463	0.089032 - 0.126243
Combined	0.036443 - 0.087892	0.098966 - 0.136364

Table 5.18: 95% confidence intervals for different systems

provides a better content coverage to satisfy the information need. On the other hand, Table 5.18 shows a high overlap between the confidence intervals of different systems. The proposed methods are also compared with a *Baseline* system. The *Baseline* is the official baseline system established in DUC-2006. We also list the average ROUGE scores of all the participating systems of DUC-2006 (i.e. *DUC-Average*). Table 5.19 presents this comparison which denotes that the **Combined** system improves the ROUGE-1, and ROUGE-2 scores over the **Baseline** system by 11.77%, and 17.78%, respectively, whereas, it performs closely to the average DUC-2006 systems.

Systems	ROUGE-1	ROUGE-2
TAC ontology	0.346982	0.070964
Random walk	0.339435	0.054753
Combined	0.358654	0.061863
Baseline	0.32095	0.05269
DUC-Average	0.37789	0.07483

Table 5.19: Comparison with DUC-2006 systems

5.5 Multi-Modality Manifold-Ranking

5.5.1 Introduction

The use of the multi-modality manifold-ranking algorithm for extracting topic-focused summary from multiple documents is proved to be very successful. In this method, the pair-wise similarity values between sentences are computed using the standard cosine similarity measure (TF*IDF). However, the major limitation of the TF*IDF approach is that it only retains the frequency of the words and disregards the syntactic and semantic information. we propose the use of syntactic and shallow semantic kernels in the multi-modality manifold-ranking algorithm for computing the relatedness between the sentences. Extensive experiments on the DUC benchmark datasets show the effectiveness of our approach.

In recent years, a variety of manifold-ranking based methods are applied successfully to topic-focused multi-document summarization. The basic manifold-ranking method is a typical graph-based summarization method that makes uniform use of the sentence-to-sentence relationships and the sentence-to-topic relationships in a manifold-ranking process (Wan et al., 2007a). In the multi-modality manifold-ranking algorithm, sentence relationships are classified into within-document relationships and cross-document relationships, and each kind of relationships are considered as a separate modality (graph) (Wan and Xiao, 2009). In these methods, the pair-wise similarity values between the sentences are computed using the standard cosine measure (TF*IDF). However, the major limitation of the TF*IDF approach is that it only retains the frequency of the words and does not take into account the sequence, syntactic and semantic structure. Thus, it cannot distinguish between “The police shot the gunman” and “The gunman shot the police”. For the task like *multi-document summarization* that requires the use of more complex syntactic and semantics, the approaches with only the standard cosine similarity measure are often inadequate to perform fine-level textual analysis.

We extensively study the impact of syntactic and semantic information in computing the similarity between the sentences in the multi-modality manifold learning framework for topic-focused multi-document summarization. We believe that the similarity measures based on the syntactic and semantic information could be helpful to characterize the relation between the sentences in a more effective way than the traditional TF*IDF based similarity measures. We run our experiments on the DUC-2006 benchmark dataset, and the results show the effectiveness of our approach.

5.5.2 Multi-Modality Manifold Ranking Model

The manifold-ranking method (Zhou et al., 2003a;b) is a universal ranking algorithm that is initially used to rank data points. This method has been successfully used for topic-focused document summarization in (Wan et al., 2007a) where the data points refer to the topic description and all the sentences in the documents. The manifold-ranking process for the summarization task can be formalized as follows (Wan and Xiao, 2009):

Given a set of data points $\chi = \{x_0, x_1, \cdots, x_n\} \subset R^m$, the first point x_0 represents the topic description (query point) and the rest n points represent all the sentences in the documents (data points to be ranked). The basic manifold-ranking algorithm makes uniform use of the sentence relationships in a single modality (Wan et al., 2007a). However, in (Wan and Xiao, 2009), the relationships between the sentences in a document set are classified as either within-document relationship or cross-document relationship to form two separate modalities to reflect the local information channel and the global information channel between the sentences, respectively. The two modalities are applied in the multi-modality manifold-ranking algorithm for ranking the sentences effectively. Based on each kind of modality, an undirected graph is built to reflect each kind of sentence relationships. Let

$W^a = \left[W^a_{ij}\right]_{(n+1)\times(n+1)}$ be the within-document affinity matrix containing only the within-document links for the $n+1$ data points, where W^a_{ij} is the cosine similarity value between x_i and x_j if x_i and x_j belong to the same document or one of x_i and x_j is x_0; Otherwise, W^a_{ij} is set to 0. Similarly, let $W^b = \left[W^b_{ij}\right]_{(n+1)\times(n+1)}$ be the cross-document affinity matrix containing the cross-document links, where W^b_{ij} is the cosine similarity value between x_i and x_j if x_i and x_j belong to different documents or one of x_i and x_j is x_0; Otherwise, W^b_{ij} is set to 0. All the relationships between the topic, x_0 and any document sentence $x_i\,(i \geq 1)$ are included in both W^a and W^b. Then, W^a and W^b are normalized by $S^a = (D^a)^{-\frac{1}{2}} W^a (D^a)^{-\frac{1}{2}}$ and $S^b = \left(D^b\right)^{-\frac{1}{2}} W^b (D^a)^{-\frac{1}{2}}$, respectively, where D^a and D^b are the diagonal matrices with (i,i)-element equal to the sum of the ith row of W^a and W^b, respectively. Then the multi-modality learning task for topic-focused summarization is to infer the ranking function f from W^a, W^b and y: $\left\{(W^a,\, D^a,\, S^a);\; \left(W^b,\, D^b,\, S^b\right);\; y\right\} \to f$.

Linear Fusion: For fusing the two modalities, we use the linear fusion scheme as this was shown to perform the best in (Wan and Xiao, 2009). This scheme fuses the constraints from S^a, S^b and y simultaneously by a weighted sum. The cost function associated with f is defined as:

$$Q(f) = \mu \cdot \sum_{i,j=0}^{n} W^a_{ij} \left| \frac{1}{\sqrt{D^a_{ii}}} f_i - \frac{1}{\sqrt{D^a_{jj}}} f_j \right|^2 + \eta \cdot \sum_{i,j=0}^{n} W^b_{ij} \left| \frac{1}{\sqrt{D^b_{ii}}} f_i - \frac{1}{\sqrt{D^b_{jj}}} f_j \right|^2 + \theta \cdot \sum_{i=0}^{n} |f_i - y_i|^2,$$

where μ, η, and θ capture the trade-off between the constraints[14].

we claim that for a complex task like topic-focused multi-document summarization where the relatedness between the document sentences is an important factor, the multi-

[14]The first two terms of the right-hand side in the cost function are the smoothness constraints for the two modalities, and the last term is the fitting constraint, respectively.

modality manifold algorithm for ranking sentences would perform more effectively if we could encode the syntactic and semantic information instead of just the standard cosine measure (i.e. TF*IDF) in calculating the similarity between sentences. In the next section, we describe how we can encode syntactic and semantic structures in calculating the similarity between sentences.

5.5.3 Syntactic and Shallow Semantic Structures

Given a sentence (or query), we first parse it into a syntactic tree using a parser like (Charniak, 1999) and then, calculate the similarity between the two trees using the *tree kernel* (discussed in Section 5.5.3). However, syntactic information is not often adequate when dealing with long and articulated sentences or paragraphs. Shallow semantic representations, bearing a more compact information, could prevent the sparseness of deep structural approaches (Moschitti et al., 2007). Initiatives such as PropBank (PB) (Kingsbury and Palmer, 2002) have made possible the design of accurate automatic Semantic Role Labeling (SRL) systems like ASSERT (Hacioglu et al., 2004).

For example, consider the PB annotation:

```
[ARG0 all][TARGET use][ARG1 the french
franc][ARG2 as their currency]
```

Such annotation can be used to design a shallow semantic representation that can be matched against other semantically similar sentences, e.g.

```
[ARG0 the Vatican][TARGET use][ARG1 the
Italian lira][ARG2 as their currency]
```

In order to calculate the semantic similarity between the sentences, we first represent the annotated sentence (or query) using the tree structures like Figure 5.3 which we call

Figure 5.3: Example of semantic trees

Semantic Tree (ST). In the semantic tree, arguments are replaced with the most important word-often referred to as the semantic head. We look for noun first, then verb, then adjective, then adverb to find the semantic head in the argument. If none of these is present, we take the first word of the argument as the semantic head. This reduces the data sparseness with respect to a typical cosine measure representation.

Tree Kernels

Once we build the trees (syntactic or semantic), our next task is to measure the similarity between the trees. For this, every tree T is represented by an m-dimensional vector $v(T) = (v_1(T),\ v_2(T),\ \cdots\ v_m(T))$, where the i-th element $v_i(T)$ is the number of occurrences of the i-th tree fragment in tree T. The tree fragments of a tree are all of its sub-trees which include at least one production with the restriction that no production rules can be broken into incomplete parts. Figure 5.4 shows an example tree and a portion of its subtrees.

Implicitly we enumerate all the possible tree fragments $1,\ 2,\ \cdots,\ m$. These fragments are the axis of this m-dimensional space. Note that this could be done only implicitly, since the number m is extremely large. Because of this, (Collins and Duffy, 2001) defines the

Figure 5.4: (a) An example tree (b) The sub-trees of the NP covering “the press”.

tree kernel algorithm whose computational complexity does not depend on m.

The tree kernel of two trees T_1 and T_2 is actually the inner product of $v(T_1)$ and $v(T_2)$:

$$TK(T_1,\, T_2) = v(T_1).v(T_2). \tag{5.13}$$

We define the indicator function $I_i(n)$ to be 1 if the sub-tree i is seen rooted at node n and 0 otherwise. It follows:

$$v_i(T_1) = \sum_{n_1 \in N_1} I_i(n_1)$$
$$v_i(T_2) = \sum_{n_2 \in N_2} I_i(n_2),$$

where, N_1 and N_2 are the set of nodes in T_1 and T_2 respectively. So, we can derive:

$$
\begin{aligned}
TK(T_1,T_2) &= v(T_1).v(T_2) \\
&= \sum_i v_i(T_1)v_i(T_2) \\
&= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \sum_i I_i(n_1)I_i(n_2) \\
&= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} C(n_1,n_2), \qquad (5.14)
\end{aligned}
$$

where, we define $C(n_1,\, n_2) = \sum_i I_i(n_1)I_i(n_2)$. Next, we note that $C(n_1,\, n_2)$ can be computed in polynomial time, due to the following recursive definition:

- If the productions at n_1 and n_2 are different then $C(n_1,\, n_2) = 0$
- If the productions at n_1 and n_2 are the same, and n_1 and n_2 are pre-terminals, then $C(n_1,\, n_2) = 1$
- Else if the productions at n_1 and n_2 are not pre-terminals,

$$
C(n_1,\, n_2) = \prod_{j=1}^{nc(n_1)} (1 + C(ch(n_1,\, j), ch(n_2,\, j))), \qquad (5.15)
$$

where, $nc(n_1)$ is the number of children of n_1 in the tree; because the productions at n_1 and n_2 are the same, we have $nc(n_1) = nc(n_2)$. The i-th child-node of n_1 is $ch(n_1,\, i)$. Note that, the tree kernel (TK) function computes the number of common subtrees between two trees. Such subtrees are subject to the constraint that their nodes are taken with all or none of the children they have in the original tree. Though, this definition of subtrees makes the TK function appropriate for syntactic trees but at the same time makes it not well suited for the semantic trees (ST). The critical aspect of steps (1), (2) and (3) of the TK function is that the productions of two evaluated nodes have to be identical to allow the match of further descendants. This means that common substructures cannot be composed by a node with only some of its children as an effective ST representation would require. Moschitti et al. (2007) solve this problem by designing the Shallow Semantic Tree Kernel (SSTK) which allows to match portions of a ST.

Shallow Semantic Tree Kernel (SSTK)

The SSTK is based on two ideas: first, it changes the ST by adding *SLOT* nodes. These accommodate argument labels in a specific order i.e. it provides a fixed number of slots, possibly filled with *null* arguments, that encode all possible predicate arguments. Leaf nodes are filled with the wildcard character * but they may alternatively accommodate additional information. The slot nodes are used in such a way that the adopted TK function can generate fragments containing one or more children. As previously pointed out, if the arguments were directly attached to the root node, the kernel function would only generate the structure with all children (or the structure with no children, i.e. empty). Second, as the original tree kernel would generate many matches with slots filled with the null label, we have set a new step 0 in the TK calculation:
(0) if n_1 (or n_2) is a pre-terminal node and its child label is *null*, $C(n_1,\, n_2) = 0$;

and subtract one unit to $C(n_1,\, n_2)$, in step 3:

$$C(n_1,\, n_2) = \prod_{j=1}^{nc(n_1)} (1 \,+\, C(ch(n_1,\, j),\, ch(n_2,\, j))) \,-\, 1. \tag{5.16}$$

The above changes generate a new C which, when substituted (in place of original C) in Eq. 5.14, gives the new SSTK.

5.5.4 Experiments and Results

We reimplement the multi-modality manifold ranking algorithm for topic-focused multi-document summarization by encoding the syntactic and semantic information to measure sentence relationships. We use the linear approach for fusing the modalities as this was shown to perform the best (Wan and Xiao, 2009). The purpose of our experiments is to study the impact of the syntactic and semantic representation introduced earlier for the summarization task. Besides using tree kernel functions for incorporating syntactic and shallow semantic structures, we also measure the sentence similarities using deep semantic analysis. For each sentence, we perform keyword expansion using WordNet to find out similar words between the sentences that gives us a similarity score.

We used the main task of DUC 2006 for evaluation 5.2. To accomplish this task, we generate summaries for 25 topics of DUC 2006 by each of our three systems as defined below:

(1) SYN: This system measures the similarity between the sentences using the *syntactic tree* and the *general tree kernel* function defined in Section 5.5.3.

(2) SHALLOW-SEM: This system measures the similarity between the sentences using the *shallow semantic tree* and the *shallow semantic tree kernel* function defined in Section 5.5.3.

(3) DEEP-SEM: This system measures the similarity between the sentences using deep semantic analysis using WordNet.

In all these experiments, the topic description is considered as a single query point, and it is processed in the same way as other sentences. In another three experiments, we introduce more query points following the guided-summarization strategy defined in the 2010 Text Analysis Conference (TAC[15]). In the guided summarization framework, each

[15]http://www.nist.gov/tac/2010/Summarization/Guided-Summ.2010.guidelines.html

topic falls into a predefined category and a list of aspects for each category is given. We use this list of aspects as additional query points in the manifold-ranking framework to conduct three more experiments forming the guided versions of the similarity measures discussed above.

Evaluation

We carried out automatic evaluation of our candidate summaries using ROUGE 3.1.4 toolkit, which has been widely adopted for automatic summarization evaluation. For all our systems in this experiment, we report the widely accepted important metrics: ROUGE-1, ROUGE-2 and ROUGE-SU.

We also show 95% confidence interval of the important evaluation metrics for our systems to report significance for doing meaningful comparison. Table 5.20 to Table 5.22 show the ROUGE-1, ROUGE-2, and ROUGE-SU scores of our six different systems. In the experiments, the regularized parameter for the fitting constraint is fixed at 0.4, as in (Wan et al., 2007a). We kept $\mu = \eta = 0.3$ as it was shown to be the most effective choice for the linear fusion scheme in (Wan and Xiao, 2009).

Systems	Recall	Precision	F-score
SYN	0.3571	0.3105	0.3320
SHALLOW-SEM	0.3814	0.2909	0.3299
DEEP-SEM	0.3889	0.2935	0.3344
SYN-GUIDED	0.3649	0.3124	0.3365
SHALLOW-GUIDED	0.3833	0.2906	0.3302
DEEP-GUIDED	0.3962	0.3006	0.3417

Table 5.20: ROUGE-1 measures

For all the systems, Table 5.23 shows the F-scores of the reported ROUGE measures

Systems	Recall	Precision	F-score
SYN	0.0638	0.0558	0.0595
SHALLOW-SEM	0.0732	0.0555	0.0631
DEEP-SEM	0.0665	0.0501	0.0571
SYN-GUIDED	0.0564	0.0481	0.0519
SHALLOW-GUIDED	0.0684	0.0519	0.0590
DEEP-GUIDED	0.0662	0.0503	0.0572

Table 5.21: ROUGE-2 measures

Systems	Recall	Precision	F-score
SYN	0.1190	0.0903	0.1025
SHALLOW-SEM	0.1406	0.0818	0.1033
DEEP-SEM	0.1452	0.0829	0.1054
SYN-GUIDED	0.1287	0.0946	0.1089
SHALLOW-GUIDED	0.1408	0.0815	0.1029
DEEP-GUIDED	0.1487	0.0859	0.1088

Table 5.22: ROUGE-SU measures

while Table 5.24 reports the 95% confidence intervals of the important ROUGE measures.

Systems	R-1	R-2	R-SU
SYN	0.3320	0.0595	0.1025
SHALLOW-SEM	0.3299	0.0631	0.1033
DEEP-SEM	0.3344	0.0571	0.1054
SYN-GUIDED	0.3365	0.0519	0.1089
SHALLOW-GUIDED	0.3302	0.0590	0.1029
DEEP-GUIDED	0.3417	0.0572	0.1088

Table 5.23: ROUGE-F scores for different systems

Systems	R-2	R-SU
SYN	0.0439 - 0.0802	0.0845 - 0.1313
SHALLOW-SEM	0.0530 - 0.0753	0.0928 - 0.1128
DEEP-SEM	0.0487 - 0.0652	0.0957 - 0.1142
SYN-GUIDED	0.0356 - 0.0636	0.1063 - 0.1107
SHALLOW-GUIDED	0.0501 - 0.0678	0.0931 - 0.1127
DEEP-GUIDED	0.0478 - 0.0663	0.0991 - 0.1183

Table 5.24: 95% confidence intervals for different systems

In Table 5.25, the proposed methods are compared with the basic manifold-ranking method (i.e. “BASIC”), standard cosine similarity based multi-modality ranking method (i.e. “COSINE”) and the NIST baseline. The NIST baseline is the official baseline system established by NIST. We also list the average ROUGE scores of all the participating systems for DUC 2006 (i.e. AverageDUC). From the tables, we can see that the proposed multi-modality manifold ranking methods based on the syntactic and semantic measures mostly outperform the NIST baseline system. They can also achieve higher ROUGE scores as comparable to the average scores of all the participating systems of DUC 2006. Our proposed systems also perform closely to the basic manifold ranking system and the standard cosine measure based multi-modality ranking method. The results also show that the guided systems often perform better than their unguided counterpart denoting the fact that the addition of the list of aspects (to increase the number of query points) improves the overall performance of the manifold ranking process.

Systems	ROUGE-1	ROUGE-2
SYN	0.3320	0.0595
SHALLOW-SEM	0.3299	0.0631
DEEP-SEM	0.3344	0.0571
SYN-GUIDED	0.3365	0.0519
SHALLOW-GUIDED	0.3302	0.0590
DEEP-GUIDED	0.3417	0.0572
BASIC	0.3953	0.0833
COSINE	0.4030	0.0850
Baseline	0.3209	0.0526
AverageDUC	0.3778	0.0748

Table 5.25: System comparison (F-scores)

5.6 Question Decomposition

5.6.1 Introduction

A complex question is a question asking about events, biographies, definitions, descriptions, reasons etc. which might not be answered by a single entity or even a single sentence as Question Answering (QA) systems often need to deal with complex information needs for this purpose. Previous works have demonstrated that decomposing a complex question into a series of simple questions and then reusing the techniques developed for answering simple questions is an effective means of dealing with the complex question answering problem. However, no study has developed any methods that can judge the significance of the decomposed questions disregarding the fact that good decomposed questions are the prerequisites for more relevant and more accurate answers. we address this challenge and propose a supervised model to learn good decompositions from complex questions. Extensive experiments and evaluations show promising results to prove the effectiveness of our approach.

Decomposing a complex question into simpler questions such that each of which can be answered individually, then using state-of-the-art QA systems to get individual answers, and combining them finally into a single answer to the original complex question has been proved to be an effective way of dealing with the complex question answering problem. For example, Harabagiu et al. (2006) proposed a method of processing complex questions that relies on a combination of (a) question decompositions; (b) factoid QA techniques; and (c) Multi-Document Summarization (MDS) techniques. The question decomposition procedure operates on a Markov chain, by following a random walk with mixture model on a bipartite graph of relations established between concepts related to the topic of a complex question and subquestions derived from topic-relevant passages that manifest these relations. Decomposed questions are then submitted to a state-of-the-art QA system in order to retrieve a set of passages that can later be merged into a comprehensive answer by a MDS system. They show that question decompositions using this method can significantly enhance the relevance and comprehensiveness of summary-length answers to complex questions. Necessity and positive impact of question decomposition have been also shown in many researches in the complex question answering domain (Lacatusu et al., 2006, Saquete et al., 2004, Hickl et al., 2006). However, issues like judging the significance of decomposed questions remained beyond the scope of all these researches till date. It is understandable that if we can enhance the quality of the decomposed questions, final answer to the original complex question would be more accurate. This will obviously require understanding the quality of the decomposed questions and then, investigating more methods to enhance the quality. In this research, we address this challenging task and come up with a supervised model of automatically learning good decompositions to complex questions which can then be passed to the traditional QA systems for more accurate answers.

To find answers for complex type of questions, it is important at first to know what information is relevant to an event, biography, definition, description or reason type question. Therefore, we assume that a set of relevant data set is given along with a complex question that certainly possesses potential answers to the complex question. However, it is still necessary to filter out the most important sentences from the given data set (that are mostly relevant to contain potential answers) since the data set may have huge number of sentences and therefore, not suitable to act as a reasonably straightforward answer to the complex question. We conduct a shallow and a deep semantic analysis between the complex question and the given document sentences in order to find out the salient sentences that can be potential candidate answers. Sentences that are picked up during this process can be long and complex to deal with. Hence, we pass the selected sentences through a sentence simplification module. Once we find the simple sentences, we use a sentence-to-question generation approach in order to generate corresponding questions. We claim that these questions are the potential candidate decompositions of the complex question in consideration. Since automatic processing of sentences is a complex task, several modules of the question decomposition framework may be erroneous and hence, not producing 100% accurate output. This is what motivates us to judge the significance of the generated decompositions by hand. We manually annotate the decomposed questions into two classes: good candidate and bad candidate, using a rich feature set considering correctness at the question level along with a coverage component that measures whether a decomposed question can actually satisfy the information need stated in the original complex question partially. We employ two supervised models: Support Vector Machines (SVM) and Maximum Entropy (MaxEnt) that are trained on this annotated data set and then, we use the learned model to produce good decompositions from the complex question automatically. Learning good decompositions from complex questions is unique and to the best of our knowledge, no other study has investigated this challenge before in our setting. Extensive experimental analysis shows significance and effectiveness of our approach in the complex question answering domain.

5.6.2 Filtering Important Sentences

Techniques to measure similarities between texts can be an effective way of judging the importance of a sentence. Word dependencies having an important role in finding similarity between two texts can be discovered using a syntactic parser. Pasca and Harabagiu (2001) demonstrated that with the syntactic form one can see which words depend on other words while successful use of syntactic features have been shown so far in *question answering* (Zhang and Lee, 2003b, Moschitti et al., 2007, Moschitti and Basili, 2006). However, shallow semantic representations can bear a more compact information preventing the sparseness of deep structural approaches and the weakness of BOW models (Moschitti et al., 2007). By abstracting over surface syntactic configurations, semantic roles typically offer a significant first step towards deeper text understanding and hold promise for a range of applications requiring a broad coverage of semantic processing (Shen and Lapata, 2007). This motivates us to use a shallow and a deep semantic analysis in order to filter out the most important sentences related to the complex question from the given document collection.

Shallow Semantic Analysis

As similarity of a document sentence with the complex question relies on a deep understanding of the semantics of both, we get the feeling that an application of Semantic Role

Labeling (SRL) system to filter important sentences might suit well. To experiment with semantic structures, we parse the corresponding sentences (and the question) semantically using a Semantic Role Labeling (SRL) system called ASSERT 3.3.5. We represent the annotated sentences using tree structures called semantic trees (ST). In the semantic tree, arguments are replaced with the most important word, often referred to as the semantic head. We look for noun, then verb, then adjective, then adverb to find the semantic head in the argument. We then used Shallow Semantic Tree Kernel (SSTK) 5.5.3 to give a similarity score between a document sentence and the complex question based on their underlying semantic structures. For example, for the following sentence *s* and complex question *q* we get a semantic score:

Query (q): Describe steps taken and worldwide reaction prior to introduction of the Euro on January 1, 1999. Include predictions and expectations reported in the press.

Sentence (s): The Frankfurt-based body said in its annual report released today that it has decided on two themes for the new currency history of European civilization and abstract or concrete paintings.

Scores: 6, 12

Average Score: 9

Deep Semantic Analysis

Different words may have similar meaning in the sentences. This insists us towards deeper semantic analysis of the text. We use the *OAK* system (Sekine, 2002) to get the stemmed words. Then, we remove the stopwords from the sentences (and the complex question) using a stopword list. For each sentence and the complex question, we perform keyword expansion using WordNet[16]. For example, the word “happen” being a keyword in the question “What happened?” returns the words: *occur, pass, fall out, come about, take place* from WordNet. Thus, we find out the similar words between the sentence-question pair that gives us a similarity score.

5.6.3 Simplifying the Sentences

Sentences in the dataset may have complex grammatical structure with multiple embedded clauses. Therefore, we simplify the complex sentences by altering lexical items, syntactic structure, and semantics etc. with the intention to generate more accurate questions. We call the generated simple sentences as *elementary sentences* since they are the individual constituents that combinedly possess the overall meaning of the complex sentence in consideration. Although many existing NLP transformations such as sentence compression, paraphrase generation etc. could be exploited in this step, we extract the simpler forms of the complex source sentence by removing phrase types such as leading conjunctions, sentence-level modifying phrases, and appositives according to the model described in (Heilman and Smith, 2010b). To identify the syntactic structure of the sentences, we use Stanford Parser (Klein and Manning, 2003). We use *Tregex*, a tree query language (Levy and Andrew, 2006) to write expressions for extracting a set of elementary sentences from any finite clauses, relative clauses, appositives, and participial phrases that are present in the original complex sentence. For example, for the following complex sentence *s*, we get four elementary sentences:

Sentence (s): The student, who is supervised by Tom, arrived at my office, gave me a

[16]WordNet (http://wordnet.princeton.edu/) is a widely used semantic lexicon for the English language. It groups English words (i.e. nouns, verbs, adjectives and adverbs) into sets of synonyms called synsets, provides short, general definitions (i.e. gloss definition), and records the various semantic relations between these synonym sets.

presentation and left unhappy.

Elementary Sentences: 1) The student is supervised by Tom. 2) The student arrived at my office. 3) The student gave me a presentation. 4) The student left unhappy.

5.6.4 Sentence-to-Question Generation

Once we get the simplified (i.e. elementary) sentences, our next task is to produce a set of possible questions from them. We claim these questions to be the candidate decompositions of the original complex question. In this research, we work on generating six simple types of questions[17]: *who,what,where,when,whom* and *how much*. We identify the candidate answer terms from the input simple sentences and replace them with suitable question words in order to generate questions. Inspired by (Mitkov and Ha, 2003, Heilman and Smith, 2010a), we encode a considerable amount of linguistic knowledge by forming a set of general-purpose rules in order to transform the simple sentences into suitable questions. In the first step, we use Stanford Parser to syntactically parse each input sentence for identifying the noun phrases (NP) and prepositional phrases (PP) that we treat as the possible candidate answer terms. Then, we decompose the main verb to deal with the tense information before inverting the subject and auxiliary verb and finally, we classify the sentence considering an opposite approach to (Roth et al., 2002) for inserting a suitable question word in place of the candidate answer term to generate a possible question.

We use OAK system (Sekine, 2002) to produce Part-Of-Speech (POS) tagged sentences. The POS tagged sentences gives us information about the verbs and their tenses. We extract all the verbs from a sentence based on this information, stem them and process further to perform necessary subject-auxiliary verb inversion. Again, we employ the OAK

[17]We restrict ourselves from generating *why* and *how* type questions since our intention is to generate simple questions as decompositions to a complex question.

system to generate Named Entity (NE) tagged sentences. NE information is necessary to identify the candidate answer terms in a sentence. A sentence may include a certain Named Entity type (among the 150 NEs defined in OAK system) such as: PERSON, LOCATION, ORGANIZATION, GPE (Geo-Political Entity), FACILITY, DATE, MONEY, PERCENT, TIME, etc. However, while manually inspecting the performance of the Oak system, we find that it cannot recognize certain NE classes. Therefore, we also use Stanford Named Entity Recognizer (NER) tool (Finkel et al., 2005) and exploit the output of both. We use these information to classify the sentences by following a two-layered taxonomy to represent a natural semantic classification for the sentences. Our sentence classifier module makes use of a sequence of two simple classifiers. The first classifies the sentences into fine classes (Fine Classifier) and the second into coarse classes (Coarse Classifier). This is a similar but opposite approach to the one described in (Roth et al., 2002). The second classifier influences the first in that its candidate labels are generated by reducing the set of retained fine classes from the first into a set of coarse classes. This set is treated as the confusion set for the first classifier, the confusion set keep shrinking till we find the coarse classes that the word belongs to. The NE information is included in a hierarchy and is used to make candidate fine and coarse classes. We define the five coarse classes as: ***1) Human:*** *Any subject or object that is a name of a person*, ***2) Entity:*** *Includes animals, plant, mountains and any object*, ***3) Location:*** *Words that represent locations, such as country, city, school, etc.*, ***4) Time:*** *Words that represent time, date or period such as year, Monday, 9 am, last week, etc.*, and ***5) Count:*** *Holds all the counted elements, such as 9 men, measurements like weight etc*. Let, the confusion set of any sentence be $C_0 = \{c_1, c_2, \cdots, c_n\}$, the set of all the coarse classes. Initially, the fine classifier determines the fine classes. Then, the set of fine classes is reduced to a coarse class determined by the class hierarchy. That is, the set $\{f_{i1}, f_{i2}, \cdots, f_{im}\}$ of fine classes is mapped into the coarse class c_i. Based on the coarse classification, we consider the relationship between the words in the sentence.

Thus, we process each sentence in a top-down manner to get it classified. We check the coarse classes according to some basic word-to-word interaction rules. Indeed, the sentence classification module outputs the question type that is used to replace the candidate answer term. We use the POS information to decompose the main verb and perform necessary subject-auxiliary inversion motivated by (Heilman and Smith, 2010a) and finally, insert the question word (with a question mark at the end) to generate suitable questions from the given elementary sentence. For example, for the following simple sentence *s*, we get one question generated as:

Sentence (s): Native Americans around the country are leaving reservations.
Generated Question: Who around the country are leaving reservations?

5.6.5 Supervised Model

Automatic processing of sentences is a complex task. In different stages of our question decomposition procedure (such as semantic parsing, syntactic parsing, POS tagging, NE tagging, elementary sentence extraction, question generation), we analyze the output and find not all of them as 100% accurate. This motivates us to introduce a supervised model that enables the system to judge the generated questions for better results and accuracy. For supervised learning techniques, annotated or labeled data is required as a precondition. We manually annotate the generated questions into two classes: good candidate and bad candidate considering a rich feature set, employ two supervised models: Support Vector Machines (SVM) and Maximum Entropy (MaxEnt) that are trained on this annotated data set and then, use the learned model to filter out good decompositions from the candidate set automatically. We describe our feature space, learning and testing modules in the following subsections.

Feature Space

We use a total of thirteen features that are divided into two major categories: one that considers the correctness at the question level and other is a coverage component that measures whether a decomposed question can actually satisfy the information need stated in the original complex question partially (if not fully). We consider these features manually by hand while annotating the training data set, on the other hand, we automatically extract these features from the questions (for both training and testing data) in order to feed them to the supervised models for learning and then, for prediction. To be specific, each question is scored according to several features related to the original sentence (that is filtered by semantic analysis), the input sentence (elementary sentence), the generated question at hand, the original complex question and a human-generated summary of the given data set (that is assumed to successfully answer the original complex question).

Correctness of Questions We manually inspect each question to measure whether they are lexically, syntactically and semantically correct or not. With the same intention, we automatically extract a composition of following features from the questions:

Grammaticality We count the number of proper nouns, pronouns, adjectives, adverbs, conjunctions, numbers, noun phrases, prepositional phrases, and subordinate clauses in the syntactic structures of the question and the input sentence. We set a certain threshold to denote the limit up to which a candidate can be termed as good. We also include some boolean features to encode the tense information of the main verb.

Length We calculate the number of tokens in the question, the original source sentence, the input elementary sentence, and the answer term (that is replaced by a question type). We set a threshold for this purpose, too.

Presence of Question Word We consider some boolean features to identify the presence or absence of a certain question type: *who,what,where,when,whom* and *how much*.

Presence of Pronouns If a question has one or more pronouns, we understand that the question is asking about something that has limited reference and hence, we consider the question as *vague*. To identify whether a question includes pronouns or not, we employ a boolean feature.

Coverage We manually judge each decomposed question against the original complex question and analyze further to find out whether they can ask about something that can be found in the given data (and/or in the given human-generated summary). This is the coverage component of our feature extraction module that can tell whether a decomposed question can somehow satisfy the requested information need. To automatically encode this feature for each question, we conduct an extensive linguistic analysis and a deep semantic analysis between the decomposed question and the original complex question in consideration.

Linguistic Analysis We use ROUGE (Recall-Oriented Understudy for Gisting Evaluation) to automatically determine the quality of a question by comparing it to the original complex question using a collection of measures (Lin, 2004). The ROUGE measures considered are: ROUGE-N (N = 1, 2, 3, 4), ROUGE-L, ROUGE-W and ROUGE-S.

Deep Semantic Analysis We conduct a deep semantic analysis between the decomposed question and the original complex question according to the procedure discussed in Section 5.6.2.

Learning and Testing

Once we get the feature values for all the decomposed questions along with the associated annotation (good or bad candidate), we feed this data[18] to the supervised learners so that a learned model is established. Later, this model is used to predict the labels for the unseen questions[19] during the testing phase. In this work, we use two well-known supervised machine learning techniques: Support Vector Machines (SVM) and Maximum Entropy (MaxEnt) as classifiers.

Support Vector Machines (SVM) SVM is a powerful methodology for solving machine learning problems introduced by Vapnik (Cortes and Vapnik, 1995) based on the Structural Risk Minimization principle. SVM finds the separating hyperplane that has maximum margin between the two classes in case of binary classification. SVMs can also handle non-linear decision surfaces introducing kernel functions. In this research, we use the default linear kernel function. To allow some flexibility in separating the classes, SVM models have a cost parameter, C, that controls the trade off between allowing training errors and forcing rigid margins. We use the default value for the trade off parameter C. We use the SVM^{light} (Joachims, 1998a) package[20] for training and testing in this work. SVM^{light} is an implementation of Support Vector Machine (Cortes and Vapnik, 1995) for the problems of pattern recognition, regression, and learning a ranking function. It consists of a learning module and a classification module. The learning module takes an input file containing the feature values with corresponding labels and produces a model file. The classification module is used to apply the learned model to new samples. We use $g(x)$, the normalized distance from the hyperplane to each sample point, x to rank the good questions.

[18]We use a set of 523 questions for training purpose.
[19]Our test data set includes 350 questions.
[20]http://svmlight.joachims.org/.

Maximum Entropy (MaxEnt) The main principle of the MaxEnt method is to model all that is known and assume nothing about that which is unknown. In other words, given a collection of facts, the model must be consistent with all the facts, but otherwise act as uniformly as possible (Berger et al., 1996). MaxEnt models can be termed as multinomial logistic regression if they are to classify the observations into more than two classes (Jurafsky and Martin, 2008). However, in this research, we used the MaxEnt model to classify the questions into two classes: good candidate or bad candidate. We build the MaxEnt system using Dr. Dekang Lin's MaxEnt package[21]. To define the exponential prior of the λ values[22] in MaxEnt models, an extra parameter α is used in the package during training. We keep the value of α as default. The output probability values corresponding to the predicted labels are used to rank the classified good questions.

5.6.6 Evaluation and Analysis

Corpus

We use DUC-2006 data (that came from the AQUAINT corpus, comprising newswire articles from the Associated Press and New York Times (1998-2000) and Xinhua News Agency (1996-2000)) to run our experiments in this research.

Manual Evaluation

During the testing phase of our experiments, supervised classifiers automatically output their predicted labels (good or bad) for the decomposed questions in consideration. To

[21] http://www.cs.ualberta.ca/~lindek/downloads.htm.

[22] λ is the associated weight for each feature, which is learned by the MaxEnt model using numerical optimization techniques.

evaluate the performance of the classifiers, we manually assess the quality of the generated questions[23]. Two university graduate students judged the questions for linguistic quality and overall responsiveness following a similar setting to the DUC-2007 evaluation guidelines[24]. The given score is an integer between 1 (very poor) and 5 (very good) and is guided by consideration of the following factors: 1. Grammaticality, 2. Correct question type, 3. Referential clarity (Presence of pronoun), and 4. Meaningfulness. They also assigned a content responsiveness score to each of question. This score is also an integer between 1 (very poor) and 5 (very good) and is based on the factor whether the question somehow helps to satisfy the information need expressed in the original complex question. We compare the top-ranked good questions with the performance of a set of randomly picked (good/bad mixed) questions. For each topic, we also judge the performance of the bad questions alone. Table 5.26 and Table 5.27 show the evaluation results for SVM and MaxEnt, respectively. Analyzing the Table 5.26, we see that SVM predicted *Good Questions* improve the linguistic quality and responsiveness scores over *Mixed (Random) Questions* by 74.06%, and 100.00%, respectively whereas they outperform the Linguistic Quality and Responsiveness scores over *Bad Questions* by 20.58%, and 12.50%, respectively. These results suggest that the SVM classifier performed well to rank the decomposed questions accurately. We also see that *Bad Questions* outperform the *Mixed (Random) Questions* in terms of linguistic quality because they were small in length and had good grammatical structure. However, they could not beat the responsiveness scores meaning that small questions have limited coverage over the requested information need. On the other hand, Table 5.27 yields that *Good Questions* underform the other two types of questions meaning the fact that MaxEnt performed poorly to classify the decomposed questions.

[23]We evaluated top 15% decomposed questions.

[24]http://www-nlpir.nist.gov/projects/duc/duc2007/quality-questions.txt.

Systems	Linguistic Quality	Responsiveness
Good Questions	4.10	3.60
Mixed (Random)	2.35	1.80
Bad Questions	3.40	1.60

Table 5.26: Linguistic quality and responsiveness scores (average) for SVM

Systems	Linguistic Quality	Responsiveness
Good Questions	2.60	1.90
Mixed (Random)	3.15	3.50
Bad Questions	3.90	3.10

Table 5.27: Linguistic quality and responsiveness scores (average) for MaxEnt

Impact of Features To analyze the impact of different features, we run another experiment considering the SVM classifier generated top-ranked questions. Table 5.28 shows detailed results. Grammaticality, Length, Pronoun, and Coverage in *Systems* column indicate that the corresponding feature is not considered during experiments, whereas All denotes the inclusion of all features. From these results, we understand that if we exclude the *Grammatically* feature, it does not affect the responsiveness score, but exclusion of *Length* produces better scores for both linguistic quality and responsiveness. On the other hand, if we do not consider the *Pronoun* feature, the scores have a negative impact. Again, omitting the *Coverage* feature decreases the responsiveness score badly and considering all the features yield a good linguistic quality while showing a moderate performance in terms of responsiveness score. In Figure 5.5, we plot a graph to show the performance of different systems considering several variants of feature space during experiments.

Variations	Linguistic Quality	Responsiveness
Grammaticality	3.70	4.60
Length	4.10	4.50
Pronoun	3.90	3.20
Coverage	4.00	3.10
All	4.10	3.60

Table 5.28: Impact of features

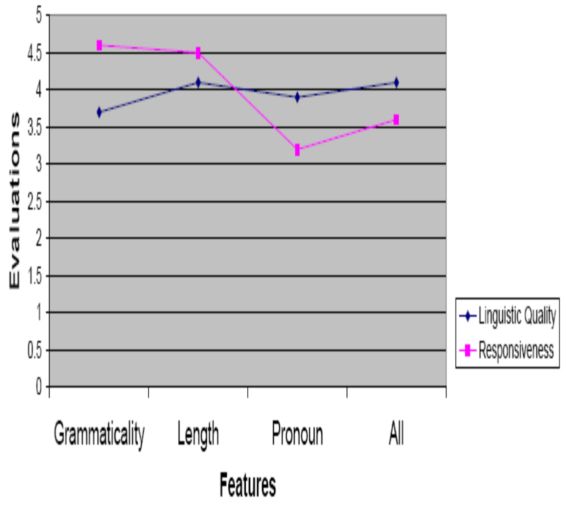

Figure 5.5: A graph plotting system variations considering different feature spaces

Automatic Evaluation

We pass the top-ranked decomposed questions to the Lemur toolkit[25]. The Lemur toolkit was developed jointly by the University of Massachusetts and Carnegie Mellon University to facilitate development of retrieval systems based on language models. The Lemur systems underlying architecture was built to support ad hoc and distributed retrieval with structured queries, cross-language IR, summarization, filtering, and categorization. For all decomposed questions, we formulate queries according to the Lemur toolkits input format and Lemur returns ranked sentences from the given data set in response to the queries. Thus, for each complex question, we generate a 250-word summary from the Lemur returned top-ranked sentences according to the DUC guidelines and evaluate the performance against four human-generated reference summaries given in DUC-2006. We evaluate the system generated summaries using the automatic evaluation toolkit ROUGE 3.1.4. ROUGE parameters were set as that of DUC-2007 evaluation setup. Thus, we investigate the performance of our supervised question decomposition systems. To show the impact of our question decomposition procedure, we compare our SVM and MaxEnt systems performance with their similar counterparts (SVM-WD and MaxEnt-WD[26]) that do not rely on question decomposition. We present the average ROUGE F-Scores of all our systems and a baseline system in Table 5.29. The baseline system generates summaries by returning all the leading sentences (up to 250 words) in the TEXT field of the most recent document(s). To compare our systems performance with the state-of-the-art systems, we also list the average ROUGE scores of all the participating systems of DUC-2006 and DUC-2007 (i.e. DUC-Avg-2006 and DUC-Avg-2007).

[25] Available at http://www.lemurproject.org
[26] WD stands for Without Decomposition

Systems	ROUGE-1	ROUGE-2
Baseline	0.3347	0.0640
DUC-Avg-2006	0.37789	0.07483
DUC-Avg-2007	0.40059	0.09550
SVM	0.3905	0.0867
MaxEnt	0.4006	0.0923
SVM-WD	0.3905	0.0867
MaxEnt-WD	0.4006	0.0923

Table 5.29: F-measures of different systems (Comparison)

Chapter 6

Conclusion

In this thesis we have investigated several methods for *complex question answering* which also can be rephrased as *topic focused multi document summarization*. We have formulated the complex question answering problem using reinforcement learning framework, which to the best of our knowledge has not been done before. We have also used unsupervised methods (manifold and random walk) for ranking and selecting document sentences for summarization and used question decomposition framework for answering complex question. We have also investigated the effect of different features in text summarization.

6.1 Reinforcement Learning

We presented a reinforcement learning formulation of the complex question answering problem. We proposed a modified version of the Watkins' $Q(\lambda)$ algorithm that is unique in how it represents the complex question answering task in the reinforcement learning framework. The main motivation of applying a reinforcement approach in this domain was to enhance real-time learning by treating the task as an interactive problem where user feedback can be aided as a reward. We simplified this assumption by not interacting with the users directly rather we employed the human-generated abstract summaries in order to provide a little amount of supervision using reward scores through textual similarity measurement. The similarity is measured using ROUGE.

We modeled the complex question answering problem in both MDP and POMDP. The main difference between MDP and POMDP is that in MDP the agent always knows which state it is in. This, combined with the Markov assumption for the transition model the optimal policy depends only on the current state. In case of POMDP the agent does not necessarily knows which state it is in. So the optimal action in a state *s* does not just depends on *s*, but also on how much the agent knows when it is in *s* (Russel and Norvig, 2003). As it is uncertain to tell whether a topic-oriented summary satisfies the user's information need or not, we argued that this uncertainty can be modeled if we consider the summarization task as a POMDP problem.

We compared the MDP-based system with a baseline and a SVM system and POMDP-based system with the NIST baseline and a MaxEnt-based system. The systems are evaluated using ROUGE and reported the significance of our results through 95% confidence intervals. Evaluation results show the effectiveness of applying the reinforcement approach for answering complex questions.

In this research, we kept the value of ε defined, through out the weight learning phase to denote a fixed probability of exploration. In future, we would experiment on tuning the value of ε where we will start by a high value to ensure more amount of exploration and less amount of exploitation while gradually decreasing this value to reduce exploration as time passes. We also plan to extend this research by experimenting on different text similarity measurement techniques such as Basic Element (BE) overlap (Hovy et al., 2006), syntactic similarity measure (Moschitti and Basili, 2006), semantic similarity measure (Moschitti et al., 2007), and Extended String Subsequence Kernel (ESSK) (Hirao et al., 2003) by using them as reward functions.

6.2 Augmenting TAC Ontologies with Random Walk Model

In this experiment, we presented a novel methodology to solve the topic-focused multi-document summarization task that uses a list of aspects (associated with each categorized topic of TAC-2010) that we call "TAC ontologies" in a random walk framework by performing a deeper semantic analysis of the source documents instead of relying only on document word frequencies to select important concepts. Experiments on several subsets of DUC-2006 and TAC-2010 data indicate that augmenting the TAC ontologies by the random walk model considerably outperforms the random walk model alone. This suggests the fact that TAC ontologies can provide a certain amount of supervision to cover all the relevant perspectives of a topic and hence, the use of it with any sophisticated model such as random walk can enhance the model's performance substantially in comparison to the model if used alone. In the future, we plan to use the TAC ontologies as a question decomposition model and focus to direct our research to find better decomposition methods that can aid to generate more accurate summary-length answers to complex questions.

6.3 Multi-Modality Manifold-Ranking

In this research, we proposed to encode the syntactic and semantic information for measuring sentence relationships in the multi-modality manifold ranking algorithm for topic-focused multi-document summarization. We parsed the sentences into the syntactic trees using the Charniak parser and applied the general tree kernel function to measure the similarity between sentences. We have redefined shallow semantic trees (STs and STNs) to represent predicate argument relations, which we automatically extracted using the ASSERT SRL system. We have used the shallow semantic tree kernel to measure the semantic similarity between two semantic trees. We also performed a deep semantic analysis in order to measure the sentence relationships. To the best of our knowledge, no other study has used syntactic and semantic information in the multi-modality manifold ranking algorithm before. We also propose to use more query points in the manifold ranking process by considering a list of aspects according to the guided summarization framework of TAC 2010. We evaluated our systems automatically using ROUGE and reported the significance of our results through 95% confidence intervals. Our systems also showed comparable results with respect to the state-of-the-art summarization systems.

6.4 Question Decomposition Framework

Here we proposed a supervised model of learning good decompositions from complex questions. Given a complex question and a set of relevant data set, our question decomposition framework filters out the most important sentences from the document collection using a shallow and deep semantic analysis. As these sentences may possess complex structures, we extracted elementary sentences from them in order to smoothen the question generation process. Then, these simple sentences are passed to the question generation module and using a set of general-purpose rules, we get a set of simple questions. We claim these questions to be the candidate decompositions of the original complex question in consideration. However, these questions may not be 100% accurate due to erroneous output in subsequent stages. So, we annotate the questions by hand considering several important features and produce a labeled data set. We employ two supervised learning techniques to learn from these data and then, they generate good decompositions from unseen complex questions automatically utilizing the learned model. To our knowledge, no other study has investigated the question decomposition task using a supervised model before. Evaluations and analysis show the effectiveness of our approach. Experiments on the DUC-2006 data set show that Support Vector Machines can be more useful than MaxEnt in this problem setting. We also show the impact of different features on performance of the SVM classifier. In future, we plan to use more sophisticated features with the intention to produce more accurate results.

Bibliography

C. Aone, M. E. Okurowski, J. Gorlinsky, and B. Larsen. A trainable summarizer with knowledge acquired from robust NLP techniques. In *Advances in Automatic Text Summarization*, pages 71–80, 1999.

A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

S. R. K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 82–90, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-45-9. URL `http://portal.acm.org/citation.cfm?id=1687878.1687892`.

E. Brill. Some advances in transformation-based part of speech tagging. In *National Conference on Artificial Intelligence*, pages 722–727, Seattle, Washington, 1994.

M. Cannataro and D. Talia. The knowledge grid. In *Communications of the ACM*, pages 46(1):89–93, 2003.

J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1998)*, pages 335–336, Melbourne, Australia, 1998.

A. Cassandra. A survey of POMDP applications. In *AAAI Fall Symposium*, pages 17–24, 1998.

Y. Chali, S. R. Joty, and S. A. Hasan. Complex question answering: Unsupervised learning approaches and experiments. *Journal of Artificial Intelligence Research*, 35:1–47, 2009.

Y. Chali, S. A. Hasan, and K. Imam. A reinforcement learning framework for answering complex questions. In *Proceedings of the 16th international conference on Intelligent user interfaces*, IUI '11, pages 307–310, New York, NY, USA, 2011a. ACM. ISBN 978-1-4503-0419-1. doi: http://doi.acm.org/10.1145/1943403.1943452. URL `http://doi.acm.org/10.1145/1943403.1943452`.

Y. Chali, S. A. Hasan, and K. Imam. Using semantic information to answer complex questions. In *Proceedings of the 24th Canadian Conference on Artificial Intelligence*, St. John's, Newfoundland and Labrador, Canada, 2011b.

E. Charniak. A maximum-entropy-inspired parser. In *Technical Report CS-99-12*, Brown University, Computer Science Department, 1999.

T. Clifton and W. Teahan. Bangor at TREC 2004: Question answering track. In *Proceedings of the Thirteenth Text Retreival Conference*, pages 426–435, 2004.

M. Collins and N. Duffy. Convolution kernels for natural language. In *Proceedings of Neural Information Processing Systems*, pages 625–632, Vancouver, Canada, 2001.

J. M. Conroy and D. P. Oleary. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 406–407, New York, NY, USA, 2001. ACM. ISBN 1-58113-331-6. URL `http://doi.acm.org/10.1145/383952.384042`.

C. Cortes and V. N. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

T. M. Cover and M. E. Hellman. The two-armed bandit problem with time-invariant finite memory. In *IEEE Transactions on Information Theory*, pages 16, 185–195, 1970.

K. Drummey, R. Donaway, and L. Mather. A comparison of rankings produced by summarization evaluation measures. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic Summarization*, NAACL-ANLP-AutoSum '00, pages 69–78, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1567564.1567572`.

D. A. Duverle and H. Prendinger. A novel discourse parser based on support vector machine classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL '09)*, pages 665–673, 2009. ISBN 978-1-932432-46-6.

H. P. Edmundson. New methods in automatic extracting. *Journal of the Association for Computing Machinery (ACM)*, 16(2):264–285, 1969.

B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. CRC Press, 1994.

G. Erkan and D. R. Radev. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479, 2004.

C. Fellbaum. WordNet - an electronic lexical database. Cambridge, MA, 1998. MIT Press.

L. Ferrier. A maximum entropy approach to text summarization. Master's thesis, University of Edinburgh, 2011.

J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, 2005.

W. N. Francis and H. Kucera. Frequency analysis of english usage. Houghton Mifflin, 1997.

R. Garside, G. Leech, and A. McEnery. *Corpus Annotation: linguistic information from computer text corpora*. Addison Wesley Longman, 1997.

S. Greenbaum, G. Leech, and J. Svartvik. Studies in english linguistics for randolph quirk. In *A tagged corpus - problems and prospects*, pages 192 – 209. Longman, 1979.

K. Hacioglu, S. Pradhan, W. Ward, J. H. Martin, and D. Jurafsky. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference/North American chapter of the Association for Computational Linguistics annual meeting (HLT/NAACL-2004)*, 2004. URL `http://www.stanford.edu/\~{}jurafsky/hlt-2004-verb.pdf`.

S. Harabagiu, F. Lacatusu, and A. Hickl. Answering complex questions with random walk models. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 220 – 227. ACM, 2006.

S. A. Hasan. Answering complex questions: Supervised approaches. Master's thesis, University of Lethbridge, 2009.

M. Heilman and N. A. Smith. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617, 2010a.

M. Heilman and N. A. Smith. Extracting simplified statements for factual question generation. In *Proceedings of the Third Workshop on Question Generation*, 2010b.

A. Hickl, P. Wang, J. Lehmann, and S. Harabagiu. Ferret: Interactive question-answering for real-world environments. In *Proceedings of the COLING/ACL on Interactive Presentation Sessions*, pages 25–28, 2006.

T. Hirao, H. Isozaki, E. Maeda, and Y. Matsumoto. Extracting important sentences with support vector machines. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 1–7, Taipei, Taiwan, 2002a.

T. Hirao, Y. Sasaki, H. Isozaki, and E. Maeda. NTT's text summarization system for duc 2002. In *Proceedings of the Document Understanding Conference*, pages 104–107, Philadelphia, Pennsylvania, USA, 2002b.

T. Hirao, J. Suzuki, H. Isozaki, and E. Maeda. NTT's multiple document summarization system for DUC 2003. In *Proceedings of the Document Understanding Conference*, Edmonton, Canada, 2003.

J. H. Holland. *Adaptation in natural and artificial systems*. MIT Press, Cambridge, MA, USA, 1992. ISBN 0-262-58111-6.

E. Hovy, C. Y. Lin, and L. Zhou. A be-based multi-document summarizer with query interpretation. In *Proceedings of the Document Understanding Conference*, Vancouver, B.C. Canada, 2005.

E. Hovy, C. Y. Lin, L. Zhou, and J. Fukumoto. Automated summarization evaluation with basic elements. In *Proceedings of the 5th Conference on Language Resources and Evaluation*, Genoa, Italy, 2006.

B. Howland, M. L. Minsky, and O. G. Selfridge. Hill-climbing: Some remarks on multiple simultaneous optimization. In *Group Report 54-15, Lincoln Laboratory, Massachusetts Institute of Technology*, 1960.

T. Joachims. *Making large-scale support vector machine learning practical*. MIT Press, Cambridge, MA, 1998a.

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning (ECML 1998).*, 1998b.

D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, 2008.

F. Karlsson, A. Voutilainen, J. Heikkil, and A. Anttila. *Constraint Grammar: A Language-Independent Framework for Parsing Unrestricted Text*. Mouton de Gruyter, Berlin / New York, 1995.

B. Katz, J. Lin, D. Loreto, W. Hildebrandt, M. Bilotti, S. Felshin, A. Fernandes, G. Marton, and F. Mora. Integrating web-based and corpus-based techniques for question answering. In *Proceedings of the Association for Computational Linguistics 39th Annual Meeting and 10th Conference of the European Chapter Workshop on Open-Domain Question Answering*, pages 426–435, 2003.

H. Kim, K. Kim, G. G. Lee, and J. Seo. MAYA: A fast question-answering system based on a predictive answer indexer. In *Proceedings of the Association for Computational Linguistics 39th Annual Meeting and 10th Conference of the European Chapter Workshop on Open-Domain Question Answering*, page 916, 2001.

P. Kingsbury and M. Palmer. From Treebank to PropBank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC)*, Las Palmas, Spain, 2002.

D. Klein and C. D. Manning. Fast exact inference with a factored model for natural language parsing. In *Proceedings of the Seventeenth Annual Conference on Neural Information Processing Systems (NIPS)*, pages 3–10. MIT Press, 2003.

A. Kotov and C. Zhai. Towards natural question guided search. In *Proceedings of the 19th International Conference on World Wide Web (WWW10)*, pages 541–550, 2010.

K. Koumpis and S. Renals. Automatic summarization of voicemail messages using lexical and prosodic features. *ACM Transactions on Speech and Language Processing*, 2, 2005. ISSN 1550-4875. URL `http://doi.acm.org/10.1145/1075389.1075390`.

T. Kudo and Y. Matsumoto. Chunking with support vector machines. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*, NAACL '01, pages 1–8, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics. URL `http://dx.doi.org/10.3115/1073336.1073361`.

J. Kupiec, J. Pedersen, and F. Chen. A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and Development in Information Retrieval*, SIGIR '95, pages 68–73, New York, NY, USA, 1995. ACM. ISBN 0-89791-714-6.

F. Lacatusu, A. Hickl, and S. Harabagiu. Impact of question decomposition on the quality of answer summaries. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, 2006.

J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 111–119, New York, NY, USA, 2001. ACM. ISBN 1-58113-331-6.

R. Levy and G. Andrew. Tregex and tsurgeon: Tools for querying and manipulating tree data structures. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC)*, 2006.

J. Li, L. Sun, C. Kit, and J. Webster. A query-focused multi-document summarizer based on lexical chains. In *Proceedings of the Document Understanding Conference*, Rochester, 2007.

C. Lin. Rouge: A package for automatic evaluation of summaries. In *Proceedings of Workshop on Text Summarization Branches Out, Post-Conference Workshop of Association for Computational Linguistics*, pages 74–81, Barcelona, Spain, 2004.

C. Y. Lin and E. Hovy. The automated acquisition of topic signatures for text summarization. In *Proceedings of the 18th Conference on Computational Linguistics*, pages 495–501, 2000. ISBN 1-55860-717-X.

C. Y. Lin and E. Hovy. From single to multi-document summarization: A prototype system and its evaluation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 457–464, Philadelphia, 2002.

C. Y. Lin and E. Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the Annual Meeting of the North American Association for Computational Linguistics*, pages 150–156, Edmonton, Canada, 2003.

D. J. Litman, M. S. Kearns, S. Singh, and M. A. Walker. Automatic optimization of dialogue management. In *Proceedings of the 18th conference on Computational linguistics - Volume 1*, COLING '00, pages 502–508, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. ISBN 1-55860-717-X.

I. Mani. *Automatic Summarization*. John Benjamins Co, Amsterdam/Philadelphia, 2001.

I. Mani, G. Klein, D. House, L. Hirschman, T. Firmin, and B. Sundheim. Summac: a text summarization evaluation. *Proceedings of the Natural Language Engineering*, 8: 43–68, March 2002. ISSN 1351-3249. URL `http://dl.acm.org/citation.cfm?id=973860.973864`.

W. C. Mann and S. A. Thompson. Rhetorical structure theory: Toward a functional theory of text organization. In *TEXT*, pages 8(3): 243–281, 1988.

D. Marcu. Improving summarization through rhetorical parsing tuning. In *Proceedings of The Sixth Workshop on Very Large Corpora*, pages 206 – 215, Montreal, Canada, 1998.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313 – 330, 1994.

K. Mckeown, R. J. Passonneau, D. K. Elson, Nenkova A., and J. Hirschberg. Do summaries help? In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 210–217, New York, NY, USA, 2005. ACM. ISBN 1-59593-034-5.

G. J. McMurtry. Adaptive optimization procedures. In *In Adaptive, Learning and Pattern Recognition Systems: Theory and Applications*, pages 243–286, 1970.

R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. In *Proceedings of the Conference of Empirical Methods in Natural Language Processing (EMNLP 2004)*, Barcelona, Spain, 2004.

M. L. Minsky and O. G. Selfridge. Learning in random nets. In *Proceedings of the Information Theory: Fourth London Symposium*, 1961.

R. Mitkov and L. A. Ha. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing*, pages 17–22, 2003.

A. Moschitti and R. Basili. A tree kernel approach to question and answer classification in question answering systems. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, Genoa, Italy, 2006.

A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar. Exploiting syntactic and shallow semantic kernels for question/answer classificaion. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 776 – 783, Prague, Czech Republic, 2007. ACL.

K. S. Narendra and M. A. L. Thathachar. Learning automata - a survey. In *IEEE Transactions on Systems, Man and Cybernetics*, pages 323–334, 1974.

V. Nastase. Topic-driven multi-document summarization with encyclopedic knowledge and spreading activation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, pages 763–772, 2008.

A. Nenkova. Summarization evaluation for text and speech: Issues and approaches. In *Proceedings of the Ninth International Conference on Spoken Language Processing (INTERSPEECH)*, Pittsburgh, PA, USA, 2006.

A. Nenkova and R. Passonneau. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, 2004.

K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *International Joint Conference on Artificial Intelligence (IJCAI-99) Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.

M. Osborne. Using maximum entropy for sentence extraction. In *Proceedings of the Association for Computational Linguistics (ACL) Workshop on Automatic Summarization*, pages 1–8, 2002.

J. Otterbacher, G. Erkan, and D. R. Radev. Using random walks for question-focused sentence retrieval. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 915–922, Vancouver, Canada, 2005.

M. A. Pasca and S. M. Harabagiu. High performance question/answering. In *Proceedings of the 24th Annual International ACL SIGIR Conference on Research and Development in Information Retrieval (SIGIR-2001)*, pages 366–374, 2001.

M. F. Porter. Readings in information retrieval. chapter An algorithm for suffix stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997. ISBN 1-55860-454-5. URL `http://dl.acm.org/citation.cfm?id=275537.275705`.

M. L. Puterman. *Markov Decision Processes–Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc, New York, 1994.

A. Resnick, G. J. Rath, and T. R. Savage. The formation of abstracts by the selection of sentences: Part 1: Sentence selection by man and machines. In *American Documentation*.

D. Roth, C. Cumby, X. Li, P. Morie, R. Nagarajan, N. Rizzolo, K. Small, and W. Yih. Question-answering via enhanced understanding of questions. In *Proceedings of the Eleventh Text REtreival Conference*, page 667, Gaithersburg, Maryland, 2002.

N. Roy, J. Pineau, and S. Thrun. Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 93–100, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

S. Russel and P. Norvig. *Artificial Intelligence A Modern Approach, 2nd Edition*. Prentice Hall, 2003.

E. Saquete, P. Martínez-Barco, R. Muñoz, and J. L. Vicedo. Splitting complex temporal questions for question answering systems. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, 2004.

K. Scheffler and S. Young. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of the Human Language Technology Conference (HLT)*, pages 12–19, 2002.

S. Sekine. Proteus project oak system (english sentence analyzer). 2002. URL `http://nlp.nyu.edu/oak`.

S. Sekine and C. A. Nobata. Sentence extraction with information extraction technique. In *Proceedings of the Document Understanding Conference (DUC 2001)*, New Orleans, Louisiana, USA, 2001.

D. Shen and M. Lapata. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 12–21, 2007.

D. Shen, J. Sun, H. Li, Q. Yang, and Z. Chen. Document summarization using conditional random fields. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 2862–2867, Hyderabad, India, 2007.

S. P. Singh, M. J. Kearns, D. J. Litman, and M. A. Walker. Reinforcement learning for spoken dialogue systems. In *Advances in Neural Information Processing Systems (NIPS)*, pages 956 – 962. MIT Press, 1999.

K. Sparck-Jones and J. Galliers. Evaluating natural language processing systems: An analysis and review. Springer-Verlag, 1996.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, Massachusetts, London, England, 1998.

W. J. Teahan. Knowing about knowledge: Towards a framework for knowledgeable agents and knowledge grids. Technical report, Artificial Intelligence and Intelligent Agents (AIIA 03.2), 2003.

C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 1979.

X. Wan and J. Xiao. Graph-based multi-modality learning for topic-focused multi-document summarization. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence (IJCAI-09)*, pages 1586–1591, 2009.

X. Wan, J. Yang, and J. Xiao. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-2007)*, pages 552–559, Prague, Czech Republic, 2007a.

X. Wan, J. Yang, and J. Xiao. Manifold-ranking based topic-focused multi-document summarization. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI-07)*, pages 2903–2908, 2007b.

S. Young. Using POMDPs for dialog management. In *Proceedings of the 1st IEEE/ACL Workshop on Spoken Language Technologies (SLT)*, pages 8 – 13, 2006.

A. Zhang and W. Lee. Question classification using support vector machines. In *Proceedings of the Special Interest Group on Information Retrieval*, pages 26–32, Toronto, Canada, 2003a. ACM.

D. Zhang and W. S. Lee. A language modeling approach to passage question answering. In *Proceedings of the 12th Text REtrieval Conference (TREC)*, 2003b.

D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schlkopf. Learning with local and global consistency. In *the Seventeenth Annual Conference on Neural Information Processing Systems (NIPS)*, 2003a.

D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schlkopf. Ranking on data manifolds. In *the Seventeenth Annual Conference on Neural Information Processing Systems (NIPS)*, 2003b.

Appendix A: TAC Ontology

1. Accidents and Natural Disasters
   (a) WHAT: what happened
   (b) WHEN: date, time, other temporal placement markers
   (c) WHERE: physical location
   (d) WHY: reasons for accident/disaster
   (e) WHO AFFECTED: casualties (death, injury), or individuals otherwise negatively affected by the accident/disaster
   (f) DAMAGES: damages caused by the accident/disaster
   (g) COUNTERMEASURES: countermeasures, rescue efforts, prevention efforts, other reactions to the accident/disaster
2. Attacks (Criminal/Terrorist)
   (a) WHAT: what happened
   (b) WHEN: date, time, other temporal placement markers
   (c) WHERE: physical location
   (d) PERPETRATORS: individuals or groups responsible for the attack
   (e) WHY: reasons for the attack
   (f) WHO AFFECTED: casualties (death, injury), or individuals otherwise negatively affected by the attack
   (g) DAMAGES: damages caused by the attack
   (h) COUNTERMEASURES: countermeasures, rescue efforts, prevention efforts, other reactions to the attack (e.g. police investigations)
3. Health and Safety
   (a) WHAT: what is the issue
   (b) WHO AFFECTED: who is affected by the health/safety issue
   (c) HOW: how they are affected
   (d) WHY: why the health/safety issue occurs
   (e) COUNTERMEASURES: countermeasures, prevention efforts
4. Endangered Resources
   (a) WHAT: description of resource
   (b) IMPORTANCE: importance of resource
   (c) THREATS: threats to the resource

   (d) COUNTERMEASURES: countermeasures, prevention efforts

5. Investigations and Trials (Criminal/Legal/Other)
   (a) WHO: who is a defendant or under investigation
   (b) WHO INV: who is investigating, prosecuting, or judging
   (c) WHY: general reasons for the investigation/trial
   (d) CHARGES: specific charges to the defendant
   (e) PLEAD: defendant's reaction to charges, including admission of guilt, denial of charges, or explanations
   (f) SENTENCE: sentence or other consequences to defendant