Statistical Machine Learning Approaches to Cross Language Text Summarization

by

Minh Le Nguyen

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Supervisor: Professor Susumu Horiguchi

School of Information Science
Japan Advanced Institute of Science and Technology

September 2004

Abstract

Automatic text summarization is the task to produce the most important content from a given text document to the user in a condensed form and in manner sensitive to the user's or application's need. Cross-language text summarization (CLTS) differs from mono-language text summarization that a given document will be automatically summarized to another language than the language of the original document. CLTS task is attracting a great deal of attention in recently years, due to the rapidly increasing amount of text information and amount of users in internet who are not familiar with a specific language. Intuitively, a CLTS system is the combination of a text summarization and a machine translation engine. If the performance of the machine translation and the text summarization system are in high quality, then a CLTS system can be established by simply combining them. Unfortunately, obtaining these systems are now desirable and challenging tasks, especially it was very difficult for some languages other than English or Japanese (e.g Vietnamese language). This is the reason why studying CLTS becomes a very difficult and challenging task.

The main goal of this research is to study an efficient method to achieve a CLTS system with a high accuracy and low cost in computational times. To obtain this, we propose statistical machine learning approaches to cross language text summarization in which Hidden Markov Models, Maximum Entropy Models, and Support Vector Machines are mainly used in the proposed CLTS. These statistical learning models are estimated based on the behavior of humans, who are professional in summarizing text document. The advantage of applying learning methods is that it can achieve a high accuracy with the minimum human effort in constructing linguistic knowledge.

The proposed system consists of three major tasks: sentence extraction, sentence reduction, and translation. First, the whole text document will be extracted to a set of important sentences which are most relevant to the gist meaning of the text document. After that, each long sentence within these important sentences will be reduced so that their meanings are unchanged. Finally, these sentences will be translated to the other language and the summary is obtained by combining the translation outputs. For sentence extraction, a corpus-based sentence extraction is investigated. In addition, we study a co-training method based on maximum entropy model which can utilize unlabeled data for improving the sentence extraction performance. Experiment results show that unlabeled data are helpful for sentence extraction using machine learning techniques and co-training method is a suitable one.

In order to reduce a long sentence to a short one, we formulate it as a process of transforming a syntactic tree of the long sentence (the large tree) to a small tree. The process is considered as a sequence of actions which transforms the large tree to a small one and the reduced sentence is obtained by simply generating from the small tree. The key problem is how to learn a sequence of actions for each syntactic tree. For solving it, we propose a deterministic sentence reduction and probabilistic sentence reduction methods which are mainly used statistical machine learning models estimated from the corpus of long sentences and their reductions. Although any statistical machine learning models

can be applied to our sentence reduction methods, in this research we investigate the use of maximum entropy models and support vector machine models to sentence reduction and experiment them on various test data. Experiment results show that the proposed methods obtain a better performance in comparison with previous work. The proposed algorithms are also closer to human manner in reducing sentences than previous works. In addition, the probabilistic sentence reductions improve the deterministic sentence reduction in term of grammticality and the measure of how the gist meaning of the long sentence retaining.

To adapt machine translation to cross language text summarization, the translation template learning (TTL) - a variant of example-based machine translation is investigated. There are two drawbacks of this translation method - one for the learning phase and another for the translation phase. In the learning phase, with the lack of linguistic knowledge the amount of template rules is large and some of them are redundant rules. To solve this problem, we propose a novel translation template learning using shallow parsing which allows incorporating linguistic information to template rules. In the translation phase, the advantage of the TTL method is that it does not need any complex parsing such as syntactic parsing or semantic parsing and it overcomes the imperfectness of the rule-based machine translation. The disadvantages of the method are that a lot of templates can be matched with an input sentence and some of them cause the translation results are less confident. In addiction, the previous methods need to evaluate all matching rules for each input sentence to obtain translation outputs, while much of them are redundant rules. The exponential calculation problem will arise when an input sentence is long and the number of template rules is large. For this problem, we present a novel method based on a HMM model that uses constraints for each input sentence with its matching rules. The proposed translation method can apply to any pairs of languages, and the application of it on English and Vietnamese shows that the proposed method improves other methods in both computational times and the translation accuracy.

We also propose a new algorithm to generate training data for both sentence extraction and sentence reduction task by automatically extracting from rough data which consist of text documents and their summaries. In final, we develop a cross language text summarization system to summary English text document to Vietnamese language. The proposed system is used a fusion of machine translation engine and a mono-language text summarization.

Key Words: Text summarization, Statistical Machine Learning, Sentence Extraction, Sentence Reduction, Example Based Machine Translation.

Acknowledgments

First, I would like to thank my advisor, Professor Susumu Horiguchi: He was a role model in many aspects and had prepared me for academic life. He taught me how to be fruitful researcher, write a good paper, and always made me believe that I could succeed. Working with professor Horiguchi, I learned the value of a vision, and persistence in materializing it, and above all, how to become a useful people.

I would like to express my sincere thanks to Professor Akira Shimazu for his guidance not only in my sub theme topic but also for many problems in natural language processing. His understanding and inspiration encourages me to finish this thesis.

I would like to express my sincere thanks to Professor Ho Tu Bao of JAIST for his supports, and encouragement for my research. His understanding and inspiration encourages me to overcome many difficulties to finish this thesis.

I would like to thank to my committee members: Professor Yasushi Hibino and Professor Teruo Matsuzawa for reading my thesis and providing valuable feedbacks.

I am grateful to my former supervisors, Dr. Pham Hong Nguyen, Dr. Ha Quang Thuy, Professor Dinh Manh Tuong, Professor Ho Si Dam, and many members of Faculty of Technology, Vietnam National University, Hanoi for their advice and encouragement throughout my studies. I would like to thank all former machine translation group, Cuong, Vinh, Thai, and LACVIET software company help me in preparing data and testing for this work. I would like to thank Dr. Masaru Fukushi (now Associate at Tohoku University) for his contributions and many helps to me during my research work.

I sincerely thank all my friends and colleagues who always supported me in times of need. I greatly appreciate to my lab members for their contributions in making a wonderful and supportive academic environment. Thank also to many Vietnamese friends at JAIST for the good time over the part three years.

I am deeply indebted to the Japanese Ministry of Education, Culture, Sports, Science and Technology for granting me a scholarship. Thanks also to the JAIST Foundation for providing me with their travel grants which supported me to attend and present my work at international conferences.

I would like to thank all members of my family for their love and support, especially my parents for everything they taught me and for all the sacrifices they made in my upbringing. I would like to thank my parents in law for their loves during my studying this research. I would like to acknowledge my brother, Nguyen Anh Tuan for his attempts everyday for recovering his health. This encourages me very much during studying the thesis.

Finally, I would like to acknowledge my wife, Kieu Que Anh and my son, Nguyen Hoang Nhat. Without their encouragements I would never have began, and much less completed this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Automatic text summarization is the task to produce the most important content from a given text document to the user in a condensed form and in manner sensitive to the user's or application's need. Many text summarization applications are now widely used, such as Search Engine Hits (summarizing the information in hit list retrieved by search engine), Hand-Held Devices (creating a screen size version of a book), and Headline generation on television [1], [2].

With the rapidly increasing number of text information and amount of users in Internet who are not familiar with a specific language, the multilingual information system becomes increasingly desirable. Automating the production of multilingual summarization is therefore widely recognized as a highly challenging task [1]. Among the multilingual summarizations, cross-language text summarization (CLTS)- the task of producing a summary in other language than the language of the given text document is very attractive.

Intuitively, a CLTS system consists of two main components, a text summarization and a machine translation engine. If the performance of the machine translation and the text summarization system are in high quality, then a CLTS system can be established by simply combining them. Unfortunately, obtaining these systems are now desirable and challenging tasks, especially for some languages other than English or Japanese (e.g Vietnamese language). This is the reason why studying CLTS becomes a very difficult and challenging task. There have been also some attempts to build a CLTS system [3], [4], [5], [6], [7], [8], [9]. The major drawbacks of the previous works are likely to treat CLTS as a translation phase and summarization phase separately. A CLTS system is established by performing a machine translation before (MT-before) or after summarizing (MT-late). Some studies claim that using machine translation before summarizing obtained a higher performance than using it after summarizing a given text document, but the computational times of this strategy is slower than using MT-Late [8], [9]. This was due to the fact that machine translation are time-consuming tasks and performing a translation for the whole text document must be slower than a summary document. In addition, almost every MT engines are designed to deal with an grammatical input sentence, not for phrases or clauses, while the output of a text summarization does not often satisfy the grammatical criterion.

The straightforward way to obtain a CLTS system with a high accuracy and low cost in computational times is that we should not only focus on enhancing the performance

of the mono-language summarization but also adapt a machine translation engine to text summarization.

There have been several kinds of mono-language text summarizations. Sentence extractions aim at extracting a set of important sentences within the given text document, sentence reduction is the task to reduce long sentences into the short sentences so that the gist meaning of the short one is the same as that of the original [1]. Sentence reduction techniques can be also used to improve the performance of sentence extractions. In additional contribution to mono-language text summarizations, Jing proposed the cut and paste text summarization system based on the behavior of professional summarizer[10]. Figure 1.1 shows a kind of mono-language text summarization, the cut and paste text summarization. This summarization system mainly consists of three components: sentence extraction, sentence reduction, and sentence combination. First, the whole text document will be extracted to a set of important sentences which are most relevant to the gist meaning of the text document. After that, each long sentence within the set of important sentences will be reduced to a condensed form by a sentence reduction method. Finally, these short sentences are combined to a summary in the same language as that of the original.

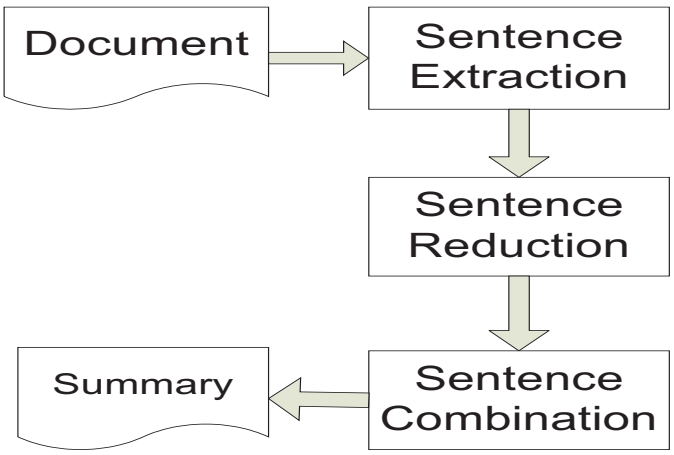

Figure 1.1: The mono-language text summarization: Cut and Paste Text Summarization

To extend a mono-language summarization system to a cross-language text summarization, we incorporate it with a machine translation engine by using a fusion strategy of machine translation and mono-language text summarization. As shown in Figure 1.2, the proposed system consists of three major tasks: sentence extraction, sentence reduction, and translation. After obtaining a set of important sentences and reduced them to short sentences. These outputs will be translated to the other language and the summary documents are obtained by simply combining these translation outputs.

Our main goal is to obtain a scalable cross language summarization with a high performance and the minimum human effort in constructing linguistic knowledge. For this reason, we mainly apply statistical machine learning methods to all components in our cross-language text summarization system.

As shown in Figure 1.2, there are three main tasks in our CLTS system which includes:

- Sentence Extraction: For this task we propose a statistical learning algorithm using

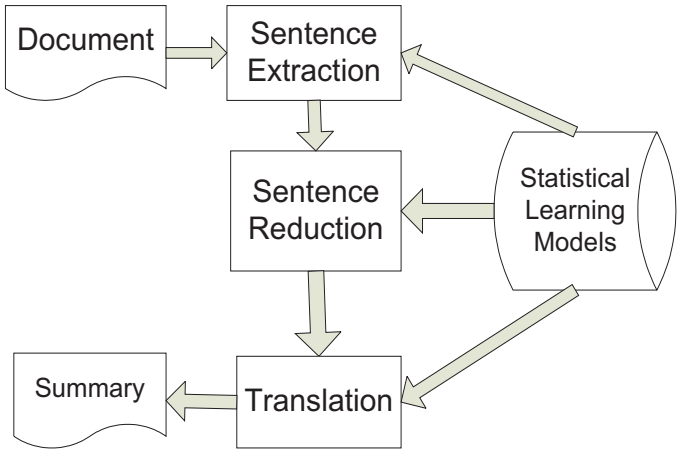

Figure 1.2: A cross-language text summarization system

unlabeled data to improve the sentence extraction accuracy.

- Sentence Reduction: We also propose new probabilistic sentence reductions using statistical machine learning modes estimated from the corpus of long sentences and their reductions. The proposed algorithms show either a better reduction performance or being closer to human manner in reducing sentences in comparison with previous work.

- Translation: We draw a new perspective for applying statistical machine learning to example based translation domain. Our translation system improves original works in both the computational times and the translation accuracy. We also propose a new algorithm to generate training data for sentence extraction and sentence reduction automatically from rough data which consist of text document and their summaries.

In final, an application of this research is to build a text summarization system to summary English text document to Vietnamese language. Our cross language text summarization system is mainly used statistical machine learning models which are estimated from training data available.

1.2 Focus of Research

To achieve a useful cross language text summarization system, four efficient tasks; sentence extraction, sentence reduction, translation, and generating training data are necessary. Four tasks in our cross-language text summarization system are defined as follows.

- **Sentence Extraction**
  Osbone [13] proposed a method of applying maximum entropy model (MEM) to sentence extraction. However, this study has discussed only the advantage of using MEM in comparison with another method (naive Bayes). There was no discussion

about selecting features on sentence extraction using MEM. In this paper, we propose a sentence extraction based on maximum entropy principle, in which these features are selected carefully. In addition, we propose a co-training version for MEM which can utilize unlabeled training data to enhance the accuracy of sentence extraction.

- **Sentence Reduction**
  Several methods have been proposed for sentence reduction in various applications [14], [15],[16] [17],[18],[19],[20]. Among these methods, the corpus based sentence reductions have attracted much attention because they provide the good way to scale up the full problem of sentence reduction using available data.

  Knight and Marcu [14] proposed a sentence reduction based on corpus using machine learning techniques. They discussed a noisy-channel based approach and a decision tree based approach to sentence reduction which involve the constraint that the word order of a given sentence and its reduced sentence are the same. Nguyen and Horiguchi [18], [19] presented a new sentence reduction technique based on a decision tree model without that constraint. They also indicated that semantic information is useful for sentence reduction tasks.

  The major drawback of previous work on sentence reduction is that those methods are likely to output "local optimal" results, which may have lower accuracy. This problem is caused by the inherent sentence reduction model; that is, only a single reduced sentence can be obtained. As pointed out by Lin [20], the best sentence reduction output for a single sentence is not approximately best for text summarization. This means that "local optimal" refers to the best reduced output for a single sentence and the best reduced output for the whole text is "global optimal". Therefore, it is very valuable if the sentence reduction task can generate multiple reduced outputs and select the best one using the whole text document. However, such a sentence reduction method has not yet been proposed so far.

  The aim of this study is to illustrate the potential of statistical machine learning in enhancing the accuracy of sentence reduction in comparison with previous work. For this purpose, we developed two statistical learning models including MEMs and SVMs to sentence reduction.

- **Translation**
  Cicekli and Günivenir [23], [24] proposed a novel machine translation system based on template translation learning. However, these are two drawbacks in the learning phase and translation phase of the original system. In the learning phase, with the lack of linguistic knowledge, the amount of template rules obtained from translation template learning is large and some of them cause the translation wrong. In addition, unreliable rules reduce the performance of translation in both accuracy and computational times. Incorporating linguistic knowledge into template rules is an expected approach. To solve this problem, we propose a novel translation template learning using shallow parsing to incorporate linguistic information to template rules. In the translation phase, previous works on template translation learning are required evaluating all matching rules for each input sentence to obtain the output results, while much of them are redundant rules. The exponential calculation problem will arise when an input sentence is long and the number of template rules is large. To

solve these problems, we propose a novel method based on a Hidden Markov Model (HMM).

- **Generating training data**
  Jing [25] proposed generating training data for text summarization as a decomposing human-written summary sentences by using a Hidden Markov Model based on a set of heuristic rules. However, when humans produce a summary document, they may use some of their own words or phrases that are coincidentally similar to a word or a phrase in the original document. In addition, the repetition words occurred in summary sentences may affect the decomposition accuracy. To cope with these cases, we use the semantic measure to discover all words in the documents that have the same meaning as a word within the summary sentences and to reduce the number of words in the summary, which have no occurrence in the original. The purpose of the position-checking is to consider whether or not the case in which distinct words within a summary sentence come from the same position in the original document can enhance the decomposition accuracy.

The portability is one of the advantages of our CLTS system because the CLTS is built by automatically estimating from the training data. It is clearly that we can implement the proposed CLTS system to any domain. The proposed system is scalable as well as the MT-Late approach and it also has the advantages in comparison with previous work in reducing sentence and extracting important sentences. In addition, the adaption of machine translation engine to text summarization could deal with the problem of translating ungrammatical sentences.

1.3 Outline of The Thesis

This dissertation is organized as follows. Chapter 2 presents previous work on cross-language text summarization. In addition, we propose a statistical machine learning approach to cross language text summarization. The proposed system mainly uses statistical machine learning for four specification parts and combining them to a cross language text summarization system. The propose cross language text summarization includes the tasks as follows.

- Generating training data for cross language text summarization.
- Sentence extraction.
- Sentence reduction.
- Machine translation in CLTS.

To provide a broader view of statistical machine learning models (SML), we introduce three models which consist of Hidden Markov Model, Maximum Entropy Models, and Support Vector Machines. Chapter 3 will overview three SML models which are mainly applied to all tasks throughout this thesis. The following chapters will introduce statistical machine learning methods for generating training data, sentence extraction, sentence reduction, and machine translation in CLTS.

To build a cross language text summarization based statical machine learning, generating training data is very essential. The technique for generating training task is described in Chapter 4.

In Chapter 5, we presents a statistical machine learning model for sentence extraction in which a maximum entropy model and its version in co-training using unlabeled data are discussed. The proposed sentence extraction method utilizes the unlabeled training data to boost the performance of sentence extraction.

Chapter 6 introduces sentence reduction as the process of transforming actions from the long sentence to the reduced sentence. We present new probabilistic sentence reduction methods using statistical machine learning in which support vector machine and maximum entropy models are used to learn syntactic tree transformation actions for sentence reduction.

Chapter 7 takes into account the uses of statistical learning to example based machine translation in order to adapt it to CLTS system. We also demonstrate a novel translation template learning technique using Hidden Markov Model which can both work well in sentence translation and sentence reduction.

Chapter 8 proposes a cross-language text summarization (CLTS) system for English and Vietnamese language. Finally, chapter 9 summarizes our contributions and draws future works.

Chapter 2

Cross Language Text Summarization

2.1 Mono-Language Text Summarization

2.1.1 Summarization machine

The main goal of a summary is to present the main ideas in a document in less space, it is clear that if all sentences in a text document were equal importance, producing a summary would not be very effective, as any reduction in the size of a document would carry a proportional decrease in its informativeness. Fortunately, information content in a document appears in burst, and one can therefore distinguish between more and less informative segments. Identifying the informative segments at the expense of the rest is the main challenge in summarization.

Two excellent books [1][2] have provided an general view for text summarization, in which many types of summary have been identified. Figure 2.1 illustrates a summariza-

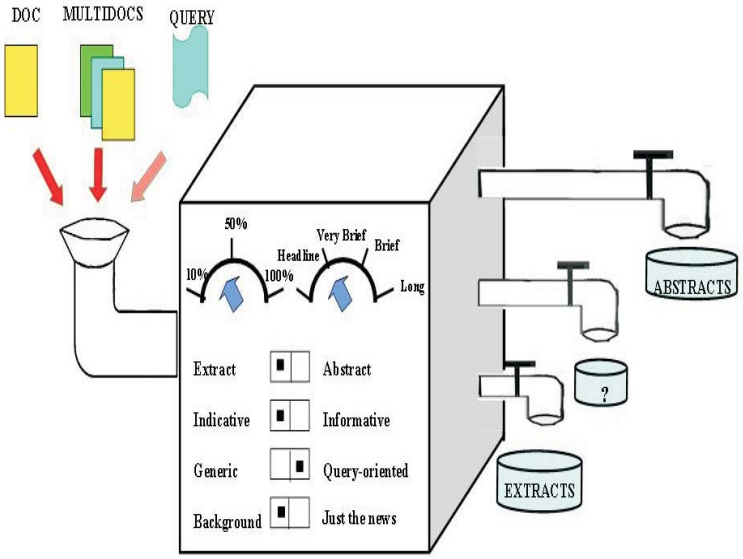

Figure 2.1: A summarization machine

tion machine which describes an overview of mono-language summarization including the

input, output, and the type of a summarization. *Indicative summaries* provide an idea of what the text is about without conveying specific context, and *Informative* one provide some shortened version of content. *Query-oriented* summaries concentrated on the reader's desired topic(s) of interest, whereas *Generic summary* reflect the author's point of view. *Extracts* are summaries created by using portions (words, sentences, etc.) of the input text verbatim, while *abstracts* are created by regenerating the extracted content. *Extraction* is the process of identifying important material in the text, *abstraction* the process of reformulating it in novel terms, fusion the process of combining extracted portions, and compression the process of squeezing out unimportant material. The need to maintain some degree of grammaticality and coherence plays a role in all four processes. The input might single document, multiple document, or users query. A summarization rate also is an essential parameter in a machine summarization. Figure 2.1 also shows the summarization size can be 10%, 50% or a headline, very brief, brief, or long summary. The output of a machine summarization can be extractions or abstraction, or somewhat that is useful for reader's point of view.

Figure 2.2 shows a real application of mono-language text summarization. This example was obtained online from the website (http://www.vnagency.com.vn/).

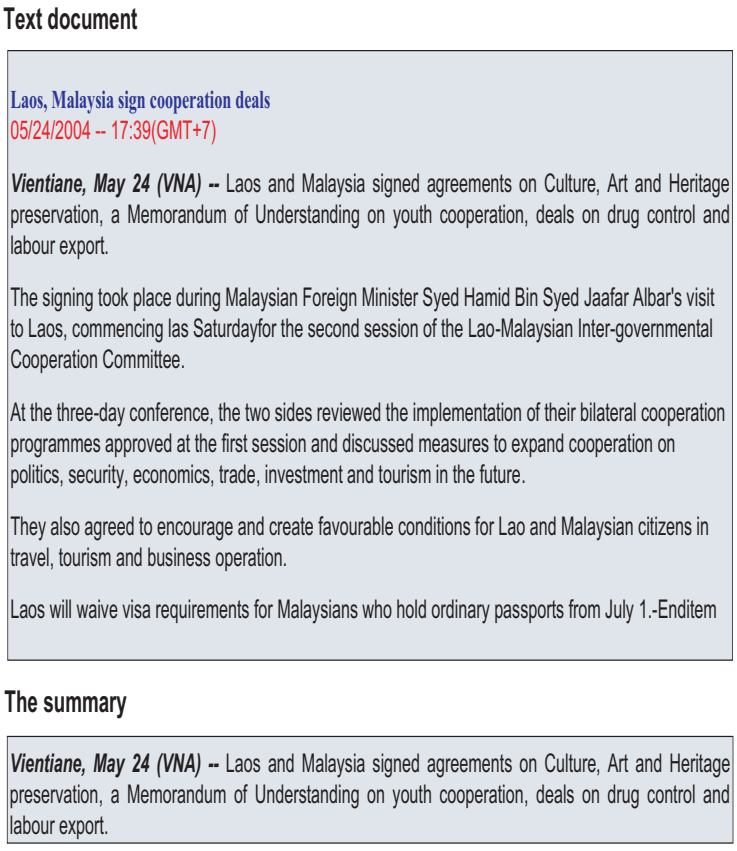

Figure 2.2: An example of a mono language text summarization

2.1.2 Extractions

Among extraction tasks, sentence extractions are used to determine the most important sentences and clauses in a document or a collection of documents. Previous approaches to text summarization can be divided to the main categories following: Heuristic approach, Knowledge base approach, shallow based approach, statistical based approach, and hybrid approaches.

Heuristic approach

The traditional approach is mainly based on the heuristic methods. We can list some of heuristic methods as bellow.

- Position-Based: The simple method is based on the assumption that sentences that occur at the beginning of document are more likely to be important than sentences that occur in middle or at the end. The simplest way is to build a summary that always selects the first sentence in a text; or the first k sentences in a text, when a summarization rate is required. Although the performance of this method varies significantly with text genre and summarization rate, the position-based method is usually capable of identifying around 33% of the important sentences in a text[11].

- Title-based method: Edmundson [11] was the first to show that the words in titles and headings are more likely to be used in important sentence in a text document than in non-important sentence.

- The cue-phrase method: Cue-phrase-based system capitalize on the observation that important sentences contains "bonus phrase" such significantly, in this paper we show, and in conclusion, while non-important sentences contain "stigma phrase" such as hardly and impossible. The cue-phrase method yielded the best results when used to identify important sentence in scientific articles.

- The word-frequency method: The important sentences in a text are those that contain words that occurs "somewhat" frequently.

- Word-based method: The most straightforward approaches apply Information Retrieval technique[12],[26],[27] to compute the similarity between the paragraphs in a texts. The paragraph that has highest collective similarity to the other paragraphs are assumed to be central/important to the document belong.

Knowledge based approach

The knowledge based approach concerned to summarize text in a specific domain. This approach relies on using rich knowledge about the domain to decide which segments in a given text document should be included in a summary. There have been some text summarization systems based knowledge approach. Paice used stylistic clues to identify important concepts in highly structured technical papers [28]. McKeown and Radeve used the results of information extraction to construct fluent summaries for a cluster of document using natural language generation technique [29]. The knowledge based approaches are suited for a specification domain, but it is hard to deploy these methods to a new domain. This was because it requires understanding text deeply, the performance of such methods also are not scalable as well.

Shallow based approach

An alternative to knowledge based approach is using shallow linguistic knowledge. This approach does not need to understand text as deep as knowledge based approach. Rather, it utilizes the lexical cohesion and coherent [30], [31], [32], [33] or discourse structure of the given text[34] to summary the text. Some researchers map texts into description logics and perform condensation operations on formal representations [35].

Statistical based approach

The statistical approach has been affirmed successfully in many NLP applications, such as machine translation, information retrieval, and information extraction. Many statistical models are applied to text summarization fields. Earliest works [12]concerned to use the Vector Space Model in Information Retrieval to measure the similarities between paragraphs, then extracts important ones. Other works [36] focus on formulating summarization as a statistical classification problem, then deploys some classification algorithms such as naive Bayes classification[37], Maximum Entropy Models[13], Support Vector Machine [38], and Weighted Probability Distribution Voting (WPDV)[39] for summarization. Some variants of statistical model including Hidden Markov Model, noisy channel are also applied to sentence extraction[40], sentence reduction [14], and web summarization [41]. The advantages of statistical based approach are that the computation times is fast and the human in constructing linguistic knowledge is not required. It is also easy to apply for summarization in other domains. The disadvantage is that it require a lot of training data and the noisy in the data can decrease the performance of text summarization.

Hybrid approach

In practice, a summarization system might combine and use more than one approach. These methods are refereed as a hybrid approach, for example a shallow approach might combine with statistical techniques.

2.1.3 Abstraction

Recently, researches have focused on generation problem for text summarization. The generation technique is treated as the key to obtain a summary with grammatical and coherent. It overcomes several obstacle problems in extraction as listed following: the extraneous phrases (Extracted sentences can be very long, containing material that are not need to be included in a summary), dangling pronouns and noun phrases, misleading information, and so on. The two main generation techniques for text summarization are discussed bellow.

Sentence Reduction

To adapt generation techniques to text summarization, several methods have been investigated. Jing have been proposed cut and paste technique for professional summarization [10], in which some professional operators including reduction, combination, and paraphrase are used as key techniques for enhancing the radiability of a summary. Jing's method prevents the removal of some important phrases that are relative to the surrounding context and produces a grammatical sentence. However, while this method exploits a

simple model for sentence reduction by using statistics computed from a corpus, a better model can be obtained by using a learning approach.

In an extreme case of cut and paste, Witbrock and Mittal [21] extracts a set of words from the input sentence and then order the words into sentence using N-gram.

Knight and Marcu also applied a generation method based statistical machine translation technique for sentence compression [14], in which the reduction process is formulated as a sequence of rewriting process to transform a long sentence to a shorter one. An example to illustrate the sentence reduction is shown bellow.

- Long sentence: *The files are stored in a temporary directory on the VAX disks , where they are converted to VMS Backup format .*
- Short sentence: *Files are stored in a temporary directory on the VAX disks .*

Sentence paraphrasers

A number of researches have studied the type of paraphrase operation employed during summarization [42][10] and applied them in implemented systems. Generation technique thus is now considering as a key to enhance the performance of text summarization. For instance, sentence reduction and information fusion [43] enhance single text summarization and multiple text summarization, respectively. However, improving their accuracy are now challenging task. The example bellow shows a sentence can be write an other sentence using sentence paraphrase.

- Original sentence: *The article was warmly discussed, which produced it a high reputation.*
- Paraphrase sentence: *The paper was hotly debated, causing a fine old uproar.*

Headline generators

The vast majority of the work on headline generation has been carried out in a statistical-based noisy-channel models [44], [45], [46]. Using a large of collections of (text, headline) tuples, which can be easily collected from the web, we can estimate probability $P(w_d|w_h)$ that reflect the likelihood of a word w_d occurring in a document when another word w_h occurs in a headline. By treating documents and headlines as bags of words, we can easily estimate the probability $P(D|H)$ of a document given a headline. Once these parameters $P(H)$ and $P(D|H)$ are estimated, we can construct document headlines by searching for sequence of words H that maximize the product $P(D|H) \times P(H)$.

2.1.4 Multi-Document Summarization

Summarizing a single text is difficult enough. But summarizing a collection of thematically related documents poses several additional challenges. Various methods have been proposed to identify cross-document overlaps. SUMMONS [48], a system that covers most aspects of multi-document summarization, takes an information extraction approach. All documents are assumed that they are parsed into templates, SUMMONS clusters the templates according to their contents, and then applies rules to extract items for summarizing. In contrast, Barzilay et al [43] parse each sentence into a syntactic dependency structure

(a simple parse tree) using a robust parser and then match tress across documents, using paraphrase rules that alter the trees as needed.

To determine what additional material should be included, Carbonell et al[49] first identify the units most relevant to the user's query and then estimate the marginal relevance of all remaining units using a measure called Maximum Marginal Relevance (MMR).

Multi-document summarization have been much attention in recently, it also is a major task on the annual competition conference on document understand (DUC-2001, DUC-2002, DUC-2003, and DUC-2004).

2.1.5 Evaluation Summaries

Evaluating the quality of a summary has proven to be a difficult problem because there is no obvious "ideal" summary. Event for news article, human summarizer tend to agree only approximately 60% of the time, measuring sentence content overlap.

Two broad classes of metrics have been developed: from metrics and content metrics. From metrics relies on grammaticality, overall text coherence, and organization and are usually measured on a point scale [47]. Content is more difficult to measure, we can use those sentences or fragments generated by humans to compare with the system outputs, and as in information retrieval, the percentage of important information present in the system's summary (precision) and the percentage of important information omitted from the summary (recall) are recoded. In the Document Understanding Conference (DUC)-01 and (DUC)-02 summarization competitions, NIST used the Summary Evaluation Environment (SEE) interface to record values for precision and recall. DUC has shown that humans are better summary producers than machines and that, for the news article genre, certain algorithms do in fact better than the simple baseline of picking the lead material.

Automatic summary evaluation is a gleam in everyone's eye. Clearly, when an ideal extract has been created by humans, extractive summaries are easy to evaluate. Evaluation for abstracts are more complex, simply using a variant of the Bilingual Evaluation Understudy (BLEU)[51] scoring method (based on a linear combination of matching n-grams between the system output and the ideal summary) developed for machine translation is promising but not sufficient[52]. Recently, Lin and Hovy developed a new evaluation scoring, the ROUGE score for text summarization [53]. This scoring system are based on the BLEU method and it was used to measure the performance of the participated text summarization systems in DUC-04[54].

2.2 Cross Language Text Summarization

Cross language summarization is a task of summarizing a given text document in one language to a summary in other language. Intuitively, a cross language text summarization consists of two main components, a summarization and a translation component. Figure 2.2 shows an example of a cross language text summarization in real application. The original document and the the summary is the English news and its summary in Vietnamese language obtained from the website (http://www.vnagency.com.vn/).

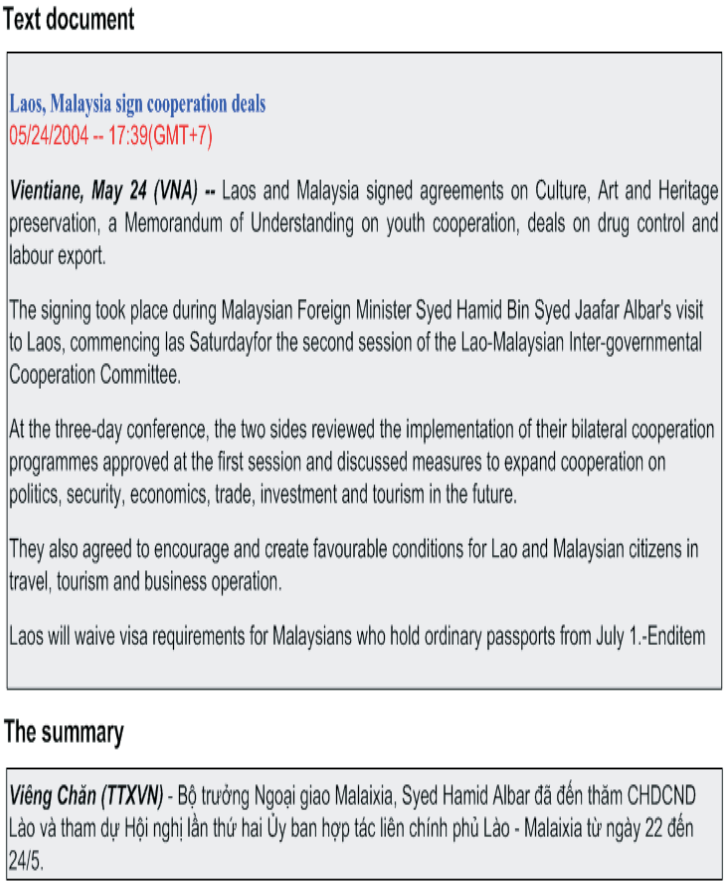

Figure 2.3: An example of cross language text summarization

2.2.1 Previous Work

Previous work on cross language summarization have been mainly focused on building a combination system between a summarization system and a translation system. Hovy and Lin [3] proposed a SUMMARIST system which extracts sentences from documents in a variety of languages, and translates the resulting summary. This system has been applied to Information Retrieval in the MuST system [4] which uses the query translation to allow a user to search for documents in variety of languages, summarize the document using SUMMARIST, and translate the summary. Ogden et al [5] proposed the Keizei which uses query to search Japanese and Korean documents in English, and displays query-specific summaries focusing on passages containing query terms. Chen and Lin [6] described a system that combines multiple monolingual news clustering components, a multilingual news clustering component, and a news summarization component. Their system clusters news in each languages into topic, then the multilingual clustering component relates the clusters that are similar to across language. A summary is generated by linking sentences that are similar to the two languages. The system has been implemented for Chinese and English. Recently, National Institute of Standards and Technology (NIST) has been issued several tasks for text summarization, such as single summarization, multi-document summarization, and multi-lingual summarization. Among them, the task of multi-lingual

summarization has been issued in this year [54] in which original document in Arabic language were translated to English document by the IBM machine translator. After that, English document are then summarized by participated summarization systems and the outputs will be evaluated by NIST. An alternative approach has been turned out to incorporate machine translation to internal steps in a mono-lingual summarization system are described in [7].

The same as that of the evaluation problem in mono-language summarization, it was very difficult for cross-language summarization. For this reason, using human evaluation the system's output are considered as the main evaluation method for cross-language text summarization [8][9]. Beside, the BLEU and ROUGE scoring method can be also used to evaluate the performance of a cross-language text summarization, those experiments described in [8][9] showed that the evaluation results using BLEU and ROUGE agree with human evaluation.

In summary, there are two main components in a multilingual summarization system which are a mono-lingual summarization and a machine translation phase. The problem here is how to incorporate a machine translation into a mono-lingual summarization. Intuitively, there are three approaches for cross language text summarization as follows.

- MT-late: A given text document is summarized, and the output is then translated into the desired language using a machine translation system (MT).
- MT-before: This approach summarizes the translated document after performing a machine translation process on the original text document.
- MT-alternative: The third approach concerned to integrating machine translation in internal step of a mono-lingual summarization system.

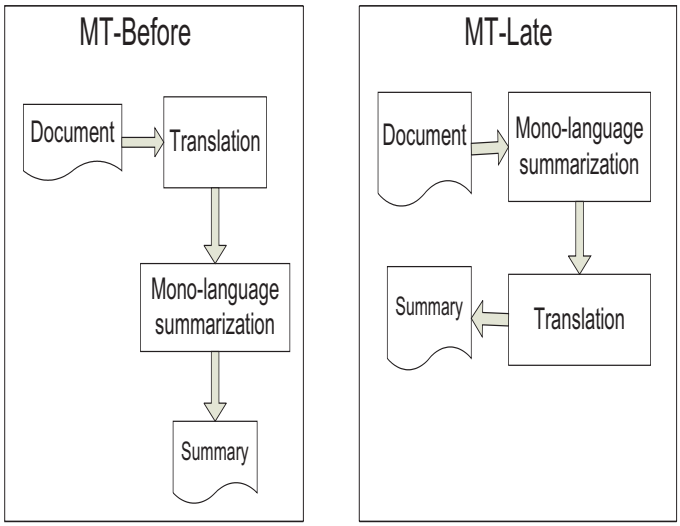

Figure 2.4: Machine translation in CLTS

Figure 2.4 shows the comparison of MT-late and MT-before in CLTS summarization. Some studies claimed that using MT-before improved MT-late in term of performance accuracy but it required much more computational times [8], [9]. To achieve a scalable cross

language text summarization with a high accuracy, we thus consider the MT-alternative approach and present a new approach to cross language text summarization in the next subsection.

2.2.2 Statistical Machine Learning for Cross-Language Text Summarization

Despite encouraging progress in the MT quality over the past years, MT outputs is still, for the most part, ungrammatical and quite hard to read. Clearly, in the first approach, performing machine translation on the whole documents costs more computational times than doing it on a summary document. However, the translation engine was designed to translate whole sentences, not phrases. It does not perform as well when the input is a list of separate phrases. In addition, summary outputs of a summarization system are still in low accuracy. These reasons thus lead us to the tradeoffs between computational times and accuracy in using MT-before and MT-late for cross language summarization. Some experiments on headline generation between English and Hindi [8][9] claimed that MT-before are better than MT-late in term of accuracy.

Our main goal is to obtain a scalable cross language text summarization with a high performance. Therefore, a MT-alternative approach using an adaption of MT to monolingula text summarization should be applicable to solve this problem. We also try to improve both the performance of text summarization and machine translation.

Statistical machine learning has been widely used in many Natural Language Processing (NLP) tasks such as pos tagging, text chunking, syntactic parsing, machine translation, and information retrieval. This is the first time SML models are applied to the field of cross language text summarization.

Our approach to CLTS including two main points:

- Using statistical machine learning to improve mono-lingual text summarization.
- Adaption of machine translation for mono-lingual summarization.

For the first point, we focus on improving sentence extraction and sentence reduction performance for mono-lingual text summarization.

For sentence extraction, a corpus-based sentence extraction is investigated. In addition, we study a co-training method based on maximum entropy model which can utilize unlabeled data to improve the sentence extraction performance. Experiment results show that the unlabeled data were helpful for sentence extraction task using machine learning technique, and co-training methods seem to be suitable for this task. Beside, to reduce a long sentence to a short sentence, we formulate it as a process of transforming a syntactic tree of the long sentence (the large tree) to a small tree. The process is considered as a sequence of actions which transforms the large tree to a small tree and the reduced sentence is obtained by simply generating from the small tree. The key technique is how to learn a sequence of actions for each syntactic tree. To solve it, a deterministic sentence reduction and a probabilistic sentence reduction method are proposed. The proposed methods are mainly used statistical machine learning models estimated from the corpus of long sentences and their reductions.

To adapt machine translation to cross language text summarization, the translation template learning - a variant of example-based machine translation is investigated. There

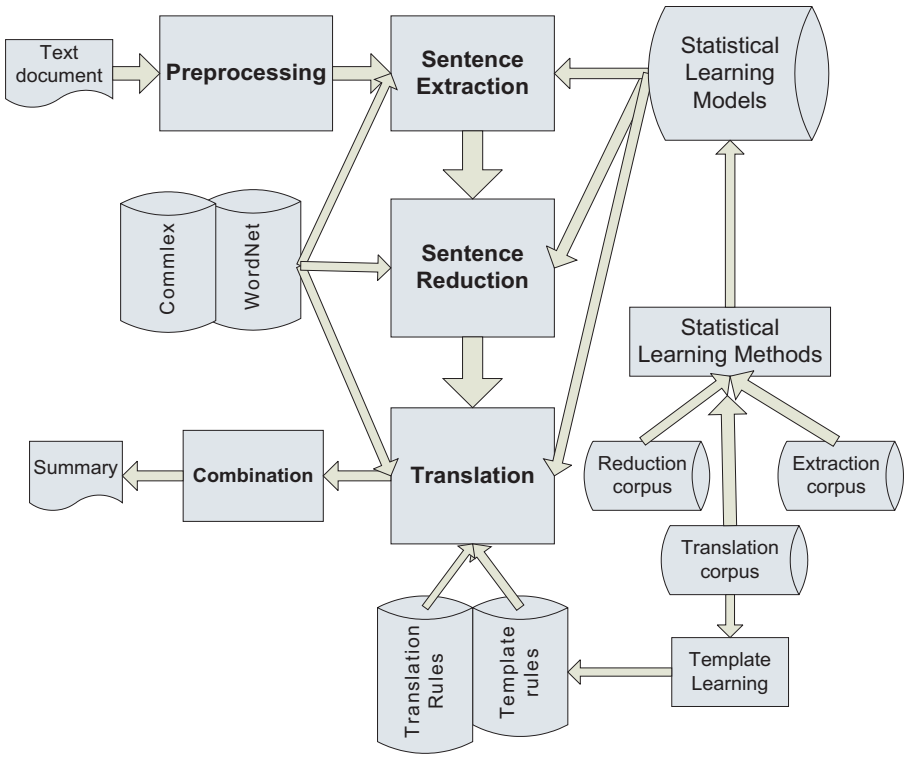

Figure 2.5: A cross language text summarization system

are two drawbacks of the translation template learning methods in the learning phase and the translation phase. In the learning phase, we incorporate shallow information to overcome the problem of generating a large amount of unreliable template rules due to the lack of linguistic knowledge. In the translation phase, the advantage of this method is that it does not need any complex parsing such as syntactic parsing or semantic parsing and overcome the imperfectness of the rule-based machine translation. The disadvantages of the method are that a lot of templates can be matched with an input sentence and some of them cause the translation results are less confident. In addiction, the previous methods need to evaluate all matching rules for each input sentence to obtain the output results, while much of them are redundant rules. The exponential calculation problem will arise when an input sentence is long and the number of template rules is large. Following that point, we present a novel method based on statistical machine learning models that use constraints for set of matching rules with each input sentence.

We then divide a CLTS system into three components. The first one concerns to extract important sentences for a given input text document. The second one aims at reducing a long sentence into a condensing version, and the last one is the adaptation of a machine translation engine to a mono-language summarization. We also incorporate knowledge based with SML models which were estimated from the training data available. The knowledge based using in the CLTS including WordNet, Comlex, and Translation Rules database.

Figure 2.5 shows a CLTS system using statistical learning models which are applied to all the components in the CLTS including sentence extraction, sentence reduction, and

translation. Figure 2.5 also shows that the statistical learning models are estimated by learning from the corpus of extraction, reduction, and translation. For adaption machine translation in the proposed CLTS system, the template learning algorithm is used to generate template rules from the translation corpus. These template rules along with translation rules are used to translate a short sentence.

As shown in Figure 2.5, the process of summarizing an input text document is that: After preprocessing, the text document is extracted to obtain a set of important sentences. These important sentences are then reduced to condensation forms and their outputs are translated to another language. In final, the translation outputs are simply concatenated to obtain a summary.

Chapter 3

Statistical Machine Learning

Statistical machine learning (SML) is widely applied to various fields on computer science such as computer vision, imagine processing, and natural language processing (NLP). It was claimed having the potential to amplify every aspect of working scientist's progress to understanding [55]. SML have been also shown the successful in dealing with the hardest problem - the ambiguous problem in NLP. Many applications including machine translation, information retrieval, and text mining are applied SML successfully. In this chapter, we will summary three common SML models including Hidden Markov Models(HMM), Maximum Entropy Models (MEM), and Support Vector Machine (SVM). The first section introduces the Hidden Markov Model, the second one shows the maximum entropy model, and the final section summaries the SVM model.

3.1 Hidden Markov Model

3.1.1 HMM

Hidden Markov Models (HMMs) are a generalization of Markov Models[56]: whereas in conventional Markov Models the state of the machine at time i and the observed output at time i are one and the same, in Hidden Markov Models the state and output are decoupled. More specifically, in an HMM the automaton generates a symbol probabilistically at each state; only the symbol, and not the identity of the underlying state, is visible. Figure 3.1 shows an example of HMM for text classification problem. To illustrate, suppose that a person is given a text document and is asked to classify it into either business categorical, sports, scientists, the weather, or politics. At first, the person read a first paragraph and see some words including shares, bank, investor; in all likelihood the text seems to be a business category. In the next paragraph they showed some words such as front, showers and rain, which is likely a weather category. Figure 3.1 shows an HMM corresponding to this process – the state corresponds the words in text document from that category. According to the values in the figure, the word taxes accounts for 2.2 percent of the words in news category, and 1.9 percent of the words in business category. Seeing the word taxes in the text document does not by itself determine the most appropriate labelling for the text.

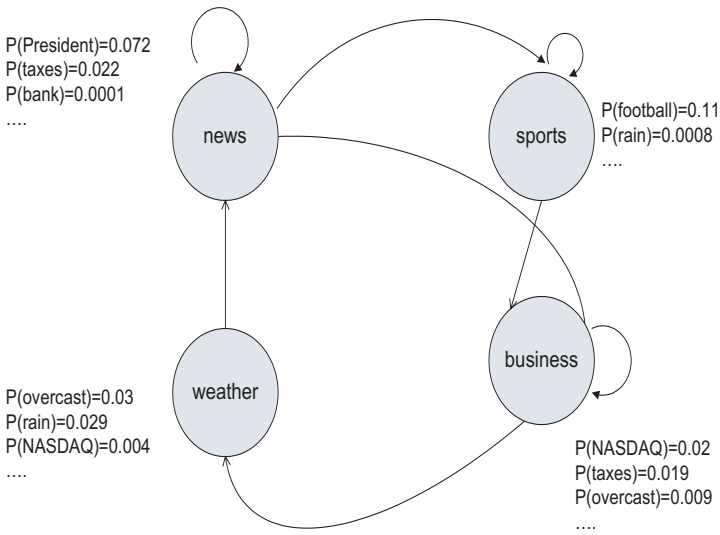

Figure 3.1: An example of HMM

3.1.2 HMM definition

An HMM is specified by a five-tuples (O, S, A, B, Π), where S and O are the set of states and the output alphabet, and Π, A, B are the probabilities for the initial state, state transition, and symbol emission, respectively.

3.1.3 Three Problems

There are three fundamental questions for a HMM model:

- Given a model $\mu = (A, B, \Pi)$, how do we efficiently compute how likely a certain observation is, that is $P(O|\mu)$?
- Given the observation sequence O and a model μ, how do we choose a state sequence $(X_1, X_2, ..., X_{T+1})$ that best explains the observations?
- Given an observation sequence O, and a space of possible models found by varying the model parameters $\mu = (A, B, \Pi)$, how do we find the model that best explains the observed data?

Probability calculation

Given the observation sequence $O = (o_1, o_2, ..., o_T)$ and a model $\mu = (A, B, \Pi)$, we wish to know how to efficiently compute $P(O|\mu)$- the probability of the observation given the model. This process is often referred to as decoding.

For any sequence of states $X = (X_1, ..., X_{T+1})$, the probability $P(O|X, \mu)$ can be calculated as follow:

$$
\begin{aligned}
P(O|X,\mu) &= \prod_{t=1}^{T} P(o_t|X_t, X_{t+1},\mu) \\
&= b_{X_1 X_2 o_1} b_{X_2 X_3 o_2} ... b_{X_T X_{T+1} o_T}
\end{aligned}
\tag{3.1}
$$

and,

$$
P(X|\mu) = \pi_{X_1} a_{X_1 X_2} a_{X_2 X_3} ... a_{X_T X_{T+1}} \tag{3.2}
$$

Since,

$$
P(O, X|\mu) = P(O|X,\mu) P(X|\mu) \tag{3.3}
$$

Therefore,

$$
\begin{aligned}
P(O, X|\mu) &= \sum_{X} P(O|X,\mu) P(X|\mu) \\
&= \sum_{X_1 ... X_{T+1}} \pi_{X_1} \prod_{t=1}^{T} a_{X_T X_{T+1}} b_{X_T X_{T+1} o_T}
\end{aligned}
\tag{3.4}
$$

To avoid the combinatorial explosion the $P(O, X|\mu)$ the dynamic algorithm can be used by using the forward procedure and the backward procedure.

The forward procedure

Let $\alpha_i(t) = P(o_1 o_2 ... o_{t-1}, X_t = i|\mu)$ is a forward variable at t step. The forward variable $\alpha_i(t)$ is stored at (s_i, t) in the trellis and expresses the total probability of ending up in state s_i at time t (given that the observations $o_1 ... o_{t-1}$ were observed). This variable is summed by probabilities for all incoming arcs at a trellis node.

- Initialization
  $\alpha_i(1) = \pi_i, 1 \leq i \leq N$
- Induction

$$
\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_{ij o_t}, \quad 1 \leq t \leq T, 1 \leq j \leq N
$$

- Total

$$
P(O|\mu) = \sum_{i=1}^{N} \alpha_i(T+1)
$$

This algorithm requires only $2N^2T$ multiplication.

The backward procedure

Let a backward variables are the total provability seeing the rest of the observation sequence given that we were in state s_i at time t. We define it as the formula bellow.

$$
\beta_i(t) = P(o_t o_{t+1} ... o_T, X_t = i|\mu)
$$

The backward procedure computes backward variables which are defined by a recursive procedure as follows:

- Initialization
  $\beta_i(T+1) = 1, 1 \leq i \leq N$

- Induction

$$\beta_i(t) = \sum_{j=1}^{N} \beta_j(t+1) a_{ij} b_{ijo_t}, \quad 1 \le t \le T, 1 \le i \le N$$

- Total

$$P(O|\mu) = \sum_{i=1}^{N} \pi_i \beta_i(1)$$

Combining them

We can use any combination of forward and backward caching to work out the probability of an observation sequence.

$$\begin{aligned}
&P(O, X_t = i|\mu) = P(o_1 ... o_T, X_t = i|\mu) \\
&= P(o_1 ... o_{t-1}, X_t = i, o_t ... o_T|\mu) \\
&= P(o_1 ... o_{t-1}, X_t = i|\mu) \times P(o_t ... o_T|o_1 o_2 ... o_{t-1}, X_t = i, \mu) \\
&= P(o_1 ... o_{t-1}, X_t = i|\mu) \times P(o_t ... o_T|X_t = i, \mu) \\
&= \alpha_{\mathrm{i}}(t)\beta_i(t)
\end{aligned}$$

Therefore,

$$P(O|\mu) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t) \quad 1 \le t \le T+1 \tag{3.5}$$

Decoding

Commonly we want to find the most likely states that best explains for the given observed sequences O. This problem is equivalent to finding the $\underbrace{\arg\max}_{X} P(X|O, \mu)$. Since the sequence O is given, the problem is equivalent to finding $\underbrace{\arg\max}_{X} P(X, O, \mu)$. The algorithm to do this problem is the Viterbi algorithm [57].

Define:

$$\gamma_j(t) = \underbrace{\max}_{X_1 ... X_{t-1}} P(X_1 ... X_{t-1}, o_1 ... o_{t-1}, X_t = j|\mu)$$

- Initialization
  $\gamma_j(1) = \pi_j, \ 1 \le j \le N$

- Induction

  $\gamma_j(t+1) = \max \gamma_j(t) a_{ij} b_{ijo_t}, \ 1 \le j \le N$

- Store backtrace

$$\psi_j(t+1) = \underbrace{\arg\max}_{1 \le i \le n} \gamma_i(t) a_{ij} b_{ijo_t}, 1 \le j \le N$$

- Backtracking. The most likely state sequence is worked out from the right backwards:

$$\begin{aligned} X^*_{T+1} &= \underbrace{\arg\max}_{1\leq i\leq N} \gamma_i(T+1) \\ X^*_T &= \psi_{X^*_{T+1}}(t+1) \\ P(X^*_T) &= \underbrace{\max}_{1\leq i\leq N} \gamma_i(T+1) \end{aligned}$$

  In practical application, one can to work out not only the best state sequence but the $n-$best sequence or graph of likely paths. In order to do this people often store the $m < n$ best previous states at a node.

Parameters Estimation

We turn out to the third problem for HMM model. Given a certain observation sequence, the model $\mu = (A, B, \Pi)$ is need to find so that it is best explain for the given observation sequence. To solve this, Maximum Likelihood Estimation methods are used. It means that we want to find the values that maximize $P(O|\mu)$:

$$\underbrace{\arg\max}_{\mu} P(O_{\text{training}}|\mu) \tag{3.6}$$

It could not to choose μ s.t (9) by analytic method, however we can locally maximize it by an iterative hill-climbing algorithm namely Baum-Welch or Forward-Backward algorithm [58]. This algorithm is a special case of Expectation Maximization algorithm. It can be summarized as follow:
Having an initial model (e.g randomly chosen), we can work out the probability of the observation sequence to revise model. We can see which state transition and symbol emission were used the most. The revised model is chosen by increasing the probability of those so that it gives a higher the probability to the observation sequence.

Let $p_t(i,j), 1 \leq t \leq T, 1 \leq i,j \leq N$ be the probability of traversing a certain arc at time t given observation sequence O (see Figure 3.2).

$$\begin{aligned} p_t(i,j) &= P(X_t = i, X_{t+1} = j|O,\mu) \\ &= \frac{P(X_t=i,X_{t+1}=j|O,\mu)}{P(O|\mu)} \\ &= \frac{\alpha_i(t)a_{ij}b_{ijo_t}\beta_j(t+1)}{\sum_{m=1}^{N}\sum_{n=1}^{N}\alpha_m(t)a_{mn}b_{mno_t}\beta_n(t+1)} \end{aligned} \tag{3.7}$$

Therefore, if summing over the time index, it gives us expectations:
Let expected number of transitions from state i to j in O be E_{ij} and Let expected number of transitions from state i in O be E_i. We obtain:

$$\sum_{t=1}^{T} \gamma_i(t) = E_i$$

$$\sum_{t=1}^{T} p_t(i,j) = E_ij$$

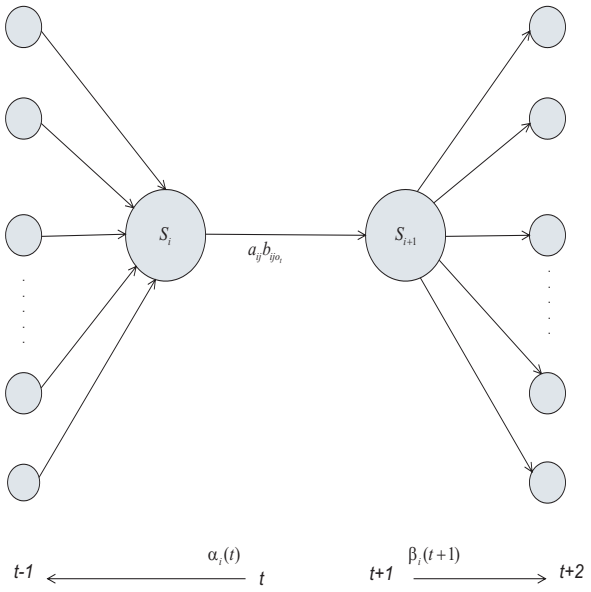

Figure 3.2: The probability of traversing an arc. Given an observation sequence and a model, we can compute the probability that the Markov process went from state s_i to state s_j at time t

With the initial model μ (chosen randomly, or preselected), we then run O through the current model to estimate the expectation of each model parameter. The model then is changed to maximize the values of the paths that are used a lot. This process will be repeated to obtain an optimal values for the model parameters μ. The re-estimation formulas are as follows:

$$\begin{aligned} \pi_i^* &= \gamma_i(1) \\ a_{ij}^* &= \frac{E_{ij}}{E_i} = \frac{\sum_{t=1}^{T} p_t(i,j)}{\sum_{t=1}^{T} \gamma_i(t)} \\ b_{ijk}^* &= \frac{E_{ijk}}{E_{ij}} = \frac{\sum_{\{t:O_t=k, 1\le t\le T\}} p_t(i,j)}{\sum_{t=1}^{T} p_t(i,j)} \end{aligned} \tag{3.8}$$

Therefore, the model $\mu = (A^*, B^*, \Pi^*)$, is derived from the model $\mu = (A, B, \Pi)$, and as proved by Baum, we have: $P(O|\mu^*) \geq P(O|\mu)$, the process will be repeat until it is satisfied the termination condition.

3.2 Maximum Entropy Model

3.2.1 The Principle of Maximum Entropy

The maximum entropy concept has a long history. Laplace enunciated the underling idea of the maximum entropy models in the principal of insufficient reason. This principle states that equal probabilities must be assigned to each competing assertion if there is

no positive reason for assigning them different probabilities. Jaynes [59] summarized the principle of maximum entropy very condensed following.

... in making inferences on the basic of partial information we must use that probability distribution which has maximum entropy subject to whatever is known. That is the only unbiased assignment we can make; to use any other would amount to arbitrary assumption of information which by hypothesis we do not have.

With having only partial information about the possible outcomes, one should choose the probabilities to maximize the uncertainty about the missing information and one should be as uncommitted as possible about missing information.

Assume that we are given a set of all possible distribution P and M constraints f_i. By applying the principle of maximum entropy, the most uniform distribution subject to the satisfaction of the given constraints is obtained. These constraints is used to select a mode p lie on the subset of P, that satisfied $E_p f_i = E_{\widetilde{p}} f_i$ for $i = 1, M$. Where $\widetilde{p}$ is a prior distribution observed from data. Since the uniform of a distribution p can be measure by its entropy, the best distribution p^* is the one with maximum entropy $H(p)$.

The appeal of maximum entropy principles are applied in many fields such as imagine processing, natural language precessing (NLP), etc. Many applications in NLP have been employed maximum entropy model such as machine translation, text summarization, and word sense disambiguation, etc. The following sections will lead the readers to the typical of maximum entropy, the conditional maximum entropy model and the answer why implying maximum entropy for NLP application are success.

3.2.2 Learning Maximum Entropy Models

Most problems in NLP can be considered as classification problems, in which the goal is to predict the class label $a \in A$ with some linguistic context $c \in C$. This can be solved by using a conditional probability $p(a|c)$. Although the co-occurrence of a and c can be extracted from a very large corpus, the probability $p(a|c)$ can not be calculated from it because the contexts are distributed sparely on NLP data. Maximum entropy models are one of the best ways to solve the problem.

Definition

Let $T = \{(a_1, c_1), (a_2, c_2), ..., (a_i, c_i), ...(a_N, c_N)\}$ be a training data set, where the action and context pair (a_i, c_i) is referred as an example and N is the number of examples in the training data. Let a *feature* be a function f:

$$f : A \times C \rightarrow \{0, 1\} \tag{3.9}$$

A feature $f(a, b)$ captures an information in b that might be useful for predicting a. Suppose that $\widetilde{p}(a, c)$ is the observed probability of the pair (a, c) in the training data set, $f_1, f_2, ..., f_k$ are features in the training data, and $E_{\widetilde{p}} f_j$ is the observed expectation of feature f_j:

$$E_{\widetilde{p}} f_j = \sum_{a,c} \widetilde{p}(a, c) f_j(a, c) \tag{3.10}$$

The model probability p^* is designed so that it is consistent with the observed expectation and likely to generalize well to unseen data. Using the principle of maximum entropy the model p^* with the highest entropy over the set of those models that are consistent with the expectation is recommended. As described in [60], the model p^* satisfies the following condition:

$$\begin{aligned} p* &= \underbrace{\arg\max}_{p \in P} H(p) \\ P &= \left\{ p | E_p f_j = E_{\widetilde{p}} f_j j = 1...k \right\} \\ E_p f_j &= \sum_{a,c} \widetilde{p}(c) p(a|c) f_j(a,c) \\ H(p) &= -\sum_{a,c} \widetilde{p}(c) p(a|c) \log p(a|c) \end{aligned} \tag{3.11}$$

Suppose that α_1, α_2,...α_k are the parameters of model ($\alpha_k > 0$ and associated with f_k). By using a Largraian representation, we obtained the form of the solution p^* described as follows:

$$\begin{aligned} p^*(a|c) &= \tfrac{1}{Z(c)} \prod_{j=1}^{k} \alpha_j^{f_j(a,c)} \\ Z(c) &= \sum_{a \in A} \prod_{j=1}^{k} \alpha_j^{f_j(a,c)} \end{aligned} \tag{3.12}$$

Here, $Z(c)$ is a normalization factor.

Let $L(p)$ be the log-likelihood of the training set according to the model p.

$$L(p) = \sum_{a,c} \widetilde{p}(a,c) \log p(a|c) \tag{3.13}$$

Because in the maximum entropy model, p^* will not assume anything beyond the evidence, and p^* will have a close fit to the observed data. Thus, we obtain an interesting relationship between maximum likelihood estimates of models from (3.13) and maximum entropy models. That is:

$$\begin{aligned} p^* &= \underbrace{\arg\max}_{q \in Q} L(q) \\ Q &= \{ p | p(a|c) = \tfrac{1}{Z(c)} \prod_{j=1}^{k} \alpha_j^{f_j(a,c)} \} \end{aligned} \tag{3.14}$$

The details of the maximum entropy framework and its duality with maximum likelihood estimates are discussed in [60].

Estimating Parameters

The simplest method for estimating maximum entropy model is the Generalized Iterative Scaling (GIS) algorithm.

Define $K_i = E_{\widetilde{p}} f_i$ the optimal α_i can be found by iteratively updating the model distribution p. Algorithm 2 shows the behavior of the GIS algorithm. It starts with arbitrary value of α_i. At each iteration, the algorithm updates α_i by comparing the expectation of f_i under the current p to the target value K_i. Then, the distribution p is re-estimate with the new value of α_i. The process is iterated until p converges.

Algorithm 1 The GIS Algorithm

1: Set $t = 1$
2: **while** p not converges **do**
3: Compute $K_i = E_{\tilde{p}} f_i$ for each f_i;
4: Update α_i according to the formula bellow
$\alpha_i^{t+1} = \alpha_i^t + \log \frac{K_i}{K_i^t}$
5: Define the next estimate function based on the new α_i

$$p^t(a|c) = \frac{1}{Z^t(c)} \exp(\prod_{i=1}^{k} \alpha_i^t f_i(a, c))$$
$$Z^t(c) = \sum_{a \in A} \exp(\prod_{i=1}^{k} \alpha_i^t f_i(a, c))$$

6: **end while**

The problem of the GIS algorithm is at computation time. It might takes a lot of times when the training data is big. For this reason, there were some other algorithms to estimate the maximum entropy mode including: the Improved Iterative Scaling (IIS) algorithm [60], and the Limited Memory BFGS algorithm (L-BFGS)[61], which is faster than GIS and IIS algorithm [62].

3.3 Support Vector Machine

Support Vector Machines are strong learning methods in comparision with decision tree learning and other learning methods. This was because their ability to generalize in high-dimensional spaces. The appeal of SVMs is based on their strong connection to the underlying statistical learning theory. That is, an SVM is an approximate implementation of the structural risk minimization (SRM) method [63]. For several natural language processing problems, SVMs have already been shown to provide a better generalization performance than traditional techniques [64].

Support vector machine (SVM)[63] is a technique of machine learning based on statistical learning theory. The main idea behind this method can be summarized as follows. Suppose that we are given l training examples (x_i, y_i), $(1 \leq i \leq l)$, where x_i is a feature vector in n dimensional feature space, y_i is the class label {-1, +1 } of x_i. SVM finds a hyperplane $w.x + b = 0$ which correctly separates training examples and has maximum margin which is the distance between two hyperplanes $w.x + b \geq 1$ and $w.x + b \leq -1$. The optimal hyperplane with maximum margin can be obtained by slowing the following quadratic programming.

$$\begin{aligned} \min \quad & \tfrac{1}{2} \|w\| + C_0 \sum_{i}^{l} \xi_i \\ s.t. \quad & y_i(w.x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{3.15}$$

where C_0 is the constant and ξ_i is called a slack variable for the non-separable case. In

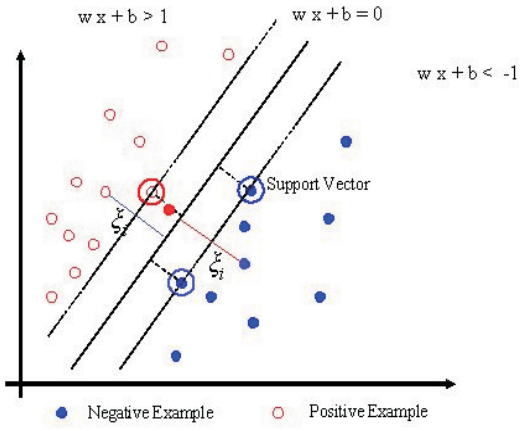

Figure 3.3: Overview of Support Vector Machine

final, the optimal hyperplane is formulated as follows:

$$f(x) = \text{sign}\left(\sum_{1}^{l} \alpha_i y_i K(x_i, x) + b\right) \tag{3.16}$$

where α_i is the Largrange multiple, and $K(x', x'')$ is called a kernel function, it calculates similarity between two arguments x' and x''. For instance, the Polynomial kernel function is formulated as follow:

$$K(x', x'') = (x'.x'')^p \tag{3.17}$$

SVMs estimate the label of an unknown example x whether sign of $f(x)$ is positive or not.

There are two advantage in using SVMs for classification as follows:

- High generalization performance in high dimensional feature space. SVMs optimize the parameter w and b of the separate hyperplane based on maximum margin strategy. It also grantees theoretically the low generalization error for an unknown example in high dimension feature [63].

- Learning with combination of multiple features is possible by virtue of polynomial kernel functions. SVMs can deal with non-linear classification.

SVMs are discriminative classifiers, and not generative multi classification models and probabilistic models like naive Bayes classifier or maximum entropy models. For this reason, there have been several attempts for extending SVMs to multi classification and to probabilistic models. Such models are experimented on several tasks and shows competitive in comparing with other multi class classification and probabilistic models.

3.4 Training Data in Statistical Machine Learning

Generating training data are very essential for many applications using statistical machine learning models. Training data can be generated by manually annotated or automatically.

For many natural language precessing tasks, a larger size amount of training data can be ensured its performance in term of high accuracy. For example, with using the big corpus e.g the Pentree Bank data for training, these tasks including pos-tagging, chunking and parsing have been showed a quite high performance. However generating training data by manually are cost and consuming time. For this reason, several attempts have been made in order to automatically generate training data for statistical machine learning approaches to natural language processing. The tasks of generating training data for machine translation were described in [65][66] and generating training data for text summarization were also presented in [10],[67],[68]. The next chapter will introduce the generating training data for our proposed text summarization system.

Chapter 4

Generating Training Data Using Decomposition Human-Written Summary Corpus

4.1 Introduction

The task of creating reliable data in support of the automatic text processing system is essential in several natural language processing. For this reason, several recent publications have focused on automatically creating training data from rough data, as described in [10], [65], [66], [67], [68].

When humans summarize text documents they generally use one of two approaches. In the first approach, they try to understand the text documents, then generate a summary in their own words. In the second approach, they reuse the original document and apply a cut and paste method. A sentence in the document might be reduced to a shorter sentence while its principal meaning is unchanged; other sentences might be selected for combining into one sentence.

Professional summarizers often use the second approach to summarize documents. Jing and McKeown [10] modelled a cut and paste automatic text summarization system. One important aspect in constructing a cut and paste summarization system is the decomposition human-written process. This process relies on aligning summary sentences with the original document and creating training data automatically from documents and the summary of documents for the automatic text summarization.

Marcu [67] presented an approach to automatically constructing of large-scale corpora for summarization research. This approach is essentially aligning the summary sentence with a semantically equivalent sentence in the document. It adopted an Information Retrieval-based approach coupled with discourse processing.

Brown et al. [65] and Gale and Kenth [66] both reported aligning sentences in a parallel bilingual corpus. They applied a Hidden Markov Model solution to bilingual corpora alignment. Jing and McKeown [68] initially addressed the decomposition of human-written summary sentences as the process of mapping summary sentences back to the original document. First, they considered that each summary word within the summary document would come from a word among a set of its occurrences in the original document. Second, they assumed that two consecutive words within the summary document should come from two nearby positions in the original document. With those assumptions, they

modelled the task of decomposing human-written summary sentences as a Hidden Markov Model based on a set of heuristic rules that they observed from human text-reusing practices. Afterwards, a dynamic programming technique, the Viterbi algorithm [57] was used to efficiently find the most likely document position for each word in the summary sentence and finally obtain the best decomposition for the summary document. Their work addresses an efficient method for the decomposition problem.

However, when humans produce a summary document, they may use some of their own words or phrases that are coincidentally similar to a word or a phrase in the original document. Moreover, summary documents usually have at least two repetition words, so the most likely document position may have two or more positions the same in the original document. To cope with these cases we extend Jing and McKeown's method by adding a semantic measure and checking the position of each word within a summary document. We use the semantic measure described in [69] to discover all words in the document that have the same meaning as a word within the summary sentences and to reduce the number of words in the summary which have no occurrence in the original. The purpose of the position-checking is to reduce the number of cases in which distinct words within a summary sentence come from the same position in the original document.

Therefore, we have to find the best sequence of word positions that minimizes the total number of repetition words.
In Section 4.2, we first mathematically formulate the summary sentence decomposition problem and then extend the HMM model in formulating the problem. Section 4.3 shows a position-checking algorithm for solving the case where the summary document has at least two repetition words. Section 4.4 presents a template HMM model using a suffix array for the human written decomposition problem. Section 4.5 discusses the performance of the decomposition of human-written summaries. Section 4.6 presents our conclusions and discusses some outstanding problems.

4.2 Decomposition Algorithm Using HMM

Jing and Mckeown [68][10] originally formulated the decomposition problem for human-written summary sentences. We have extended their formula by using a semantic measure [70][71][73][72] to solve the case in which a word in the summary document comes from a similar word in the original document.

4.2.1 Formulation

Let a summary document be represented as a word sequence: $\{I_1, I_2...I_N\}$, where I_j is the j^{th} word in N words of the summary document. L_j is defined as a set of features of the word I_j in the original document; M_j is defined as a number of features in L_j and $L_j[t]$ be the t^{th} feature in L_j .

Each word within the summary document may occur in the original document or be similar to a word in the original document. We denote the position of each word within the original document by the sentence position and the word position within this sentence. *Multiple occurrences* of a word I_j in the document can be represented by a set of word positions:

$$\{(s_{j_1}, w_{j_1}), (s_{j_2}, w_{j_2})...(s_{j_k}, w_{j_k})...(s_{j_m}, w_{j_m})\} \tag{4.1}$$

Where s_{j_k} is the sentence position of word I_j in the document and w_{j_k} is the word position within the sentence that has position s_{j_k} in a document and m is the total number of occurrences.

We use a semantic distance to define which words in the summary document are similar to a word in the original. The maximum of the semantic distance between two words is equal to 1 if they are the same. To define the semantic distance between two words we use the WordNet database with the method described in [70].

Let the *multiple similarity occurrence* of a word in the summary document be all its similar words in the original document. Thus, the *multiple similarity occurrence* of a word I_j in the document is expressed by:

$$\{(s_{j_1}, w_{j_1}, d_{j_1}), (s_{j_2}, w_{j_2}, d_{j_2})...(s_{j_m}, w_{j_m}, d_{j_m})\}. \tag{4.2}$$

Each feature $(s_{j_m}, w_{j_m}, d_{j_m})$ consists of a sentence position s_{j_m}, a word position w_{j_m} and a *semantic distance* d_{j_m} between word I_j and the word in the document that has sentence position s_{j_m} and word position w_{j_m}. The value of d_{j_m} is always greater than a *constant*. The value to be defined in our experiments is that $d_{j_m} = 1$ if a word at position (s_{j_m}, w_{j_m}) is an occurrence of I_j otherwise $d_{j_m} < 1$.

Using these notations, the decomposition problem can be formulated as follows:

Given a word sequence $\{I_1, I_2, .., I_j, ...I_N\}$ and multiple similar occurrences of word I_j $\{(s_{j_1}, w_{j_1}, d_{j_1}), (s_{j_2}, w_{j_2}, d_{j_2})...(s_{j_m}, w_{j_m}, d_{j_m})\}$, determine the most likely feature for each word in the sequence.

Figure 4.1 shows an example of the decomposition problem when considering the document "WSJ870702-0104" from DUC2001 training data and its summary document. This figure shows the task of choosing the most likely feature of each word in the sequence: an intellectual leader of. The multiple similar occurrence of the word "an" had 11 features as shown in the figure: {(99,23,1),(20,4,1),(129,21,1)}.

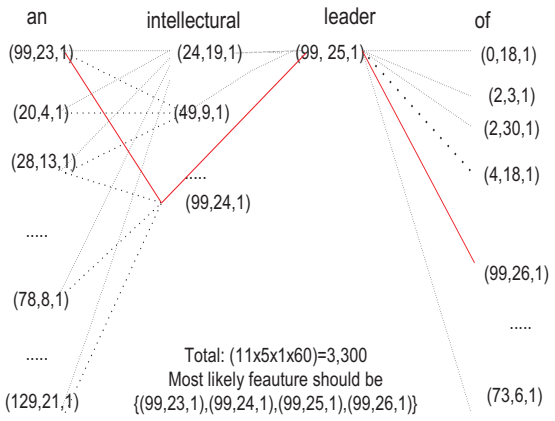

Figure 4.1: An example of decomposition problem

The most likely feature of the word "an" is (99, 23, 1). Similarly, the words intellectual, leader, of has 5 occurrences, 1 occurrence, 60 occurrences respectively. There are

3300 ways to generate the summary sentence "an intellectual leader of" from the original document. The dotted lines from "an" to "of" shows sequences of features. The continuous line shows the most likely feature sequence. To define the most likely document feature we extend the HMM from [68] based on heuristic rules and apply the Viterbi algorithm [57] to that HMM.

4.2.2 Heuristic rules

The question: "How does a word depend on the positions of the words surrounding it in the document?" emerges when humans summarize documents. Jing [68] pointed out the following heuristic rules:
(H1) Two adjacent words in a summary are most likely to come from two adjacent words in the document.
(H2) Adjacent words in a summary are highly likely to come from the same sentence in the document, retaining their relative precedent relation.
(H3) Adjacent words in a summary are highly likely to come from the same sentence in the document but in the reverse of their relative orders.
(H4) Adjacent words in a summary can come from nearby sentences in the document and retain their relative order.
(H5) Adjacent words in a summary can come from nearby sentences in the document and reverse their relative orders.
(H6) Adjacent words in a summary are not likely to come from sentences far apart.

Let X and Y be two word positions in a document: $(s_1, w_1); (s_2, w_2)$, respectively, and $F(X|Y)$ be a transition function between (s_1, w_1) and (s_2, w_2). Using the heuristic rules (H1-H6) $F(X|Y)$ is defined in Table 4.1, where the value of P_1 through P_6 are parameters and to be estimated manually.

4.2.3 HMM Solution

Let $Prob(I_i(s_i, w_i, d_i)|I_j(s_j, w_j, d_j))$ be the probabilistic transition between two features (s_i, w_i, d_i) and (s_j, w_j, d_j) of two words I_i and I_j separately. Since two words in the summary document should come from two words in the original document, the probabilistic transition is expressed by

$$\begin{aligned} &Prob(I_i(s_i, w_i, d_i)|I_j(s_j, w_j, d_j)) \\ &= F((s_i, w_i)|(s_j, w_j)) \times d_i \times d_j. \end{aligned} \tag{4.3}$$

Using the heuristic rules described in Table 4.1 we can calculate $F((s_i, w_i)|(s_j, w_j))$. Thus, two important aspects of defining the HMM model are as follows:
1) The state of the HMM is a feature that consists of sentence position, word position and semantic distance.
2) The probabilistic transitions between two features are defined in (1).
Using the Bigram model, the probabilistic transition function is defined as,

$$Prob(I_1, I_2, ...I_N) = \prod_{i=1}^{n-1} Prob(I_{i+1}|I_i). \tag{4.4}$$

The task of finding the maximal likelihood of word sequence $I : \{I_1, I_2...I_N\}$ is equivalent to defining a feature of each word within a sequence that maximizes (4.4).

Table 4.1: Transition function table

Rule 1:
if $((s_1 = s_2)$ and $(w_1 = w_2 - 1)$ **then**
$F(X|Y) = P_1$(the maximal value)
end if
Rule 2:
if $((s_1 = s_2))$ and $(w_1 < w_2 - 1)$ **then**
$F(X|Y) = P_2$ (the 2nd highest value)
end if
Rule 3:
if $((s_1 = s_2)$ and $(w_1 > w_2)$ **then**
$F(X|Y) = P_3$ (the 3rd highest value)
end if
Rule 4:
if $((s_2 - CONST < s_1 < s_2)$ **then**
$F(X|Y) = P_4$(the 4th highest value)
end if
Rule 5:
if $(s_2 < s_1 < s_2 + CONST)$ **then**
$F(X|Y) = P_5$ (the 5th highest value)
end if
Rule 6:
if $(|s_2 - s_1| >= CONST)$ **then**
$F(X|Y) = P_6$ (the smallest value)
end if

4.2.4 Parameters Estimating

Jing and MacKewon [68] estimated the values of P_1 through P_6 by manually selecting the values that give the best performance in the decomposition task. In the interest of speed, it is important to be able to define these values of P_1 through P_6 automatically. To define the values of P_1 through P_6 automatically, we use a GA algorithm, in which the objective function is the difference between outputs of the sequence of features using the Virterbi algorithm with the gold-standard sequence of features. The Viterbi algorithm uses P_1 through P_6 as parameters and then defines the values P_1 through P_6, thus the probability function (4.3) can be calculated directly. The semantic distance of two similar words was defined by using the WordNet database and the measures described in reference [70][71][72][73].

4.2.5 Position-checking algorithms

The position-checking algorithm solves the problem of finding a sequence of features that maximizes both position-checking and formula (4.4). The complexity of this problem is NP-Hard. To solve it we propose two heuristic algorithms. The first is an inside checking algorithm which integrates the position-checking with a dynamic programming algorithm (the Viterbi algorithm). The second is an outside position-checking algorithm. The Viterbi algorithm was modified to find the best sequences of features. The position-

checking algorithm was then used to select the sequence with the smallest possible number of repetition positions.

Inside position-checking algorithm

The inside position checking algorithm integrates a position-checking function during the running time of the Viterbi algorithm at each dynamic step. The algorithm tries to find a maximal likely subsequence of features while avoiding the case in which two identical features appear inside that sequence in each dynamic step. We describe some important aspects of the algorithm as follows:

Let $[i, j]$,$Seq[i, j]$, and $SCORE[i, j]$ be an index of a feature $L_i[j]$, a like sequence of features at $[i, j]$, and a maximal accumulated probabilistic value for $Seq[i, j]$, respectively. $BACK[i, j]$ is used by the backtracking process to the likely sequence $Seq[i, j]$. $BACK[i, j] = k$ if and only if $SCORE[k, j - 1] \times Prob(I_j = L_j[i]|I_{j-1} = L_{j-1}[k])$ is maximal.
Using this definition, we describe the Viterbi algorithm for decomposition of human written summary sentences as described in Algorithm 2.

There are three steps in the Viterbi algorithm for the decomposition problem: initialization, iteration and sequence identification. The algorithm is explained more detail by Viterbi [57].

To integrate a position-checking function during the runtime of the Viterbi algorithm, we modify it by performing position checking after each step of the dynamic programming algorithm.

If the summary document has at least two words the same, then the sequence of likely features may also have at least two positions the same. If the output sequence of likely features has two positions the same, then exist an index $[i, j]$ satisfies $Seq[i, j]$ has two features the same. Thus, the position-checking function is equivalent to the process of checking whether or not the sequence $Seq[i, j]$ has some features which are the same. To understand this, assume that I_m, I_n, and I_k are three repetition words in the summary document and consider the example in Figure 4.2.

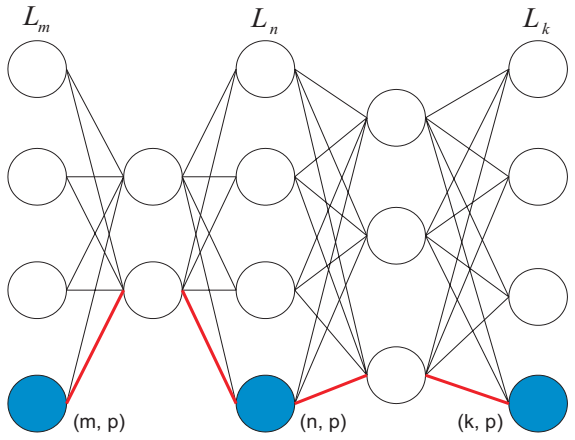

Figure 4.2: An example of repetition position problem.

Algorithm 2 The Viterbi algorithm

Input: Given a word sequence $I_1, I_2, .., I_j, ..., I_N$, and a list of possible template features L_j of each word I_j and Bigram probabilities.

Output: the most likely sequence of features
$P[1], P[2]..P[j]..P[N]$ with $(j, P[j])$ is position of word I_j.

{**Initialization Step**}

for $i = 1$ to M_1 **do**
  $SCORE[i, 1] = 1$; $BACK[i, 1] = 0$;
end for
for $j = 1$ to N **do**
  Sort each list L_j follow increasing of position
end for

{**Iteration Step**}

for $j = 2$ to N **do**
  if L_j is not empty **then**
    for $i = 1$ to M_j **do**
      $SCORE[i, j] = max(SCORE[k, j-1] \times Prob(I_j = L_j[i] | I_{j-1} = L_{j-1}[k])$ with $k = 1, M_{j-1}$
      $BACK[i, j]$ =index of k that gave the max
    end for
  else
    for $i = 1$ to M_{j-1} **do**
      $SCORE[i, j] = SCORE[i, j-1]$
      $BACK[i, j] = j - 1$
    end for
  end if
end for

{**Sequence Identification Step**}

$P[N] = i$ that maximizes $SCORE[i, N]$
for $i = N - 1$ to 1 **do**
  $P[i] = BACK[P[i+1], i+1]$
end for
for $j = 1$ to N **do**
  if L_j is empty **then**
    $P[j]$ =empty
  end if
end for

Figure 4.2 shows three identical features which are blue circles, suppose that they are indexed in L_m, L_n, and L_k as $(m,p),(n,p)$, and (k,p), respectively. The task of inside position-checking is to avoid the appearing of $(m,p),(n,p)$, and (k,p) in the likely sequence of features. To check the sequence $Seq(k,p)$, the naive algorithm has to back up to all positions j where words I_j and I_k are repetitions, while the efficient algorithm only backs up to the position j where I_j and I_k are repetitions and j is the first position before k. Let $K(j)$ be a set of elements i satisfying $I_i = I_j$; $i < j$ and $K(j)[t]$ be the t^{th} element ; $|K(j)|$ is the total number of elements in $K(j)$.

Algorithm 3 A naive checking position algorithm

1: $u = i; v = j;$
2: **while** $v >= K(j)[1]$ **do**
3: $u = BACK[u,v];$
4: **if** u in $K(j)$and $L_j[i] = L_v[u]$ **then**
5: return true;
6: **end if**
7: $v = v - 1;$
8: **end while**
9: return false;

Algorithm 3 shows a naive algorithm for position-checking. There are three steps from the line 3 to the line 7 in the Algorithm 3, thus it takes $3(j - K(j)[1])$ steps in the worst case .

To reduce the total number of steps, an efficient algorithm using stack operations was considered as follows.
Let $ST_w[i][t]$ be a stack corresponding to $L_j[i]$ with $I_j = w$ at a step t . The number of steps for position checking of a word w will be the number of occurrences of w within the summary sentence.

Algorithm 4 An efficient checking position algorithm

Input: i,j,t
Output: repetition or not
1: $v = j$;
$u = i;$
{back to previous repetition }
2: **while** $v >= K(j)[|K(j)| - 1]$ **do**
3: $u = BACK[u,v];$
4: $v = v - 1;$
5: **end while**{copy elements}
6: $ST_w[i][t] = ST_w[u][t-1];$
7: **if** Found $L_j[i]$ in $ST_w[i][t-1]$ **then**
8: return true;
9: **end if**
10: Push $L_j[i]$ into stack $ST_w[i][t];$
11: return false;

In Algorithm 4, the efficient algorithm only backs to the first repetition position and

finds all the elements in the stack at that position to check whether it is a repetition or not. Afterward, all elements will be stored in the current stack, for checking the following position. Thus, we only backtrack to $j - K(j)[|K(j)|]$ for position-checking at (i, j). From the line 3 to the line 4 the algorithm takes two steps and one step for the line 6. Note that the stacks are sorted and using binary searching (line 7) it takes $\log(|K(j)|)$ steps, since the number of elements in the stack $ST_w[i][t]$ is smaller than $|K(j)|$. The worst case in line 10 takes $\log(|K(j)|)$ steps. Thus, the total number of steps is $2(j - K(j)[|K(j)|]) + 2 \times \log(|K(j)|) + 1$. We can say that the Algorithm 4 is always efficiency than the Algorithm 3 by following expression.

Let T_{AL2} and T_{AL3} be the computational steps of Algorithm 3 and Algorithm 4 respectively:

$$\begin{aligned} T_{AL2} = 3(j - K(j)[1]) = j - K(j)[1] \\ +2(j - K(j)[|K(j)|]) \\ +2(K(j)[|K(j)|] - K(j)[1]) \end{aligned} \tag{4.5}$$

$$\begin{aligned} T_{AL3} = 2(j - K(j)[|K(j)|]) \\ +2 \times \log(|K(j)|) + 1 \end{aligned} \tag{4.6}$$

Since $K(j)[|K(j)|] - K(j)[1] > |K(j)|$ then the equation above is given by.

$$\begin{aligned} T_{AL2} > j - K(j)[1] + 2(|K(j)|) \\ +2(j - K(j)[|K(j)|]). \end{aligned} \tag{4.7}$$

Since $j - K(j)[1] > 0$ and $2(|K(j)| > 2(log(|K(j)|) + 1$
then the right hand side of (A.3) is expressed by.

$$\begin{aligned} j - K(j)[1] + 2(|K(j)|) + \\ 2(j - K(j)[|K(j)|]) \\ > 2(j - K(j)[|K(j)|]) + 2log(|K(j)|) + 1. \end{aligned} \tag{4.8}$$

From (A.3) and (A.4),we have

$$\begin{aligned} T_{AL2} > 2(j - K(j)[|K(j)|]) + \\ 2 \times \log(|K(j)|) + 1. \end{aligned} \tag{4.9}$$

From (A.2) and (A.5), we obtain $T_{AL2} > T_{AL3}$.

Outside position-checking algorithm

The goal of this algorithm is to use position checking on the output from the Viterbi algorithm. We firstly modify the Viterbi algorithm to obtain the multiple best sequences of features that satisfy the set of heuristic rules. Afterwards, a sequence among the multiple best sequences of features whose has the smallest number of repetition features is selected by a searching algorithm.

Table 4.2 shows an extension of the Viterbi algorithm in order to obtain the best sequences features. Since we may have multiple sequence of features that $SCORE[i, j]$ are maximize. We extend the structure of $BACK[i, j]$ in the Viterbi algorithm as an array,

while it is an element in the original algorithm. The method of obtaining $BACK[i,j]$ is to add all indices k that maximize $SCORE[i,j]$. To obtain a feature sequence that has a small enough number of repetition features we rely on an optimal searching problem. We use the A* algorithm and a GA algorithm for handling this process with the target function as the number of repeated features within a sequence.

Table 4.2: Differences between algorithms

Viterbi Algorithm	The Modified Algorithm
$BACK[i,j] = k$ that gave $SCORE[i,j]$ maximal	$BACK[i,j]$= Set of all indices k that gave $SCORE[i,j]$ maximal

Therefore, the main difference between this algorithm and the inside position-checking algorithm is the use of the position-checking strategy. One uses it during the running time of the Viterbi algorithm, the other is uses it on the result of the modified Viterbi algorithm.

4.2.6 Template HMM using Suffix Array

Template HMM Model

The HMM model described in Section 4.2 should be extended for the following reasons:
1. To handle the case in which some phrases in a summary document may come from an occurrence or a paraphrase in the original document.
2. Working on phrases may reduce the total number of repetition positions. In addition, information from a phrase is preferable to information from a word.

Following these points, each hidden state in the HMM is to be considered as an occurrence of a phrase within the summary document for a new HMM model. The position-checking algorithm is then applied to the new HMM model in the decomposition problem.

We use a suffix array [75] to define all occurrences of a substring within the summary document in the original document. The execution time is *O(w + log N)*, where w is the length of substring and N is the length of the original document. Then, the decomposition problem of the word sequences: $\{I_1, I_2, ..., I_N\}$ is equivalent to the decomposition of a "template" sequence: $\{T_1, T_2, ..., T_N\}$. Let each template T_k be defined as: $T_k = \{\{I_k\}; \{I_k, I_{k+1}\}...; \{I_k, I_{k+1}..., I_N\}\}$.
A multiple occurrence of template T_k is all occurrences of each substring that start from index k and their similar occurrences in the original document. To define a similar occurrence of a phrase we used a small data set of our own paraphrases database. Using a suffix array we are able to define all occurrences of T_k. Thus, the task of finding the likelihood feature sequence for the word sequences $I : \{I_1, I_2...I_N\}$ is equivalent to obtaining the maximum value of Equation (4.10) by the Viterbi algorithm.

$$Prob(I_1, I_2, ..I_N) = \prod_{i=1}^{N-1} Prob(T_{i+1}|T_i) \tag{4.10}$$

First, T_k is an interval of position in the original document and the semantic distance of two words should be extended to the semantic distance between two phrases.

Second, we design a probabilistic transition between two template states T_k and T_l as follows:
Assume that T_k and T_l are two intervals in the original document and to be defined as $T_k[(x_k, y_k), (x_k, y'_k)]$ and $T_l[(x_l, y_l), (x_l, y'_l)]$ respectively, in which (x_k, y_k) is a position as mentioned in the formula section. The probability $Prob(T_k|T_l)$ depends on the assignment between two intervals of each element T_k and T_l . $Prob(T_k|T_l)$ will be defined by using a function $F(T_k|T_l)$ for two intervals T_k and T_l as shown in Table 3.

The values of P_1 through P_7 can be estimated by using the algorithm described in

Table 4.3: Template transition function table

Rule 1:
if $(x_k = x_l)$ and $(y'_k = y_l - 1\ )$ **then**
$F(T_k|T_l) = P_1$ (maximal value)
end if
Rule 2:
if $(x_k = x_l)$ and $(y'_k < y_l - 1\ )$ **then**
$F(T_k|T_l) = P_2$ (the 2nd highest value)
end if
Rule 3:
if $(x_k = x_l)$ and $(y_k > y_l\ )$ **then**
$F(T_k|T_l) = P_3$ (3rd highest value)
end if
Rule 4:
if $(x_l - CONST < x_k < x_l\ )$ **then**
$F(T_k|T_l) = P_4$ (4th highest value)
end if
Rule 5:
if $(x_l < x_k < x_l + CONST)$ **then**
$F(T_k|T_l) = P_5$ (5th highest value)
end if
Rule 6:
if $(|x_k - x_l| >= CONST)$ **then**
$F(T_k|T_l) = P_6$ (small value)
end if
Rule 7:
if $(x_k = x_l\ )$and $(y_k < y_l < y'_k\ )$ **then**
$F_(T_k|T_l) = P_7$ (2nd small value)
end if

Section 4.2.4. This process is similar to the process of estimating parameters for the normal model. Using the function above, we can calculate the transition probability $Prob(T_{i+1}|T_i)$ directly. We assume that all notations in the HMM model have the same meaning in the template-HMM model.

In Equation (4.10) when T_k is an occurrence of the substring $I_k, I_{k+1}, ...I_{k+l}$ then the following states $T_{k+1}, T_{k+2}, ...T_{k+l}$ should be set to a "special" state because each element I_{k+l} cannot appear in two different states. The number of following "special" states is equal to the length of state T_k. The "special" state will store all the information of its

previous state and the transition between the "special" state and following state is equal to the transition between the previous state and the following state. In addition, The number of "special" states at L_k will be equal to the number of states in L_{k-1}, which has an interval length greater than 0 and $Prob(T_k|special) = 1$.

The Template-Viterbi Algorithm

To satisfy the new requirements of the template model, the Viterbi algorithm is modified to a new algorithm, a Template-Viterbi algorithm. For the template-viterbi algorithm, each template feature t will contain two information values that are the *len* of intervals and the *prev* value, they are defined as $t.len$ and $t.prev$ respectively . The *prev* value is pointed to the previous feature during the dynamic process.

Algorithm 5 The Template-Viterbi algorithm

Input: Given a word sequence $I_1, I_2, .., I_j, ..., I_N$, and a list of possible template features L_j of each word I_j and Bigram probabilities.

Output: the most likely sequence of features
$P[1], P[2]..P[j]..P[N]$ with $(j, P[j])$ is position of word I_j.

{Initialization Step}

The same Viterbi-algorithm

{Iteration Step}

for $j = 2$ to N **do**
  if L_j is not empty **then**
    $num = M_j$ {number of features}
    push all elements in L_{j-1} has length is greater than 0 into L_j
    for $i = 1$ to M_j **do**
      if $L_j[i].len > 0$ and $i >= num$ **then**
        $L_j[i].len = L_j[i].len - 1$
        $SCORE[i, j] = SCORE[L_j[i].prev, j - 1]$
        $BACK[i, j] = L_j[i].prev$
      else
        $SCORE[i, j] = max(SCORE[k, j - 1] \times Prob(I_j = L_j[i]|I_{j-1} = L_{j-1}[k])$ with $k = 1, M_{j-1}$ and $L_{j-1}[k].len > 0$
        $BACK[i, j]$ =index of k that gave the max
      end if
    end for
  else
    Push all elements in L_{j-1} has length is greater than 0 into L_j
    for $i = 1$ to M_{j-1} **do**
      $SCORE[i, j] = SCORE[i, j - 1]$
      $BACK[i, j] = j - 1$
    end for
  end if
end for

{Sequence Identification Step}

Similar to the Viterbi algorithm

Algorithm 5 described a Template-Viterbi algorithm for the template-HMM model.

The initialization step and the sequence identification step are similar to the Viterbi algorithm. The difference between two algorithms is at the iteration step. In the Template-Viterbi algorithm at the step j all elements in $L_{j-1}[k]$ satisfying $L_{j-1}[k].len > 0$ with $k = 1, M_{j-1}$ will be stored into $L_j[k]$ and their length interval will be decreased by 1, as in line 5, line 7 and line 8. All the other elements in L_{j-1} will be used in the finding maximal process from the line 10 to the line 11.

Example

Assume that the phrase *"there is a"* comes from the original document at the following positions {(1,2), (1,3), (1,4)}. The template model includes three elements:
$T_1 : \{[(1,2),(1,4)],[(1,2),(1,3)],[(1,2),(1,2)]\}$
$T_2 : \{[(1,3),(1,4)],[(1,3),(1,3)]\}$
$T_3 : \{[(1,4),(1,4)]\}$
Two sequences for the template model will be:
$S_1 : \{[(1,2),(1,4)], special, special\}$
$S_2 : \{[(1,2),(1,3)], special, [(1,4),(1,4)]\}$
In the sequence S_1 the states [(1,2),(1,4)] are followed by the "special" state and all its information is to be stored in the "special" state.
Thus, $Prob(S_1) = Prob([(1,2),(1,4)]|special) \times Prob([(1,2),(1,4)]|special) = 1$.
In the sequence S_2 the state [(1,2),(1,3)] is also followed by the "special". Thus, $Prob(S_2) = Prob([(1,2),(1,3)]|special) \times Prob([(1,2),(1,3)]|[(1,4),(1,4)])$
$= Prob([(1,2),(1,3)]|[(1,4),(1,4)])$.
Thus, the sequence S_1 is preferred to the sequence S_2. Although the two likely position sequences are the same (1,2),(1,3),(1,4), the information stored in the last state of S_1 is a phrase [(1,2),(1,4)] while in the last state of S_2, it is just a word [(1,4),(1,4)].

The position-checking algorithm with a suffix array may improve the accuracy because it inherits the advantages of the position-checking algorithm while additionally working on intervals. This results in reduced probabilities for exiting repetition features in the output. The difference between the two algorithms described above is that the position-checking algorithm is applied in a new style of HMM compared with the old model. Both the inside and outside position-checking algorithms can be applied to that HMM model in the decomposition problem.

4.3 Experiments

4.3.1 Experimental environment

For our experiment, we used the entire DUC2001 training data includes 319 pairs of documents. The average number of words in a summary document was 100. The average number of words in an original document was 973.7 . The average number of occurrences of words and the number of repetition words in the original documents was 8.24 and 39.04 respectively. These parameters are large enough for checking-point algorithms. In the second experiment, we used the telecommunication corpus that we obtained from free daily news service Communication-Related Headlines provided by Benton Foundation (http://www.benton.org).

The original Viterbi algorithm and our modified Viterbi algorithm were tested on a Pentium III PC, 1,200MHz in a Windows XP environment.

4.3.2 Experiment of parameters estimating

We used the training extract directory in the DUC2001 data to estimate parameters for our HMM model. This directory contains human-generated extracts for single document summarization based on the DUC01 training data (total 146 documents). The extracts are sentence -based and were derived to cover the same information content given in the DUC01 training abstracts. this set of extracts and summary documents was used to generate the corresponding sequence of features by applying our decomposition program with the default parameters ($P_1 = 0.9, P_2 = 0.8..., P_6 = 0.3$). Afterward, we manually corrected each sequence of features generated by our decomposition method.

The results became gold-standard data and were used for the training and evaluation processes. The GA algorithm as described in Section 4.2.4 was used to estimate the parameters' values for both the normal model and the template model. The training data are the set of documents and their summaries which correspond to the gold-standard sequence of features.

Our population size for the GA algorithm was 200 chromosomes and the number of generations of GA algorithm was limited to 40. Each gene in our experiment is a vector and must satisfy a condition that its order is decreased. Parameter values for both the normal model and the template model were obtained after running the GA algorithm.

4.3.3 Experiment accuracy

We conducted experiments to evaluate the accuracy of the decomposition task as follows. The corrected outputs of the decomposition task were used to evaluate the accuracy of our decomposition results. We randomly selected 32 documents and their summaries from the total of 319 documents. The WordNet database was used to randomly replace words with their synonyms. We only replaced nouns because the only four main grammatical objects in the WordNet database are: *"noun", "verb", "adjective"* and *"adverb"*, and summary documents are rich in nouns. Finally, corrected decompositions were made for these documents. Table 4.4 reports the average number of output repetition positions after using

Table 4.4: Repetition output results of algorithms

Algorithms	Arg.Repetition Words	Execution.time
Baseline	5.24	24.85(second)
Ins.PC	0	29.54(second)
Out.PC	2.60	31.29(second)
Temp.Suff	1.60	36.41(second)

the baseline from [68] along with inside check, outside check, semantic distance and combination between outside position-check algorithm and suffix array indexing algorithms. The number of output repetition features in the baseline algorithm is 5.24. The total number of output repetition features is reduced when the position-checking algorithm is

applied. The inside position-checking algorithm reports zero repetition features because this method always avoid the case that two same features appear in the process of finding a best sub-sequence. The result of the outside algorithm is 2.6 because the process of optimal searching tries to find a sequence that has total repetition features as few as possible. This number will be reduced if we increase the times for searching. The algorithm using position-checking with a suffix array has the smallest total number of repetitions (1.6) because it inherits the advantage of the position-checking algorithm. In addition, this algorithm uses substrings for indexing so, the probability of exiting with repetition features in the output is reduced. The execution times in Table 4.4 show that all of the algorithms are fast enough. Our algorithms are slower than the execution times of the original algorithm but that difference is acceptable.

Table 4.5: Occurrence words

Parameter.Eval	Base Line	Semantic Distance
Avg. occurrences	20.82	13.2

The average of the total number of repetition positions that appeared in the output of a decomposition task for each summary sentence was 5.24. There were no repetition positions (features) in the output from the decomposition task when using the position-checking function.

Table 4.5 shows that the average of the total number of words that had no occurrence in the original document was 20.82, and after applying the semantic distance measure, this value decreased to 13.2 words. This was because some words in the summary document corresponded semantically to some words in the original document. We suspect that this value would be decreased if the semantic distance measure was extended by measuring words in the summary document against phrases in the original document.

To compare the accuracy of the two methods, we manually produced the correct output for each document, and evaluated the accuracy for each output of the decomposition task. We use three measures as follows,

$$precision = \frac{total\ correct\ phrases}{total\ phrases\ in\ the\ output\ sequence}$$

$$recall = \frac{total\ correct\ phrases}{total\ phrases\ in\ the\ gold\text{-}standard\ sequence}$$

$$F - measure = \frac{2 \times precision \times recall}{precision + recall}$$

The baseline method does not combine the semantic distance measure and position-checking function. We tested four methods and tested on the same data in order to compare their accuracies. We also compared the semantic distance method integrated with the inside position-checking function and the outside position-checking separately with the original algorithm.

Table 4.6 shows the decomposition accuracies of four algorithms: Baseline algorithm, inside position-checking, outside-position checking, and outside position checking on the template model using suffix array. The algorithm using position-checking on the template

Table 4.6: Decomposition Accuracy result

Algorithm	Precision	Recall	F-measure
Baseline	0.791	0.700	0.738
Ins.PC	0. 841	0.741	0.785
Out.PC	0. 845	0.745	0.786
Temp.Suffix	0. 873	0.770	0.810

HMM model gave the best accuracy (0.87) because this algorithm inherits the advantage of a position-checking algorithm and the advantage of defining a subsequence of words within the summary document in the original document. Therefore, our algorithms ensure both accuracy and execution speed when compared to the original algorithm. The accuracy is improved and the execution times are sufficiently fast.

4.3.4 Human Judgments of Decomposition Results

In this portion of the experiment, we re-ran Jing and McKewon's experiments by using human judgment for the decomposition task in a telecommunications corpus [68]. We first selected 50 summaries from a telecommunication corpus and ran the decomposition program. We then asked humans to judge the decomposition results, in order to compare our results with the original .

The judge was asked to decide whether the decomposition results were correct. A result was considered correct when all three questions posed in the decomposition were correctly answered. The decomposition program needed to define three questions: (1) Is a summary sentence constructed by reusing the text from the original document? (2) If so, what phrases in the sentence come from the original document? (3) From what part of the original document do the phrases come?

The 50 summaries contained a total of 300 sentences. The accuracies of the algorithms as determined by human judgment are computed by the rate of total correct phrases and total sentences.

Table 4.7 shows the decomposition results using human judgment of the baseline method, the inside position-check, the outside position-check, and the template HMM. Table 4.7 indicates that our algorithms outperformed the baseline algorithm in decomposing summary documents. The template HMM archived the best accuracy. The results of the baseline, the inside position-check and the outside position-check were not so much differences. This was because in the telecommunication corpus, human prefers use paraphrases to express meaning of a phrase than use synonyms to express meaning of a word. The template HMM outperformed other methods in decomposing summary documents because it can use paraphrase database in defining a subsequence of words within the summary document in the original document.

4.4 Conclusions

In this paper, an HMM solution was extended for the decomposition problem, to adapt to the case in which a word within a summary sentence does not appear in the original

Table 4.7: Human Judgment of Decomposition Results

Baseline	Ins.PC	Out.PC	Template.Suffix
0.913	0.914	0.916	0.943

document. The Viterbi algorithm was modified by adding position-checking to prevent errors when a likely feature sequence has at least two identical features. The template HMM model using suffix array which has the advantage of the position-checking algorithm and also utilizes rich information from phrases was also presented.

The experiment using DUC2001 data and telecommunication corpus showed that our methods were more accurate than the baseline algorithm for the test data. We believe that with a good semantic distance measure between two phrases, the decomposition task will be further improved.

Work on extending the semantic measure for the decomposition task is currently underway. Use of a fixed model compared with the Bigram model shows promise.

Chapter 5

Sentence Extraction Based Statistical Learning

5.1 Introduction

Sentence extraction is the task of identifying important sentences in the text. The majority of early extraction research focused on the development of relatively simple surface-level techniques that tend to signal important passages in the source text. Typically, a set of features is computed for each passage, and ultimately these features are normalized and summed. The passages with the highest resulting scores are sorted and returned as the extract. Early techniques for sentence extraction computed a score for each sentence based on the features such as positions in the text [77], word and phrase frequencies [79], key phrases (e.g., "In conclusion...") [11]. Recent extraction approaches use more sophisticated techniques to determine which sentences to extract; these techniques often rely on machine learning to identity important features, on natural languages analysis to identify key passages, or on relations between words rather than bags of words.

The application of machine learning to summarization was pioneered by Kupiec [36]. In this work they developed a summarizer using a Bayesian Classifier to combine features from a corpus of scientific articles and their abstracts. Aone et al. [76] and Lin [83] experimented with other forms of machine learning and its effectiveness. Learning individual features has been also reported by Lin and Hovy [84]. In these tasks, the affect of position sentences, important words and phrases to the selection of sentences were investigated. Some recent works [40] has turned to the use of Hidden Markov Model (HMMs) and pivoted QR decomposition to reflect the fact that the probability of inclusion of a sentence in an extract depends on whether the previous sentence has been included as well. Hirao and Matsumoto [38] applied support vector machines to sentence extraction and showed an advantage in comparison with earlier sentence extraction methods because of using high dimension space of features. Osborne [13] proposed an alternative approach to sentence extraction using a maximum entropy model. The author indicated that with a set of dependent features, maximum entropy models were not only suitable for sentence extraction but also outperformed sentence extraction using naive Bayses classifier.

Although using machine learning to sentence extraction is one of the best approaches, training data for the learning purpose are not much available. For this reason, several researchers have attempted to produce training data automatically based on text documents and their summaries [25],[67]. The results for this task are good for some kinds of

text documents (e.g news), but human corrections are still required.

Co-training are considered as suitable methods in dealing with unlabeled data to enhance the performance of a learning task [78]. For example, they have been successful applied to various natural language processing problems such as word sense disambiguation [85], named entity recognition[86], noun phrase bracketing [87], and statistical parsing [88]. In this chapter, we will show the potential of co-training method in dealing with unlabeled data for sentence extraction task. We also propose a co-training version based on maximum entropy classification so called Co-MEM and indicate that Co-MEM is a suitable technique for sentence extraction task.

The rest of this chapter is structured as follows. Section 5.2 introduces the sentence extraction using MEM. Section 5.3 presents Co-training technique for sentence extraction. Section 5.4 presents implementation and experimental results; Section 5.5 gives our conclusions and presents some remaining problems to be solved in our future works.

5.2 Sentence Extraction using MEM

As present in Chapter 3, the parametric form a conditional maximum entropy is as follows [60]

$$P(c|s) = \frac{1}{Z(s)} \exp(\sum_i \lambda_i f_i(c, s)) \tag{5.1}$$

$$Z(s) = \sum_c \exp(\sum_i \lambda_i f_i(c, s)) \tag{5.2}$$

Here c is a label (from a set of label C) and s is a sentence we are interested in labelling. C consists of two labels: one indicating that a sentence should be in the summary ('true') and another label indicating that the sentence should not be in the summary ('false'). Using maximum entropy model, the training data for sentence extraction are viewed as a set of features. For example, a feature captured the idea that the abstract-worthy sentence contain the words "in conclusion" can be expressed as follows;

$$f_i(c, s) = \begin{cases} 1 & \text{if } s \text{ contains "in conclusion" and } c = true \\ 0 & otherwise \end{cases} \tag{5.3}$$

The problem of maximum entropy modes (MEMs) is how to estimate the weight values which are associated with features. The detail of the maximum entropy model are discussed in Chapter 3. The key problem of using maximum entropy is how to determine features and feature selections. The following subsections will present the feature sets and feature selections for our sentence extractions task.

5.2.1 Features

Feature sets for our maximum entropy are selected based one some methods bellow:

- **Location method:** Including the position of sentences within documents. These sentences in the beginning or in the end of a given text document are highly relevant to the text's gist meaning.

- **Length method:** A short sentence is preferred to be an important sentence. The length here means the number of words in the sentence. We use there value of the length sentence, whether a sentence was less than 6 words, whether it was greater than 20 words, or whether it was between two ranges. The feature encoded whether a previous sentence was less than 5 words or longer is also used. This was because it captures the idea that summary sentences tend to follow headings (which are short).

- **Relevant to title:** These sentences are similar to the title of a given texts are likely to be an important sentence. We compute the number of words in the title occurs in the sentence. It was normalized by the length of the sentence. We used four values: no title, $> 0.01, > 0.05$, and > 0.1.

- **Term frequency and document frequency (tf-idf):** Term frequency ($freq_i$), i.e the number of times a word appears in the document. This is a measure of how salient the word is with the document. Document frequency (n_i) is the number of documents in which the word a appears. This measures how informative the words is: If a word appears many times in only one document then it is informative. Assume that the number of documents in corpus is N, the ratio between the two frequencies is compared in order to find thematic words:
$score(w_i) = freq_i \times log(N) - log(n_i)$

- **Cue phrase:** A sentence used some phrases which emphasize the author's view can be an important sentence. Such a phrase is called cue phrase. For instance the phrase "In this paper we..." suggest that the sentence will include the aims of the paper.

- **Distance:** Distance of a word within a sentence to its previous occurrence has been shown useful for sentence extraction. The features are used to recognize the style of human in writing [39]. We divide the five following values for the features. ≤ 1, 2-3, 4-7, 8-15, and 16+.

- **Similar distance occurrences:** Distance of a word within a sentence to its previous similarity. Five values: ≤ 1, 2-3, 4-7, 8-15, and 16+ are used in the similar distance feature.

- **Section Structure:** Sentence in section (e.g., Introduction or Conclusion) are distinguished according to their occurrences. We used three values, occurs in first, second, or last of paragraph.

- **Name Entity:** The boolean value '1' indicates that a certain Named Entity class appears in a sentence. There are eight Named Entity classes:
PERSON, LOCATION, ORGANIZATION, ARTIFACT, DATE, MONEY, PERCENT, TIME.

- **Compound words:** Using a list of compound words, a sentence will be decomposed into a sequence of compound words. They are used a features.

In final, we deleted any feature that occurred less than 4 times.

5.2.2 Summary Size

A summary size is considered as a number of words or number of sentences in a summary. Controlling summary size is interested in almost every works on sentence extraction. This subsection will address a method of using maximum entropy mode to control a size of sentence extractions. The natural way is that we rank all sentences by decreasing score obtained from a measure method. We then obtain the top k sentences so as to their size is equal to the summary size. For a given sentence s, using conditional maximum entropy we could obtain two scores by the probabilities $P('true'|s)$ and $P('false'|s)$. A sentence obtain a high score $P('true'|s)$ and a small score $P('false'|s)$ is likely chosen as an important sentence. In contrast, the sentence with a high score $P('false'|s)$ and a small score $P('true'|s)$ is likely to be a unimportant sentence. This intuition motivates a score measure for a sentence s as $P('true'|s) - P('false'|s)$.

5.3 Co-Training Sentence Extraction

5.3.1 Co-Training Algorithm

Co-training is one of a weakly supervised learning technique which uses an initial small amount of labeled data to automatically bootstrap larger sets of automatically training data. It is in general applied to the problem where there are two distinct views of example in the data-set. The algorithm learns separate classifiers over each of the view, and augments a small set of labeled example by incorporating unlabeled example. In the final, it combines their predictions to decrease classification errors.

Algorithm 6 shows the outline of the co-training algorithm in which the two distinct views V_1 and V_2 are selected to learn labeled and unlabeled data. Two classifiers h_1 and h_2 over each view are then upgraded incrementally with a few initial labeled examples. At every iteration, each classifier chooses unlabeled examples and adds them to the set of labeled examples, L. The selected unlabelled examples are those which each classifier can determine their labels with the highest confident. After that, the classifier are trained again using the new labeled examples. The final classifier are obtained by combining of the two classifiers of the two views. Assume that the probabilities to obtain a class c_j for a given example x of view V_1 and V_2 are $P_{h_1}(c_j|x)$ and $P_{h_2}(c_j|x)$, respectively. The probability of the final model can be then estimated as

$$P(c_j|\mathrm{x}) = P_{h_1}(c_j|\mathrm{x})P_{h_2}(c_j|\mathrm{x}) \tag{5.4}$$

5.3.2 Two Views and Selection Examples

Two Views

The set of features mentioned above are divided into two views as follows:

- Distance feature including all features of a sentence which are relevant to other sentences such as distance occurrence, and similar distance occurrence.
- Isolated Feature: The single features of a sentence, it includes all information feature for a sentence.

Algorithm 6 An outline of the co-training algorithm.

Given L: a set of labeled examples, and U : a set of unlabeled examples.

while $|U| > 0$ **do**
  Train classifier h_1 on view V_1 of L.
  Train classifier h_2 on view V_2 of L.
  Allow h_1 to determine labels of examples in U
  Allow h_2 to determine labels of examples in U
  Determine $U' \subset U$, whose elements are most confidently labelled by h_1 and h_2.
  $U = U \backslash U'$
  $L = L + U'$
end while

Table 5.1: Two views

Feature name	Relation views	Isolated views
Position	N	Y
Sentence Length	N	Y
Named Entity	N	Y
Section structure	N	Y
Similar distance	Y	N
Distance	Y	N
Tif-dif	N	Y
Cue phrase	Y	N
Relevant to title	Y	N
Compound noun title	N	Y

We divide a set of features into two views including the relation view and the isolated view which are as shown in the Table 5.1. A feature which belongs a view was noted 'Y'. Otherwise, it was noted 'N'.

Using each view to extract features we then have two maximum entropy models correspond with each view.

Example Selection

A unlabeled example is considered to be a training example, if both the maximum entropy models are highly agreement in labeling or the measure score is greater than a threshold. Since it is difficult to determine the threshold, we use a simple method as follow. The selected unlabeled examples are those that each classifier can determine their labels with the highest confident. We sort all unlabeled examples according to their values of agreement and take the k top ones. In our experiment, k was set to 50.

5.4 Experiments

To evaluate the performance of the proposed sentence extraction algorithm, we implement it on C++ with Windows XP environment and test it on the DUC data and the Comp-lang data [37]. These were 80 conference papers, taken from the Comp-lang preprint achieve, and converted it to XML format. The XML annotated documents were then additionally manually marked-up a tags indicating the status of various sentences. Here are some properties of the documents. On average, each document contained 8 sentences that were marked as being abstract-worthy. The document on average contained in total 174 sentences. We randomly selected 70 documents to generate training data for our maximum entropy models in which the average number of examples is 11,077. The remaining 10 documents are used for testing the performance of extraction task.

We used two different experiments as bellow: In the first experiment, we use our Co-MEM method with a small number of training data and a larger number of unlabeled data in order to compare with the sentence extraction based maximum entropy using a larger number of labeled data. In the second experiment, we want to know how unlabeled data can effectively increase the performance of sentence extraction using a small number of training data.

For the first experiment, we selected randomly a subset of 6,000 labeled examples in the total 11,077 examples as a training data. The remaining examples were served as unlabeled data for the Co-MEM algorithm. With these data, we estimate the MEM model and Co-MEM for sentence extraction using the GIS training algorithm (See the chapter 3 for the detail of the algorithm).

Figure 5.1 depicts F-measure scores of the Co-MEM and MEM for sentence extraction on various summarization sizes. The results show that using a small labeled examples and larger number of unlabeled examples, we can obtain a comparable results to using whole labeled data. It shows that using Co-MEM slightly outperforms using MEM for the whole training data. This was because using Co-MEM we might remove some noisy examples which are not helpful for maximum entropy models.

For the second experiment we collected training data from DUC corpus by performing our decomposition program to obtain sentence extraction corpus and sentence reduction corpus (see Chapter 3). After manually corrected them, we obtained more than 6,000 sentences in which each sentence corresponds to a label that reflects the sentence is important or not. The remaining data are treated as a set of unlabeled data for the co-training algorithm. We used 20,000 unlabeled data for our experiment. The F-measure score is used to compare the performance of the MEM and the Co-MEM for sentence extraction.

Figure 5.2 clearly indicates that the precision and recall of using Co-MEM are better than those of MEM for various summarization size. It also shows that the performance of MEM for sentence extraction is better than that of the leading based method. This was because MEM could combine many other useful features than the position features. These results thus show that using unlabeled data can boost the performance of sentence extraction in comparison with using only training data.

5.5 Conclusions

In this chapter, we propose a novel sentence extraction using maximum entropy model in which a new set of features is used to improve the performance on sentence extraction.

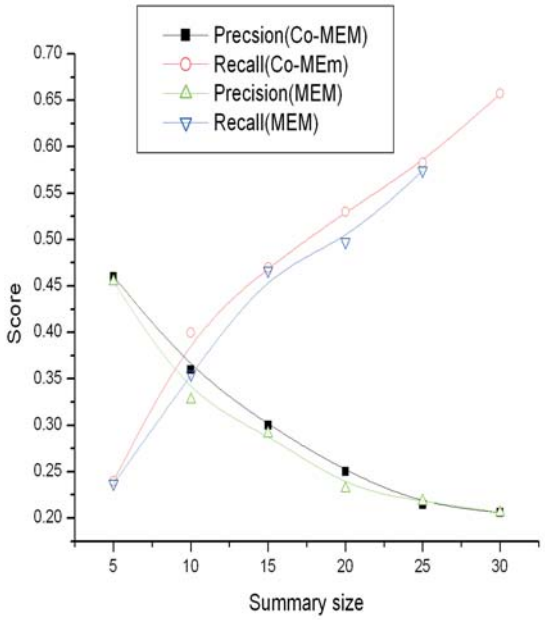

Figure 5.1: The performances of Co-MEM using a part of training data and MEM using whole data on sentence extraction with various size of summary.

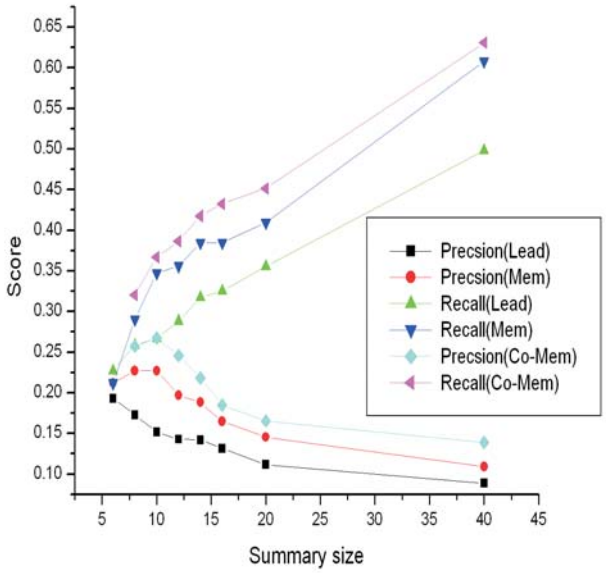

Figure 5.2: The performances of Co-MEM, MEM, and Lead method on sentence extraction with various size of summary.

Experiment results on DUC corpus and cmpl corpus show that the proposed method improved the Lead-based one. We also propose a co-training using maximum entropy model (Co-MEM) which can utilize the unlabeled data to enhance the performance of sentence extraction task. Experiment results show that the Co-MEM model outperformed the conventional maximum entropy model on sentence extraction in term of precision and recall measures. This clearly indicates that unlabeled data is useful for sentence extraction using machine learning and co-training is a suitable technique.

In future work, we will focus on testing the Co-MEM on a larger corpus such as DUC-2002 and DUC-2003. Other week learning techniques than the co-training algorithm are also considered for applying to sentence extraction problem.

Chapter 6

Statistical Machine Learning for Sentence Reduction

6.1 Introduction

Automatic text summarization research has focused on the extraction or identification of important clauses, sentences, and paragraphs in the given texts. The essence of this research is sentence reduction, i.e., reducing long sentences to short sentences so that the gist meaning of the short sentence is the same as that of the original [1].

Several methods have been proposed for sentence reduction in various applications. Grefenstette [15] proposed a method to remove phrases in sentences to produce a telegraphic text that could be used to provide audio scanning services for the blind. Dolan [16] proposed a method to remove clauses in sentences before indexing the document for information retrieval. Those methods remove phrases based on their syntactic categories, without relying on the context of the words, phrases, and sentences around them. Hence, these methods, which do not exploit semantic information, are unsuitable for text summarization.

The most popular methods of sentence reduction for text summarization are corpus based methods. Jing [17] studied a method to remove extraneous phrases from sentences by using multiple sources of knowledge to decide which phrases could be removed. Those multiple sources include syntactic knowledge, contextual information, and statistics computed from a corpus that consists of examples written by human professionals. Jing's method prevents the removal of some important phrases that are relative to the surrounding context and produces a grammatical sentence. However, while this method exploits a simple model for sentence reduction by using statistics computed from a corpus, a better model can be obtained by using a learning approach.

Knight and Marcu [14] proposed a sentence reduction based on corpus using machine learning technique. They discussed a noisy-channel based approach and a decision tree based approach to sentence reduction. Their algorithms provide the good way to scale up the full problem of sentence reduction using available data. However, these algorithms involve the constraint that the word order of a given sentence and its reduced sentence are the same and this prevents an efficient text summarization. Nguyen and Horiguchi [18], [19] presented a new sentence reduction technique based on a decision tree model without that constraint. They also indicated that semantic information is useful for sentence reduction tasks.

The major drawback of previous work on sentence reduction is that those methods are likely to output local optimal results, which may have lower accuracy. This problem is caused by the inherent sentence reduction model; that is, only a single reduced sentence can be obtained. As pointed out by Lin [20], the best sentence reduction output for a single sentence is not approximately best for text summarization. This means that local optimal refer to the best reduced output for a single sentence and the best reduced output for the whole text is global optimal. Therefore, it is very valuable if the sentence reduction task can generate multiple reduced outputs and select the best one using the whole text document. However, such a sentence reduction method has not yet been proposed.

The aim of this chapter is to illustrate the potential of statistical machine learning in enhancing the accuracy of sentence reduction in comparison with previous works. For this purpose the statistical learning models including MEMs and SVMs are used in our sentence reduction algorithms.

Beside, a novel deterministic method for sentence reduction using SVMs and a two-stage method using pairwise coupling [100] are described. Furthermore, to solve the problem of generating multiple best outputs, we propose a probabilistic sentence reduction model, in which a maximum entropy model and a two-stage probabilistic SVMs using pairwise coupling are discussed.

The rest of this chapter will be organized as follows: Section 6.2 presents the previous works on sentence reduction and our deterministic sentence reduction using SVMs and MEMs. We also discuss remained problems of deterministic sentence reduction. Section 6.3 presents a probabilistic sentence reduction using statistical learning to solve this problem. Section 6.4 presents implementation and experimental results; Section 6.5 gives our conclusions and presents some remained problems to be solved in our future work.

6.2 Deterministic Sentence Reduction

Several studies have been proposed for sentence reduction [14], [17], [19]. Among these methods, the corpus based methods described in [14] showed an advantage in comparison with the others. These methods can provide a good way to scale up the full problem of sentence reduction, as vast amounts of train data are widely available in the form of document/abstract pairs. The sentence reduction model as described in [14] could not deal with the changeable order problem and did not use semantic information. To overcome these problems, we extended the sentence compression decision tree model with an assumption that supporting semantic information will generate action decisions with more accuracy. To integrate semantic information into our model, the original sentence was parsed into a syntactic tree using Charniak's parser[91]. Afterward, the syntax tree was enriched with semantic information using the WordNet database[92] and a sub-categorization table that describes the syntactic and semantic role for verbs. The following sections present a sentence reduction technique based on a decision tree model using rich semantic information.

We will now briefly outline the corpus based sentence reduction algorithms and present a novel deterministic sentence reduction method using SVMs. Furthermore, we discuss their disadvantages and point out a solution to overcome these problems.

6.2.1 Problem Description

The problem of sentence reduction is to reduce original sentences into short sentences while retaining the gist meaning. In the corpus based decision tree approach, a given input sentence is parsed into a syntax tree and the syntax tree is then transformed into a small tree to obtain a reduced sentence. The syntax trees follow the phrase structure grammar. Figure 6.1 shows the syntax tree of the sentence "He is a student". Each word is allocated at leaf node and the parent node is the part of speech (Pos) tag for the word. The intermediate nodes in the syntax tree are labelled by grammar symbols. For example, the grammar symbol "VP" has a children node "AUX" which is the POS tag of the word "is".

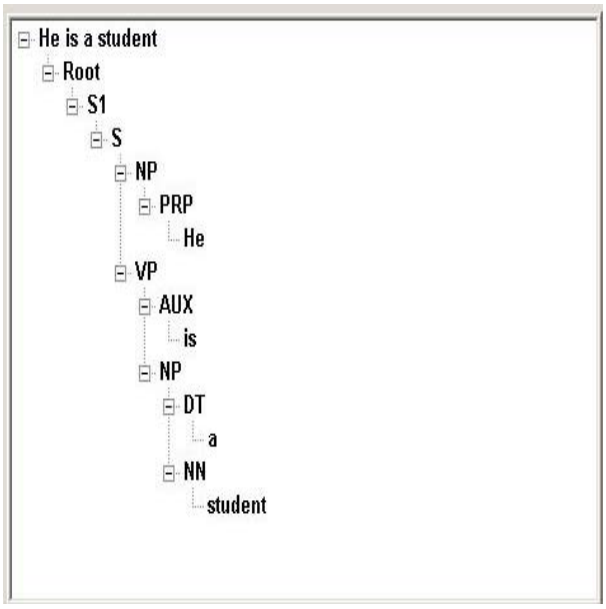

Figure 6.1: The syntax tree of the sentence "He is a student"

Let t and s be syntax trees of the original sentence and a reduced sentence, respectively. The process of transforming syntax tree t to small tree s is called a rewriting process [14]. To transform the syntax tree t to the syntax tree s, some terms and five rewriting actions are defined.

An *Input list* consists of a sequence of words subsumed by the tree t where each word in the Input list is labelled with the name of all syntactic constituents in t. Let CSTACK be a stack that consists of sub trees in order to rewrite a small tree. Let RSTACK be a stack that consists of sub trees, which are removed from the Input list in the rewriting process.

- SHIFT action transfers the first word from the Input list into CSTACK. It is written mathematically and given the label SHIFT.
- REDUCE(lk, X) action pops the lk syntactic trees located at the top of CSTACK and combines them into a new tree, where lk is an integer and X is a grammar symbol.

- DROP X action moves subsequences of words that correspond to syntactic constituents from the Input list to RSTACK. Both REDUCE and DROP actions are used to derive the structure of the syntactic tree of the reduced sentence. They are written as DROPX with X as a grammar symbol.

- ASSIGN TYPE X action changes the label of trees at the top of the CSTACK. These POS tags might be different from the POS tags in the original sentence. These actions are written as ASSIGN TYPE X, in which X represents a POS tag.

- RESTORE lk action takes the lk^{th} element in RSTACK to remove that element into the Input list, where lk is an integer. This action is designed with the assumption that a sub-tree removed from the input list still affects the current decision.

For convenience, let *configuration* be a status of Input list, CSTACK and RSTACK. Let *current context* be the important information of a configuration. The important information are defined as a vector of features using heuristic methods as in [14], [18].

We are now summarizing a sentence reduction algorithm based on a decision tree model. The main idea behind the corpus based decision tree approaches are to automatically learn a set of rules from a corpus, and to apply the rules repeatedly to reduce a sentence. Here, one rule corresponds to the function that maps the current context to a rewriting action. These rules are learned automatically from the corpus of long sentences and their reduced sentences [14], [18]. Algorithm 7 shows that a given input sentence is parsed and each word in the Input list is matched to the corresponding word in the sentence and the sequence of syntactic constituents beginning with each word. An Input list and two stacks for a given input sentence are obtained by line 1 and line 2, respectively. The current context is obtained from line 4. An action and a parameter for that context

Algorithm 7 A sentence reduction algorithm based on decision tree

```
1: The input sentence is parsed into a syntax tree
2: Create an input list and set CSTACK and RSTACK to empty.
   s= Initial configuration
3: while not terminal condition do
4:    context=get-context(s);
5:    action= get-action(context);
6:    parameter=get-parameter(action);
7:    switch (action)
8:    case SHIFT: SHIFT();
9:    case ASSIGN TYPE: ASSIGN-TYPE(parameter);
10:   case REDUCE: REDUCE(parameter);
11:   case DROP: DROP(parameter);
12:   case RESTORE: RESTORE(parameter);
      {Obtain a new configuration s}
13: end while
14: Obtain a reduced tree
15: Generate a reduced sentence from the reduced tree
```

are obtained by using the function *get-action* in line 5 and *get-parameter* in line 6. The functions *get-action* and *get-parameter* are used to get action and parameter information

for performing the action SHIFT, DROP, RESTORE, ASSIGN-TYPE and REDUCE. The algorithm repeats these actions until the terminal condition is satisfied; that is, when Input list is empty and there is only one sub-tree in CSTACK with its root node as the one of the *terminal symbols* (the symbol to recognize it as a root symbol). Summarily, Algorithm 7 is a deterministic sentence reduction version, its behavior is illustrated in the following section.

6.2.2 Example

Figure 6.2 shows an example of applying a sequence of actions to rewrite the input sentence (a, b, c, d, e), where each character is a word. The original tree t is transformed to the Input list, in which the square and circle represent the intermediate node and the leaf node of the syntax tree, respectively. The structure of the Input list, two stacks, and the term of a rewriting process based on the actions mentioned above are illustrated in Figure 6.2. For example, in the first row, DROP H deletes the sub-tree with its root node H in the Input list and stored it in the RSTACK. The Input list is changed to the new status, with the sub-tree with its root node A as shown in the second line. The reduced tree s can be obtained after applying a sequence of actions as follows: DROP H; SHIFT; ASSIGN TYPE K; DROP B; SHIFT; ASSIGN TYPE H; REDUCE 2 F; RESTORE H; SHIFT; ASSIGN TYPE D; REDUCE 2G. In this example, the reduced sentence is (b, e, a). As we can see, the SHIFT action and ASSIGN TYPE K action rewrites the POS tag of the word b in step 2-3; the REDUCE actions modify the skeleton of the tree given as input in steps 7, 8 and 11. DROP actions delete a sub tree from the Input list and store it into RSTACK.

6.2.3 Learning Reduction Rules

As mentioned above, the action for each configuration can be guessed by using a learning rule, which map a context to an action. To obtain such rules, the configuration is represented by a vector of features with a high dimension. After that, we estimate the training examples by using several support vector machines to deal with the multiple classification problem in sentence reduction.

Features

One of the important task in applying SVMs to text summarization is to define features. In this section, we describe features used in our sentence reduction models.

The features are extracted based on the current context. As it can be seen in Figure 6.3, a context includes the status of the Input list and the status of CSTACK and RSTACK. We define a set of features for a current context as follows:

Operation feature

These features reflect the number of trees in CSTACK and RSTACK, and the type of last five actions. We also used the features represents for the information of two stacks, as the information denotes the syntactic category of the root nodes of the partial trees built up to a certain time. We considered the ten last partial trees in CSTACK and RSTACK for obtaining syntactic category of their root nodes, and the Pos tag of a last word in

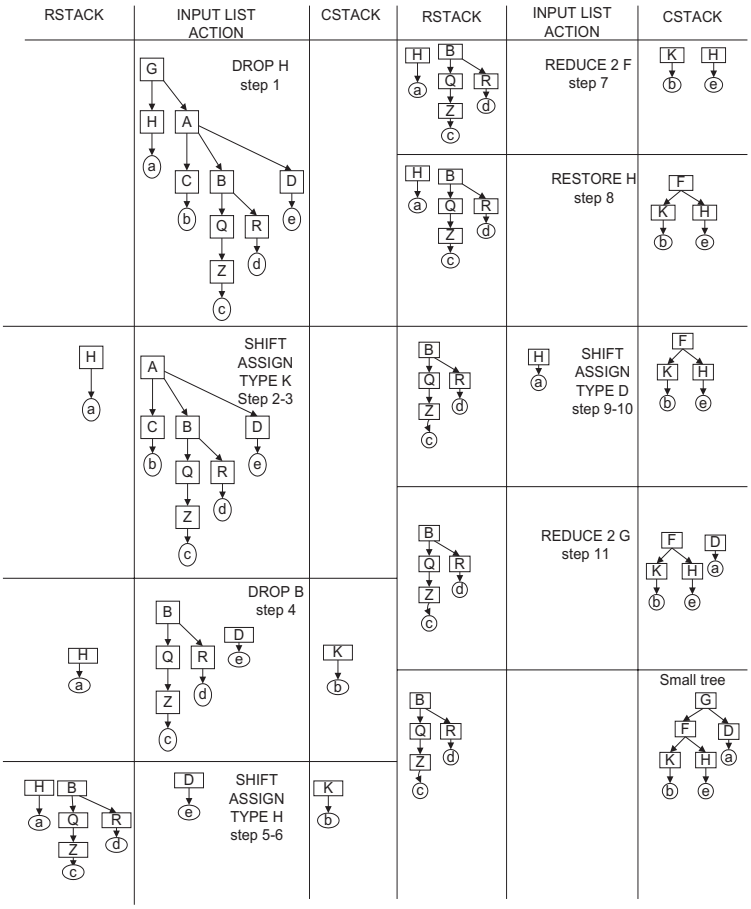

Figure 6.2: An Example of Rewriting Process

CSTACK is also considered as a feature.

Original tree feature

These features denote the syntactic constituents that start with the first unit in the Input list. For example, in Figure 6.3 the syntactic constituents are labels of the current element in the Input list from "NP" to the verb "convince".

Semantic features

The followings are used in our model as semantic information.

- Semantic information about current words within the Input list; these semantic types are obtained by using the named entities such as LOCATION, PERSON, ORGANIZATION and TIME within the input sentence. To define these name entities, we use the method described in [94].
- Semantic information about whether or not a word in the Input list is a *head word*.
- The word relations such as whether or not a word is in a relation with other words in the sub-categorization table. These relations and the sub-categorization table are obtained by using the COMMLEX database [97].

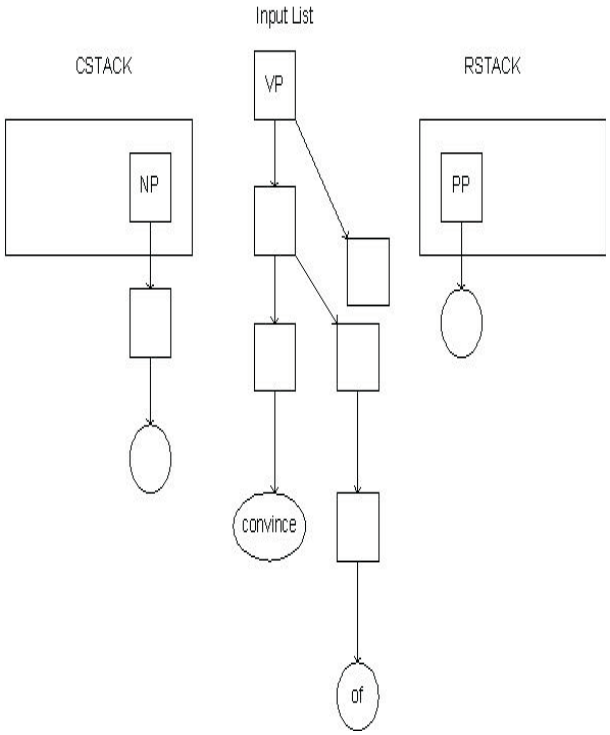

Figure 6.3: Example of Configuration

Using the semantic information, we are able to avoid deleting important segments within the given input sentence. For instance, the *main verb*, the *subject* and the *object* are essential and for the noun phrase, the head noun is essential, but an adjective modifier of the head noun is not. For example, let us consider that the verb "convince" was extracted from the COMLEX database as follows.

convince
NP-PP: PVAL ("of")
NP-TO-INF-OC

This entry indicates that the verb "convince" can be followed by a noun phrase and a prepositional phrase starting with the preposition "of". It can be also be followed by a noun phrase and a to-infinite phrase. This information shows that we cannot delete an "of" prepositional phrase or a to-infinitive that is the part of the verb phrase.

Two-stage SVMs Learning using the Pairwise Coupling

Using these features we can extract training data for SVMs. Here, a sample in our training data consists of a pairs of a feature vectors and an action. The algorithm to extract training data from the training corpus are modified using the algorithm described in our pervious work [18].

Since the original support vector machine (SVM) is a binary classification method while the sentence reduction problem is formulated as multiple classification, we have to find a method of adapting support vector machines to this problem. The multi-class classification problem refers to assigning each observation into one of k classes. As two-

class problems are much easier to solve, many authors proposed to use two-class classifiers for multi-class classification. For multi-class SVMs, one can use the strategies such as *one-vs all*, *pairwise comparison* or *DAG graph* [102]. In this paper, we use the pairwise strategy that constructs a rule for discriminating pairs of classes and then selecting the class with the most winning among two class decisions.

Since the number of class labels in our training data can be large, the cost of using the pairwise method could be high. Fortunately, for the sentence reduction problem, the class labels belong to one of five groups: SHIFT, REDUCE, DROP, ASSIGN TYPE and RESTORE. So, we could boost the training time and the reduction performance by using a simple method described below.

Suppose that the examples were divided into five groups $m_1, m_2, ..., m_5$. Let SVMC be multi-class SVMs using pairwise coupling method and let SVMC-i be multi-class SVMs for a group m_i. The idea of our two-stage strategy for SVMs multiple classification is that a given example can be recognized by using two stages separately. In the first stage, we use one SVMC classifier to recognize to which group a given context e should belong. Assume that e belongs to the group m_i. In the second stage, each classier SVMC-i is used to recognize a specific action for e. The five other classifiers SVMC-1, SVMC-2,..., SVMC-5 are trained by using those examples which have actions belonging to SHIFT, REDUCE, DROP, ASSIGN TYPE and RESTORE.

Table 6.1 shows the distribution of examples on five data groups. The SVMC-1, SVMC-2,..., and SVMC-5 are trained using SHIFT-GROUP, REDUCE-GROUP,..., and RESTORE-GROUP, respectively.

Table 6.1: Distribution of example data on five data groups

Name	Number of examples
SHIFT-GROUP	13,363
REDUCE-GROUP	11,406
DROP-GROUP	4,216
ASSIGN-GROUP	13,363
RESTORE-GROUP	2,004
TOTAL	44,352

6.2.4 Disadvantage of Deterministic Sentence Reductions

The main idea behind Algorithm 7 is that it uses a rule in the current context of the initial configuration to select a distinct action in order to rewrite an input sentence into a reduced sentence. After that, the current context is changed to a new context and the rewriting process is repeated for selecting an action that corresponds to the new context. The rewriting process is finished when it meets a termination condition. Clearly, Algorithm 7 only generates a single reduced sentence and it does not approximate the best text summarization results. It is because that if early steps of Algorithm 7 fail to select the best actions, then the possibility of obtaining a wrong reduced output becomes high.

One way to solve this problem is to select multiple actions that corresponds to the context at each step in the process of rewriting. However, the question that emerges here is how to determine criteria to use in selecting multiple actions for a context. If this problem can be solved, then multiple best reduced outputs can be obtained for each input sentence and the best one will be selected by using the whole text document.

We propose a model for selecting multiple actions for a context in sentence reduction as a probabilistic sentence reduction and present a variant of a probabilistic sentence reduction in the next section.

6.3 Probabilistic Sentence Reduction Using Statistical Learning

6.3.1 The Probabilistic Models

Let A be a set of k actions

$$A = \{a_1, a_2...a_i, ..., a_k\} \tag{6.1}$$

and C be a set of n contexts

$$C = \{c_1, c_2...c_i, ..., c_n\} \tag{6.2}$$

A probabilistic model α for sentence reduction will select an action $a \in A$ for the context c with probability $p^\alpha(a|c)$. The $p^\alpha(a|c)$ can be used to score action a among possible actions A depending on the context c that is available at the time of decision. There must be several methods to estimate such scores. We called these probabilistic sentence reduction methods. The conditional probability $p^\alpha(a|c)$ are estimated using a variant of probabilistic support vector machine, which is described in following sections.

6.3.2 Maximum Entropy Learning

We used the same feature set as mention above for maximum entropy estimation. The contextual predicates of an action a are used to encode the derivation in the corpus of long sentences and reduced sentences. The algorithm to extract training events from the training corpus was modified using the algorithm described in our pervious work [18]. Here is the summary of the extraction training events algorithm. First, we parse all long sentences and their corresponding reduced sentences into syntactic trees [91]. We then use the larger tree and small tree to find the sequence of actions in order to rewrite the larger tree into the small tree. At each action, the training event is extracted based on the current context information, namely the Input list, and two stacks. The training events are already defined as (a, c) in section 3.2.1. Here, each (a, c) represents an action a and is encoded as $(a, cp_1, cp_2, ...cp_l)$, where $cp_1...cp_l$ are contextual predicates such that $cp_i(c)$ =true, for $1 \leq i \leq l$, and c is the context in which action a occurred. After obtaining training events from the manual corpus of long sentences and reduced sentence, feature selection and parameter estimation are used to obtain the maximum entropy model for sentence reduction. Feature selection task and parameter estimation are described in the following sections.

6.3.3 Features Selection and Parameter Estimation

Let F be a set of features that occurred on the training data. Feature selection is the process of choosing a useful subset of features F from a set of all possible features to be used in the maximum entropy. There are several feature selection methods [106]. Among them, the simplest uses a count cutoff technique [103]. This method is useful for some tasks such as part of speech tagging [103] and NP chunking [105]. Choosing the best selection feature method for a sentence reduction task is out of the context of this paper. We will address this problem in future articles. In our framework, we use the count cutoff technique which can described as follows:

$$F = \{f | f(a,c) = \begin{cases} 1 \text{ if } \quad cp(c) = \text{true } \&\& \ a = a' \\ 0 \text{ otherwise} \end{cases} \\ \text{where } \quad cp \in CP \quad \text{and} \quad a' \in A \ , \sum_{a,c} f(a,c) \geq 4\}$$

The training set data is used to estimate the parameters of a probability model p.. Each feature f_j corresponds to parameter α_j. The parameters $\{\alpha_1, \alpha_2, ..., \alpha_k\}$ are found by using the Limited Memory BFGS algorithm (L-BFGS) [61], which is faster than GIS and IIS algorithm [62], to estimate the maximum entropy model.

6.3.4 Probabilistic SVM Models using the Pairwise Coupling

For convenience, we denote $u_{ij} = p(a = a_i | a = a_i \vee a_j, c)$. Given a context c and an action a, we assume that the estimated pairwise class probabilities r_{ij} of u_{ij} are available. Here r_{ij} can be estimated by some binary classifiers. For instance, we could estimate r_{ij} by using the SVM binary posterior probabilities as described in [99]. Then, the goal is to estimate $\{p_i\}_{i=1}^k$, where $p_i = p(a = a_i | c)$, $i = 1, 2, ..., k$. For this propose, a simple estimate of these probabilities can be derived using the following voting method:

$$p_i = 2 \sum_{j:j \neq i} I_{\{r_{ij} > r_{ji}\}} / k(k-1)$$

where I is an indicator function and $k(k-1)$ is the number of pairwise classes. However, this model is too simple; we can obtain a better one with the following method.

Assume that u_{ij} are pairwise probabilities of the initial model subject to the condition that $u_{ij} = p_i/(p_i + p_j)$. In [100], the authors proposed to minimize the Kullback-Leibler (KL) distance between the r_{ij} and u_{ij}

$$l(p) = \sum_{i \neq j} n_{ij} r_{ij} log \frac{r_{ij}}{u_{ij}} \tag{6.3}$$

where r_{ij} and u_{ij} are the probabilities of a pairwise a_i and a_j in the estimated model and in our model, respectively, and n_{ij} is the number of training data in which the classes are a_i or a_j.

To find the minimize of equation (6.3), they first calculate

$$\frac{\partial l(p)}{\partial p_i} = \sum_{i \neq j} n_{ij} (-\frac{r_{ij}}{p_i} + \frac{1}{p_i + p_j}).$$

Thus, letting $\Delta l(p) = 0$, they proposed to find a point satisfying

$$\sum_{j:j\neq i} n_{ij}u_{ij} = \sum_{j:j\neq i} n_{ij}r_{ij}, \qquad \sum_{i=1}^{k} p_i = 1, \quad p_i > 0, i = 1, ..., k.$$

Such a point can be obtained by using Algorithm 8. Algorithm 8 converges to these

Algorithm 8 An estimation algorithm

1: Start with some initial $p_i > 0$ and u_{ij}=$p_i/(p_i + p_j)$
2: **while** not terminal condition **do**
3: $\alpha = \frac{\sum_{j:j\neq i} n_{ij}r_{ij}}{\sum_{j:j\neq i} n_{ij}u_{ij}}$
4: $u_{ij} = \frac{\alpha u_{ij}}{\alpha u_{ij}+u_{ji}}$, $u_{ji} = 1 - u_{ij}$
5: $p_i = \alpha p_i$
6: **end while**

probabilities of actions and context that we need. The detail of Algorithm 8 was described elsewhere in [100]. It is applied to obtain a probabilistic SVM model for sentence reduction using a simple method as follows. Assume that our class labels belong to l groups:

$$M = \{m_1, m_2...m_i, ..., m_l\}$$

where l is a number of groups and m_i is a group e.g., SHIFT, REDUCE ,..., ASSIGN TYPE. Then the probability $p(a|c)$ of an action a for a given context c can be estimated as follows.

$$p(a|c) = p(m_i|c) \times p(a|c, m_i) \tag{6.4}$$

where m_i is a group and $a \in m_i$. In this chapter, $p(m_i|c)$ and $p(a|c, m_i)$ are estimated by using Algorithm 8.

6.3.5 Probabilistic sentence reduction algorithm

After obtaining a probabilistic model p, we then use this model to define function score, by which the search procedure ranks the derivation of incomplete and complete reduced sentences. Let $d(s) = \{a_1, a_2, ...a_d\}$ be the derivation of a small tree s, where each action a_i belongs to a set of possible actions. The score of s is the product of the conditional probabilities of the individual actions in its derivation.

$$Score(s) = \prod_{a_i \in d(s)} p(a_i|c_i) \tag{6.5}$$

where c_i is the context in which a_i was decided. The search heuristic tries to find the best reduced tree s^* as follows;

$$s^* = \underbrace{\text{argmax}}_{s\in \text{tree}(t)} \quad Score(s) \tag{6.6}$$

where $tree(t)$ are all the complete reduced trees from the tree t of the given long sentence. Assume that for each configuration the actions $\{a_1, a_2, ...a_n\}$ are sorted in decreasing order according to $p(a_i|c_i)$, in which c_i is the context of that configuration. Algorithm 9

shows a probabilistic sentence reduction using the top K-BFS search algorithm. The idea of this algorithm is close to the breadth-first search algorithm (BFS) but it is extended by the following strategy. The heuristic algorithm uses a breadth-first search which does not expand the entire frontier, but instead expands at most the top K scoring incomplete configurations in the frontier; it is terminated when it finds M completed reduced sentences, or when all hypotheses have been exhausted. A configuration is completed if and only if the Input list is empty and there is one tree in the CSTACK. Note that the function *get-context(h_i, j)* obtains the current context of the j^{th} configuration in h_i. This function is the same as the get-context in Algorithm 7. The function *Insert(s,h)* ensures that the heap h is sorted according to the score of each element in h. Essentially, in implementation we can use a dictionary of contexts and actions observed from the training data in order to reduce the number of actions to explore for a current context.

Algorithm 9 A probabilistic sentence reduction algorithm

M=20
K=20
Q=0.95
CL={Empty}
$i = 0$
h_0={ Initial configuration}
while $|CL| < M$ **do**
  if h_i is empty **then**
    break;
  end if
  u =min($|h_i|, K$)
  for $j = 1$ to u **do**
    c=get-context(h_i, j)
    Select m so that $\sum_{i=1}^{m} p(a_i|c) < Q$ is maximal
    for l=1 to m **do**
      parameter=get-parameter(a_l);
      Obtain a new configuration s by performing action a_l with parameter
      if Complete(s) **then**
        Insert(s, CL)
      else
        Insert(s, h_{i+1})
      end if
    end for
  end for
  $i = i + 1$
end while

6.4 Evaluation

We implemented the sentence reduction based on support vector machines in the Pentium IV, 2.6GHz on Windows XP environment and executed the experiments.

To evaluate sentence reduction algorithms, we randomly selected 32 pairs of sentences from our parallel corpus, which is refereed to as the test corpus. We used the same corpus as described in [14], which includes 1,035 sentence pairs for training our model and decision trees. The training corpus of 1,035 sentences extracted 44,352 examples, in which each training example is associated to an action. The SVM tool, LibSVM tool [101] is applied to train our model. The training examples were divided into SHIFT, REDUCE, DROP, RESTORE, and ASSIGN groups. To train our support vector model in each group, we used the pairwise method with the polynomial kernel function, in which the parameter p in (3.17) and the constant C_0 in equation (3.15) are 2 and 0.0001, respectively. The L-BFGS algorithm [61] are used to train the maximum entropy model for sentence reduction.

Figure 6.4 shows sentence reduction accuracy as a function of the number of rounds (steps) of the L-BFGS algorithm on reduction training data. A new model is obtained after one round of the L-BFGS algorithm. Using this model we are able to define the accuracy of predicting a given event by comparison with its correspond action in the training data. The accuracy is computed as the percent of the total correct actions on the training events. It seems that the training algorithms using L-BFGS converge and are sufficiently fast. After 900 rounds of L-BFGS algorithm, we obtained an accuracy of 92.510%. We applied the L-BFGS algorithm until convergence and achieved an accuracy of 94.874% after 8,000 rounds of the L-BFGS algorithm. With the results shown in Figure 6.4, we can say that the L-BFGS algorithm is suitable to generate a maximum entropy model for sentence reduction.

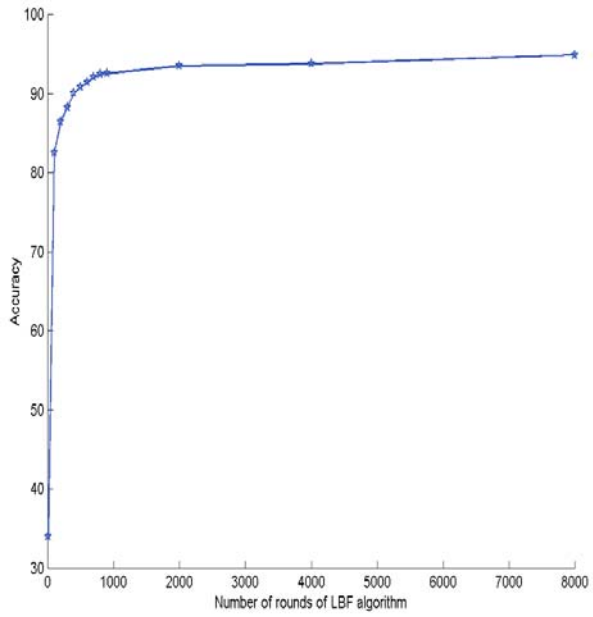

Figure 6.4: The relation of accuracy and number of rounds of the L-BFGS algorithm on reduction training data

The algorithms [14] and [18] served as the baseline1 and the baseline2 to compare with the proposed algorithms. The deterministic sentence reduction using SVM and the

probabilistic sentence reduction were named as SVM-D and SVMP, respectively. For convenience, the 20 top reduced outputs using SVMP is called SVMP-20. We used the same evaluation method as described in [14] to compare the proposed methods with previous methods. For this experiment, we presented each original sentence in the test corpus to three judges who are specialists in English, together with three sentence reductions: the human-generated reduction sentence, the outputs of the proposed algorithms, and the output of the baseline algorithms.

The judges were told that all outputs were generated automatically. The order of the outputs was scrambled randomly across test cases. The judges participated in two experiments. In the first, they were asked to determine on a scale from 1 to 10 how well the systems did with respect to selecting the most important words in the original sentence. In the second, they were asked to determine the grammatical criteria of reduced sentences.

Table 6.2 shows the results of English language sentence reduction using a support vector machine in comparison with the baseline methods and with human reduction. Table 6.2 shows *compression rates*, and *mean* and *variance* results across all judges, for each algorithm. The results show that the length of the reduced sentences using decision trees is shorter than using SVMs, and indicate that our new methods outperform the baseline algorithms in grammatical and importance criteria. We used the t-test at the $p < 0.05$(95% confident) of significant difference for each judger and average scores of them. Table 6.2 shows that human reduction is statistical significant better than all sentence reduction methods and the proposed methods are statistical significant better than the previous methods at the importance criterion but not gramticality.

Table 6.2: Experiment results with Test Corpus

Method	Compression	Grammaticality	Importance
Baseline1	57.19%	8.60 ± 2.8	7.18 ± 1.92
Baseline2	57.15%	8.60 ± 2.1	7.42 ± 1.90
MEM	58.25%	8.70 ± 1.3	7.50 ± 1.50
SVM-D	57.65%	8.76 ± 1.2	7.53 ± 1.53
SVMP-20	59.45%	8.82 ± 1.0	8.13 ± 0.52
MEM-20	60.24%	8.84 ± 1.0	8.14 ± 0.54
Human	64.00%	9.05 ± 0.3	8.50 ± 0.80

Table 6.2 also shows that the first 10 reduced sentences produced by MEM-20 and SVMP-20 (the SVM probabilistic model) obtained the highest accuracies.

We also compared the computational times of sentence reduction using support vector machine with that in the previous methods. The results reported in Table 6.3 are the computational times for performing reduction the test set where the average length of sentences was 21 words for three methods, baselines (sentence reduction based decision tree), SVM-D and SVMP-20. Table 6.3 shows that the computational times of SVM-D and SVMP-20 are slower than that of the baseline.

We also investigated how sensitive the proposed algorithms are with respect to the training data by carrying out the same experiment on sentences of different genres. We created the test corpus by selecting sentences from the web-site of the Benton Foundation

Table 6.3: Computation times of performing reductions on test-set. Average sentence length was 21 words.

Method	Time (sec)
Decision-Tree	138.25
MEM	11.46
SVM-D	212.46
MEM-20	1050.4
SVMP-20	4020.25

(http://www.benton.org). The long sentences in each news article were selected as the most relevant sentences to the summary of the news. We obtained 32 summary sentences among 32 news in the web-site and selected 32 long sentences such that each sentences corresponds to a summary sentence. In addition, we obtained 32 headlines for each item. The 32 sentences are used as a second test for our methods.

As shown in Table 6.2, the outputs of the twenty top reduced sentences using SVMP-20 and MEM-20 yield accuracies higher than previous work. However, it is valuable if a best reduced sentence among these outputs for the whole document can be selected automatically. For this purpose we use a simple ranking criterion: the more the words in the reduced sentence overlap with the words in the headline, the more the important sentence is. We call a sentence satisfying this criteria a *relevant candidate*.

For a given sentence, we used a simple method, namely SVMP-R to obtain a reduced sentence by selecting a *relevant candidate* among the twenty top reduced outputs using SVMP-20. First, a set of most relevant candidates are obtained from the twenty top reduced candidates. After that, the reduced output is given by selecting the one with a highest score computed by SVMP.

Table 6.4 depicts the experiment results for the baseline methods, SVM-D, SVMP-R, and SVMP-20. The results show that, when applied to sentence of a different genre, the performance of SVMP-20 degrades smoothly, while the performance of the deterministic sentence reduction methods (the baselines and SVM deterministic) drop sharply. This indicates that the probabilistic sentence reduction using support vector machine is more stable than the deterministic one.

Table 6.4 shows that the performance of SVMP-20 is also close to the human reduction outputs and better than previous work. In addition, the performance of SVMP-R outperforms the deterministic sentence reduction algorithms and the differences between SVMP-R's results and those for SVMP-20 are small. This indicates that we can obtain reduced sentences which are relevant to the headline, while ensuring the grammatical and the importance criteria compared to the original sentences.

Currently, our method works well in the sentence reduction task, but the larger number of labeled examples is still required. In future work, we will investigate a method which can utilize unlabeled examples to enhance the performance of sentence reduction. Use of a co-training method therefore with shows promise.

Figure 6.5 shows some examples of SVM reduction against reduction by the decision-tree method. In sentence 1, the reduction based on the SVM seems better than decision-tree reduction, and the results of SVM and decision-tree reduction are the same in sen-

Table 6.4: Experiment results with Benton Corpus

Method	Compression	Grammaticality	Importance
Baseline1	54.14%	7.61 ± 2.10	6.74 ± 1.92
Baseline2	53.13%	7.72 ± 1.60	7.12 ± 1.90
SVM-D	56.64%	7.86 ± 1.20	7.23 ± 1.53
SVMP-R	58.50%	8.35 ± 0.90	7.64 ± 0.65
SVMP-20	59.65%	8.70 ± 0.95	8.01 ± 0.61
Human	64.00%	9.01 ± 0.25	8.40 ± 0.60

tences 2 and 3.

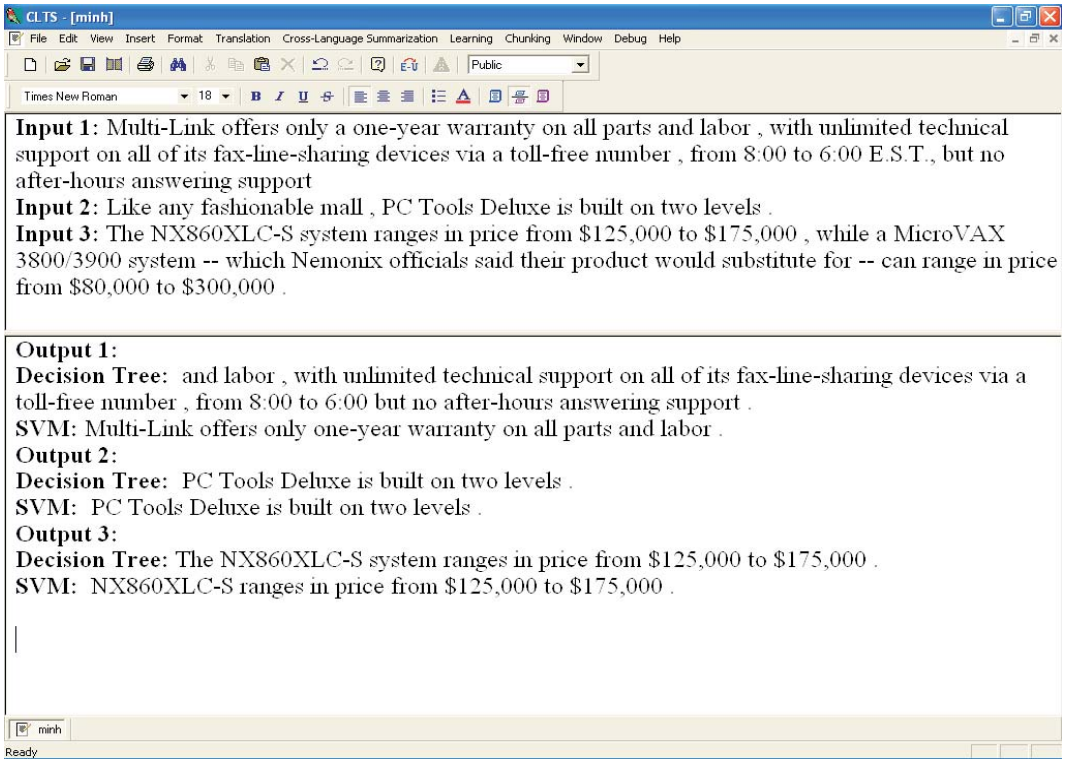

Figure 6.5: Examples of sentence reduction using statistical machine learning and decision tree

To see the impression on the reduction results of multiple candidates and a single candidate, we show an example of 10 reduction outputs obtained by SVM-20 and a single reduction obtained by SVM as shown in Figure 6.6. This example shows that Output6 is better than the output of SVM.

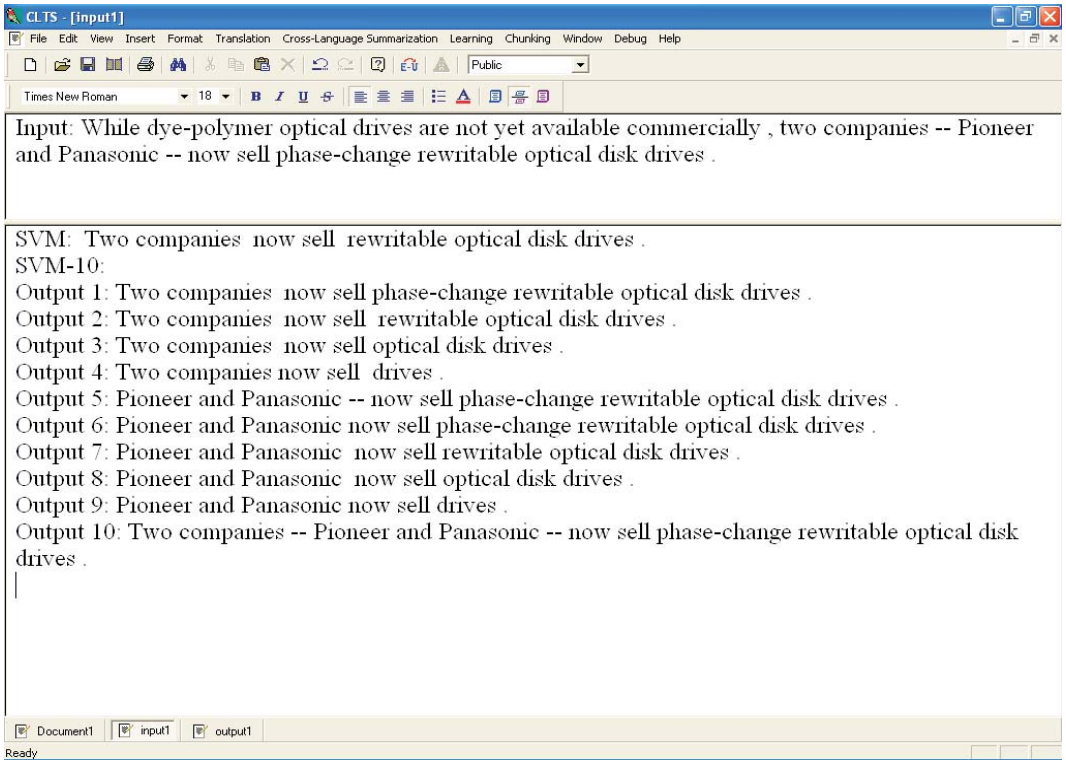

Figure 6.6: Examples of multiple sentence reduction output using statistical machine learning

6.5 Conclusions

We have presented a new sentence reduction approach that enables a long sentence to be rewritten into reduced sentences based on statistical machine learning in which SVM and MEM are used. The reduction sentence using SVM and MEM achieves better performance when compared with earlier methods. The proposed reduction approach can generate multiple best outputs. This enhances the performance of text summarization since it can use the whole text document to select the best reduced sentence among multiple reduction outputs. Experiments showed that the running time required to reducing sentences is reasonable and the top 20 reduced sentences returned by the reduction process might yield accuracies dramatically higher than previous work. In addition, using a simple ranking method with a global information of full text document to the top 20 reduced sentences showed an encouraging results. We suspect that a good selection method for the output of our probabilistic sentence reduction might achieve a better result.

Work on integrating the proposed sentence reduction with a text summarization system is underway.

Chapter 7

Machine Translation in Cross Language Text Summarization

This chapter introduces an example based machine translation method with the use of shallow information obtained from a chunking process. The proposed translation method can be applied for both translation and reduction in a CLTS system. We call this adaptive translation in CLTS, or chunking based example based machine translation(CEBMT).

7.1 Introduction

As described in chapter 2, the problem of cross language text summarization is how to combine a translation engine with a mono-language summarization system. Meanwhile, the translation engine is designed to translate whole sentences, not phrases. It does not perform as well when the input is a list of separate phrases, while summary outputs in a summarization system can often be in the form of a list of phrases. Our main idea here is with developing a translation engine which can be applicable applied for both the whole sentences and a list of sperate phrases. We also investigate the use of this method for reducing an input sentence that does not need any parsers. For this reason, we focus on using example based machine translation method in CLTS.

For convenience, let us summary the idea of example based machine translation (EBMT) as follow. EBMT originally proposed by Nagao [107], is one of the main approaches of corpus-based machine translation. Following Nagao's original proposal, several methods based on EBMT were presented [108], [109], [110], [111]. The excellent review paper of EBMT [112] described the main idea behind EBMT as follows. A given input sentence in a source language is compared with the example translations in the given bilingual parallel text to find the closet matching examples so that they can be used in the translation of that input sentence. After finding the closest matching for the sentence in the source language, parts of the corresponding sentence in the target language are constructed using structural equivalences and deviances in those matches.

One of the approaches which applied successful to translation from English to Turkish is the translation template learning method (TTL)[113][23]. This algorithm relies on the technique using the similarity and difference between two translation examples in the bilingual corpus to build template rules for translation. The advantage of this method is that it only uses the morphological parsing in the source sentence and the target sentence for both learning phase and translation phase in the machine translation system. In the

learning phase, it can generate template rules automatically in a simple way. In the translation phase, it shows a good translation result using those template rules learned from a small bilingual corpus as reported in[23].

In this paper, we focus on investigating the use of translation template learning method to apply for both translation and sentence reduction in CLTS system. We also study the application of this method to the problem of sentence reduction. Intuitively, when considering long sentences as a source language and reduced sentences as a target language, the problem of sentence reduction is equivalent to the translation problem.

Although the translation template system is suitable as well to machine translation, there are some drawbacks as follows.

In the learning phase, with the lack of linguistic knowledge the amount of template rules using translation template learning is large and some of them causes the translation wrong. It is clear that linguistic information is useful for translation. In addition, unreliable rules may reduce the performance of translation in both accuracy and computational times. Incorporating linguistic knowledge into template rules is therefore an expected approach. However, the problem here is how to obtain linguistic knowledge and how to incorporate them into translation template learning. The recent study using named entity information has been shown an improvement of the translation results [114]. But this work is only used to encode examples for the source language and has not applied to TTL method.

On the other hand, shallow parsing has been applied successfully to various nature language processing applications because of their accuracies as well as the easy implementing on other language than English. Many applications on natural language processing has been used shallow parsing as the way to obtain linguistic information of language. In this paper, we propose a novel translation template learning using shallow parsing to incorporate linguistic information to template rules.

In the translation phase, the advantage of this method is that it does not need any complex parsing such as syntactic parsing or semantic parsing and overcome the imperfectness of the rule-based machine translation. The disadvantages of the method are that a lot of templates can be matched with an input sentence and some of them cause the translation results are not confident. To overcome this problem, Öz [115] presents a method which allows sorting template rules according to their confident factors. The translation results are sorted using their scores through the value of confident factors. However, this method needs to evaluate all matching rules for each input sentence to obtain the output results, while much of them are redundant rules. The exponential calculation problem will arise when an input sentence is long and the number of template rules is large. Following that point, we present a novel method based on a HMM model that uses constraints for set of matching rules with each input sentence. Thus, the translation results of an input sentence are obtained by finding a set of template rules that is most likely with our HMM model.

The rest of this chapter is organized as follows. Section 7.2 introduces an architecture of our chunking-based example-based machine translation system. Section 7.3 presents a learning phase in the architecture by describing an algorithm of shallow translation template learning. Section 7.4 gives a template translation learning using shallow parsing. Section 7.5 presents a translation phase that describes the template translation, the shallow template translation, and the combination of them with HMM model. Section 7.6 introduces the application of CEBMT to sentence reduction. Section 7.7 gives some

experimental results and section 7.8 shows our conclusions.

7.2 Translation Template Learning

7.2.1 Template Rules

Let SL and TL be the source language and the target language. Let $S_1S_2...S_n \leftrightarrow T_1T_2...T_k$ be a template rules, in which S_i is a sequence of words or a variables in SL and T_i is a sequence of words, so called a constant element, or a variables in TL. In addition, each variable in the left side is aligned with a variable in the right side. A variable in the left side and a variable in the right side of a template rule can be received a phrase or a word in SL and TL, respectively. A lexical rule is defined as a template rule that has no variable inside.

An example of a template rule is shown in Figure 7.1.

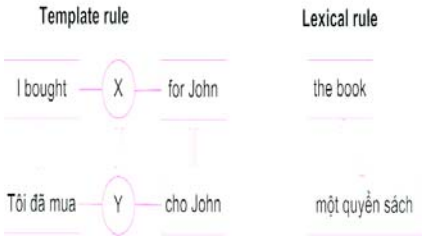

Figure 7.1: An example of template rules

7.2.2 Learning Template Rules

This section describes the original translation template learning[24]. Suppose that two examples E_a and E_b taken from a bilingual parallel corpus. In which, a translation example $E_a^1 \leftrightarrow E_a^2$ is composed of a pair of sentences, E_a^1 and E_a^2, which are translations of each other in English and Vietnamese.

Let a similarity between two sentences of a language be a non-empty sequence of common items (root words or morphemes) in both sentences. A difference between two sentences of a language is a pair of two sequences (D_1, D_2) where D_1 is a sub-sequence of the first sentence, D_2 is sub sequence of the second sentence, and D_1 and D_2 do not contain any common item.

Given two translation examples (E_a, E_b), we first find similarities between the constituents of E_a and E_b and formulated it as a match sequence $M_{a,b}$ in the following form.

$$M_{a,b} = S_0^1 D_0^1 ... D_{n-1}^1 S_n^1 \leftrightarrow S_0^2 D_0^2 S_1^2 ... D_{m-1}^2 S_m^2 \qquad for \qquad n, m \geq 1 \tag{7.1}$$

where S_k^1 represents a similarity (a sequence of common items) between E_a^1 and E_b^1. Similarly, $D_k^1 : (D_{k,a}^1, D_{k,b}^1)$ represents a difference between E_a^1 and E_b^1, where $D_{k,a}^1, D_{k,b}^1$ are non-empty differing items between two similar constituents S_k^1, S_{k+1}^1. For example,

Figure 7.2 shows two translation examples and the values of $M_{a,b}$ (e.g S_0^1 = "I bought the", $D_{0,a}^1$= "book")

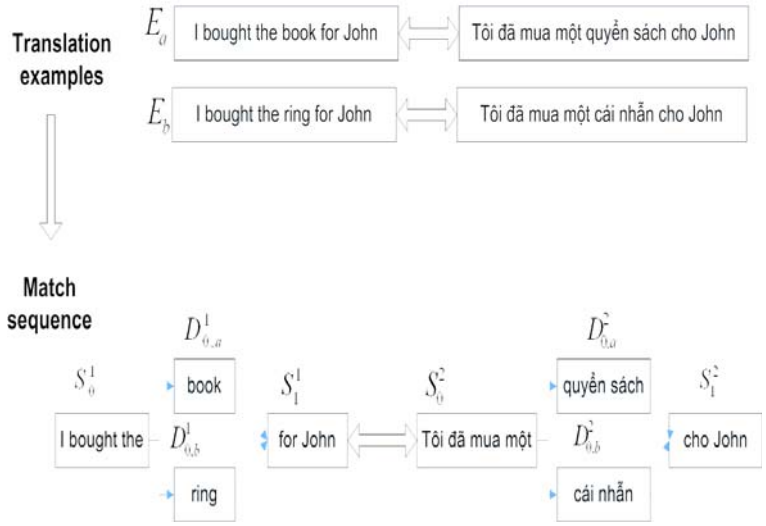

Figure 7.2: Translation examples-Match sequence

After a match sequence is found for two translation examples, we used the two different learning heuristics to infer translation templates from that match sequence [24]. The two heuristic algorithms try to locate corresponding differences or similarities in the match sequence. The first heuristics, the similarity translation template learning (STTL) tries to locate all corresponding differences and generate a new translation template rule by replacing all differences with variables. The second heuristics can infer translation templates by replacing similarities with variables, if it is able to locate corresponding similarities in the match sequence. These translation templates are called difference translation templates learning (DTTL). The STTL and DTTL are combined as the template learning algorithm. From the corpus, the TTL algorithm tries to infer translation templates using the two algorithms above. After all translation template rules are learned, they are sorted according to their specificities based on the simple criterion as follows. Given two templates, one that has a higher number of terminals is more specific than the other.

7.2.3 Translation using template rules

The original translation method [113] can be summarized as follows: Given an input sentence $e_1e_2...e_m$ (e_i is a word) and a set of template rules $r_1, r_2, ..., r_d$, find all translation results for each rule r_i(i=1, d).

Assuming that the rule r_i is defined as $S_1S_2..S_n \leftrightarrow T_1T_2...T_k$, the original method tries to find all possible ways to replace the variables with phrases in SL so that the input sentence $e_1e_2...e_m$ can be produced from this rule. Next, it finds each corresponding phrase in TL within the set of lexical rules with a phrase in SL, in order to transform the input sentence into the target language. It is obvious that the process above relied on a recursive algorithm, in which each variable S_j in a template rule r_i may consist of several lexical rules satisfying the left side of lexical rules are matched with a substring in the input

sentence. Therefore, with an input sentence we may have several output translations. The example following illustrates the behavior of the original template translation.

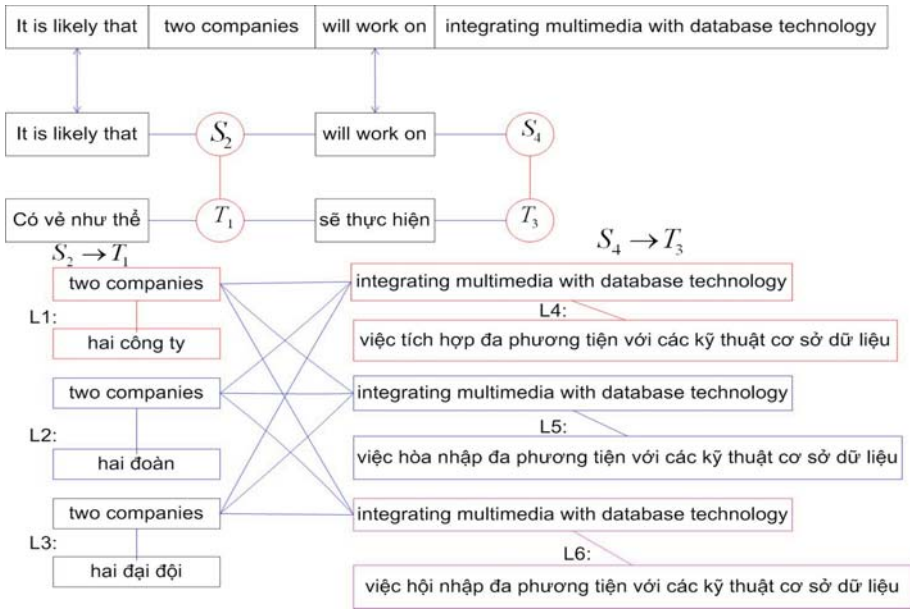

Figure 7.3: An example of translation using template rules

Figure 7.3 shows the number of translation outputs for the given input sentence "It is likely that two companies will work on integrating multimedia with database technology"

Two phrases "It is likely that" and "will work on" within the input sentence are matched with the template rule. The reduction algorithm then tries to find all possible choices to replace variables S_2 and S_4 using lexical rules. For variables S_2 and S_4 it finds all lexical rules whose left side is matched with "two companies" and "integrating multimedia with database technology", respectively. Figure 7.3 shows three choices for S_2 and S_4, from which we have six translation results. Intuitively, the best translation output is the sequence of lexical rules L_1 and L_4.

There are two obstacles when using the original template reduction:

- How to determine the best translation outputs when using template translation.
- Suppose that a template rule has t variables and each variable has l matched lexical rules, so we have l^t choices for translation. How can we deal with this exponential calculation?

To solve the problems, we develop a HMM-based method described in next sections.

7.3 Statistical Learning for Translation Template Rules

7.3.1 Template Translation Using HMM modelling

The main idea is that instead of considering all lexical rules, we use a dynamic programming algorithm to find a likely sequence of lexical rules for a given input sentence. Suppose

Table 7.1: Probability table

	L_4	L_5	L_6
L_1	0.5	0.3	0.2
L_2	0.4	0.3	0.3
L_3	0.4	0.3	0.3

that the probabilities $P(l_i|l_j)$ between two lexical rules l_i and l_j are given in Table 7.1. From this table, the likely sequence of lexical rules is (l_1, l_4), and these lexical rules lead us obtain the best translation output.

Given an input sentence $e_1e_2...e_m$ (e_i is a token) and a set of template rules $r_1, r_2, ..., r_d$. Our problem here is to find a sequence of lexical rules that their translation results most explain the given sentence. This problem is equivalent to that of finding all likely translation results for each rule $r_i(i = 1, ..., d)$.

For the rule r_i: $S_1S_2...S_j...S_N \leftrightarrow T_1T_2...T_j...T_K$ in the above example, each constant S_j $(j = 1, ..., N)$ can be associated with a phrase in the right side of the rule r_i, and each variable S_j $(j = 1, ..., N)$ within the rule can be associated with set of lexical rules whose left side is a substring that starts from a possible position within the input sentence.

Consider a lexical rule as a hidden state and a substring in the input sentence as an observed symbol. The problem of reduction is then equivalent to finding a lexical rule for each variable. To find the most likely sequence of lexical rules, we must find a sequence of lexical rules that maximized the probability $P(r_i|e_1e_2...e_m)$. Since $r_i : S_1S_2, ...S_N \leftrightarrow T_1T_2...T_K$ and the map between $S_1S_2...S_N$ and r_i is one by one, we obtain

$$P(r_i|e_1, e_2, ..., e_m) = P(S_1S_2...S_N|e_1e_2...e_m) \tag{7.2}$$

Applying Bayes rule, we have

$$P(S_1S_2...S_N|e_1, e_2, ..., e_m) = \frac{P(e_1e_2...e_m|S_1S_2...S_N)}{P(e_1...e_m)} \times P(S_1S_2...S_N) \tag{7.3}$$

Since $e_1e_2...e_m$ is a sequence of input words, the probability $P(e_1e_2...e_m)$ is given, we need to maximize the following

$$P(e_1e_2...e_m|S_1S_2...S_N) \times P(S_1S_2...S_N) \tag{7.4}$$

Using the Bigram model, $P(e_1e_2...e_m|S_1S_2...S_N)$ can be approximated as

$$P(e_1e_2...e_m|S_1S_2...S_N) = \prod_{i=1}^{N} P(e_{j_i}...e_{j_i+l}|S_i) \tag{7.5}$$

which $e_{j_i}...e_{j_i+l}$ matches the left side of a lexical rule matching with S_i, and $j_i, j_i + 1, ..., j_i + l$ is a sequence of word positions within the input sentence e.
$P(S_1S_2...S_N)$ can be also approximated as

$$P(S_1S_2...S_N) = \prod_{i=1}^{N-1} P(S_{i+1}|S_i) \tag{7.6}$$

Thus, we get

$$P(e_1e_2...e_m|S_1S_2...S_N) \times P(S_1S_2...S_N) = \prod_{i=1}^{N-1} P(S_{i+1}|S_i) \times \prod_{i=1}^{N} P(e_{j_i}...e_{j_i+l}|S_i) \quad (7.7)$$

To find the sequence of lexical rules that maximizes the formula (7.7), the Viterbi algorithm [57], a kind of dynamic programming, can be used. If the rule has t variables and each variable consists of l elements then the complexity is $l^2 \times t$, while the recursive way be l^t. Thus, each rule can define a translation score, and output translation for the input sentence can be sorted according to their score values. Therefore, our HMM-based method can avoid the exponential calculation problem by using dynamic programming. In addition, it can sort translation results according to the better accuracy without any complex process on set of template rules. Interestingly, it draws a new perspective for applying statistical machine learning theory to example based machine translation.

7.3.2 Estimation of HMM model for translation

We will now describe our HMM-based method translation. An HMM is specified by a five-tuples (O, L, A, B, Π), where L and O are the set of lexical rules and the output alphabet, and Π, A, B are the probabilities for the initial state, state transition, and symbol emission, respectively.

The HMM-based method for translation is estimated by using the Baum-Welch learning [58] described as follows. The corpus of source sentences and target sentences are used to generate observed sequences. Each input sentence will be translated by using lexical rules if its translated output is the same with a translated sentence in the corpus. After obtaining a sequence of lexical rules, a sequence of observed symbols is generated because each observed symbol is in the left side of a lexical rule. Therefore, using a set of template rules and the corpus, we can generate a training data O_{train} in the form

$$\begin{aligned} O_{t_1}O_{t_1+1}...O_{m_1} &\leftrightarrow l_{t_1}l_{t_1+1}...l_{m_1} \\ O_{t_2}O_{t_2+1}...O_{m_2} &\leftrightarrow l_{t_2}l_{t_2+1}...l_{m_2} \\ &.... \\ O_{t_k}O_{t_k+1}...O_{m_k} &\leftrightarrow l_{t_k}l_{t_k+1}...l_{m_k} \end{aligned}$$

Here $O_{t_k}O_{t_k+1}...O_{m_k}$ is a sequence of observed symbols associated with a sequence of lexical rules $l_{t_k}l_{t_k+1}...l_{m_k}$.

Denote by $C(l^j)$,$C(l^j, l^k)$ and $C(O^j, l^k)$ the number of occurrences of the lexical rule l^j, the number of occurrence of the lexical rule l^j following the lexical rule l^k, and the number of occurrences of the observed symbol O^j corresponding to the lexical rule l^k, respectively. With these notations, the initialization algorithm for estimating an HMM model on the training data above is described in Algorithm 10.

To avoid the data spareness problem, we used a additive smoothing technique [117] for probabilities in Algorithm 10.

Assume that we are given O'_{train} is an unlabeled training data which consists of only a set of observed sequences. After initializing the probabilities of observed symbols and lexical rules using Algorithm 10 on the training data O_{train}, the Baum-Welch learning[58] is used to estimate the HMM for sentence reduction by maximizing $P(O'_{train}|A, B, \Pi)$.

Algorithm 10 An initialization algorithm

for all lexical rule l^j **do**
  for all lexical rule l^k **do**
    $P(l^j|l^k) = \frac{C(l^j,l^k)}{C(l^j)}$
  end for
end for
for all lexical rule l^j **do**
  for all observed symbols O^l **do**
    $P(O^l|l^j) = \frac{C(O^l,l^j)}{C(l^j)}$
  end for
end for

7.4 Chunking Based Example Based Translation

7.4.1 Chunking Based Example Based Translation Architecture

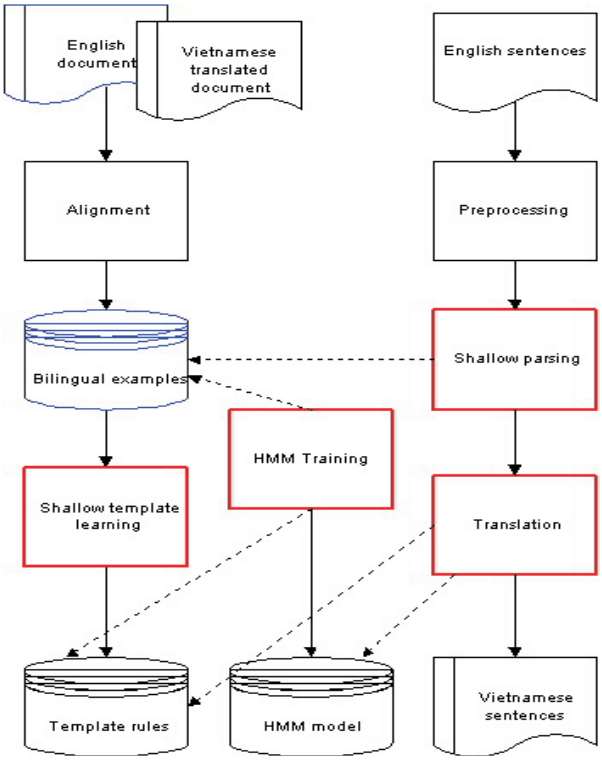

Figure 7.4: The Framework of Chunking Based Example Based Translation

The template translation learning[24] is modified to a new one by incorporating linguistic information by using a shallow parsing. We call the modified algorithm as a shallow template translation learning algorithm. This algorithm is considered as the main component in a chunking based example based machine translation system.

Figure 7.4 shows a chunking-based example-based machine translation system (CEBMT) which consists of two phases, a learning phase and a translation phase. In the learning

phase, an alignment process is used to align English sentences with Vietnamese sentences. They were then edited manually by human expert to obtain bilingual examples. These bilingual examples were annotated linguistic information by performing a shallow parsing to chunk each sentence into chunking labels. A shallow temple learning algorithm is then used to generate template rules from examples. Afterward, a Hidden Markov Model for the system is established by using the corpus and the template rules. In the translation phase, an input sentence were chunked by using a shallow parsing, then a variant of the dynamic algorithm (i.e the viterbi algorithm) is used to find translation outputs for that input sentence.

7.4.2 Translation Template Learning Based Shallow Parsing

Shallow Parsing

Shallow parsing is the task of dividing sentences into non-overlapping segments on the basis of fairly superficial analysis. By shallow parsing, a sentence of n words is divided into m chunking labels. These chunking labels are in the syntactic level such as noun phrase, verb phrase or in high level description such as named entity types (PERSON, LOCATION, NAME, DATE. etc). This task has attracted much attention on natural language processing community in recently and achieve a high performance. For example, the task of NP chunking obtained a high accuracy (96.29%)[116].

The use of chunking is applied to many tasks including information extraction, text summarization, and information retrieval. In this section, we investigate the use of it to generate reliable template rules for template learning. The translation template learning using shallow parsing is named as shallow translation template learning.

Match Sequence incorporating Linguistic Information

Recall that $M_{a,b}$ is a match sequence of two examples E_a and E_b in which a segment within $M_{a,b}$ is associated with an linguistic information. A similarity is labeled if and only if its common items is the same type of phrase in both examples. A different segment is labeled if and only if two components in this segment come from the same type of phrase in both example. Suppose that, LS_k^1 is a chunking-symbol associated with a similarity segment S_k^1 and LD_k^l is a symbol associated with a different segment D_k^l, where k=0, 1, 2,... and i=0, 1.

Figure 7.5 shows the two examples are chunked using a shallow parsing to obtain a match sequence and a sequence of chunking-symbols (e.g LD_0^1 =NP, LD_0^2 =VP, LD_0^3=NP, LD_0^4 =PP). After a match sequence is found for two translation examples the template learning algorithms infer translation templates from that match sequence to obtain template rules.

In the following sections, the learning similarity translation template and the learning difference translation template are manipulated on match sequences. The main goal of our algorithm is to produce template rules which are associated linguistic information where each variable in template rules is labeled by using a shallow parsing. Furthermore, the set of lexical rules can be classified into several classes, which class are part of speeches or symbols of shallow parsing.

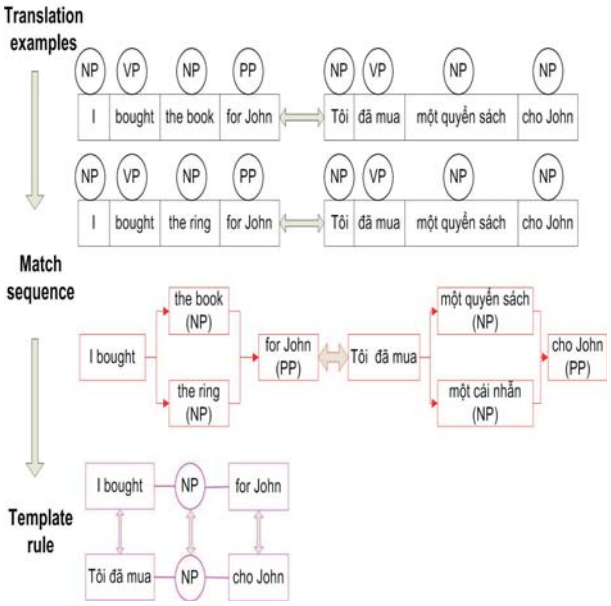

Figure 7.5: Example of chunking translation template learning.

Learning similarity translation templates using shallow parsing

If there exists only a single difference in both sides of a match sequence, i,e, $n = m = 1$, then these different constituents must be the translation of each other. In other words, we are able to locate the corresponding difference in the match sequence. In this case, the match sequence must be in the following form.

$$S_0^1 X^1 S_1^1 \leftrightarrow S_0^2 X^2 S_1^2 \quad if \quad (\ X^1 \leftrightarrow X^2) \tag{7.8}$$

Recall that LD_0^1 and LD_0^2 are shallow information of two variables X_1 and X_2 respectively. The template rule above can be expressed as in the form following:

$$S_0^1 LD_0^1 S_1^1 \leftrightarrow S_0^2 LD_0^2 \quad if \quad (LD_0^1 \leftrightarrow LD_0^2) \tag{7.9}$$

Furthermore, the following two lexical translation templates are learned from the corresponding differences $(D_{0,a}^1, D_{0,b)^1)}$ and $(D_{0.a}^2, D_{0,b}^2)$

$$\begin{array}{l} D_{0,a}^1[LD_0^1] \leftrightarrow D_{l,a}^2[LD_0^2] \\ D_{0,b}^1[LD_0^1] \leftrightarrow D_{l,b}^2[LD_0^2] \end{array} \tag{7.10}$$

The lexical translation template above explains the chunk label LD_0^1 and LD_0^2 are associated with the two lexical rules $D_{0,a}^1 \leftrightarrow \mathrm{D}_{0,a}^2$ and $D_{0,b}^1 \leftrightarrow \mathrm{D}_{0,b}^2$, respectively. For example, since the match sequence contains a single difference in both sides, the following similarity translation template and two lexical translation templates from the corresponding difference "(the book, the ring) and (mot quyen sach, mot cai nhan)" can be inferred.

"I bought" NP "for John" ↔ "Toi da mua" NP "cho John"

"the book"[NP] ↔ "mot quyen sach"[NP]

"the ring" [NP] ↔"mot cai nhan"[NP]

If the number of differences are equal on both sides, but more than one, $1 < n = m$. Assume that $n-1$ corresponding difference in the match sequence were located as follows: $((D^1_{k_1} D^1_{l_1})...(D^1_{k_{n-1}} D^2_{l_{n-1}}))$
It means that the k_i difference $(D^1_{k_i,a}, D^1_{k_i,b})$ in the source language corresponds to the l_j difference $(D^2_{l_j,a}, D^2_{l_j,b})$ in the target language following the rules below.

$$\begin{array}{l} D^1_{k_i,a}[LD^1_k] \leftrightarrow D^2_{l_j,a}[LD^2_k] \\ D^1_{k_i,b}[LD^1_k] \leftrightarrow D^2_{l_j,b}[LD^2_k] \end{array} \tag{7.11}$$

Assume that the unchecked corresponding differences are $D^1_{k_n}$ and $D^2_{l_n}$. Since all difference elements in the left side of the match sequence aligns with only difference elements in the right side of the match sequence. Thus, the different element $(D^1_{k_n})$ has to align with the different element $D^2_{l_n}$. Therefore, we are able to learn a template rule for the case of n differences in both sides if at least $n-1$ differences were located. From the match sequence $M_{a,b}$ we try to replace the difference in both sides with variables, $D^1_{k_i}$ with $LD^1_{k_i}$ and $D^2_{k_i}$ with $LD^2_{k_i}$ respectively. Let $M_{a,b}WDC$ be a match sequence that all differences in $M_{a,b}$ are replaced by chunking labels.

$$M_{a,b}DVC = S^1_0 LD^1_0 ... LD^1_{n-1} S^1_n LD^1_n \leftrightarrow S^2_0 LD^2_0 S^2_1 ... LD^2_{n-1} S^2_n LD^2_n \qquad for \qquad 1 \leq n \tag{7.12}$$

Thus, we get a new template rule by a mathematic induction as follow:

$$M_{a,b}DVC \qquad if \qquad LD^1_{k_1} \leftrightarrow LD^2_{l_1} ...and \qquad LD^1_{k_n} \leftrightarrow LD^2_{l_n} \tag{7.13}$$

In addition, the two lexical translation templates are learned from the last corresponding differences.

$$\begin{array}{l} D^1_{k_n,a}[LD^1_{k_n}] \leftrightarrow D^2_{l_n,a}[LD^2_{l_n}] \\ D^1_{k_n,b}[LD^1_{k_n}] \leftrightarrow D^2_{l_n,b}[LD^2_{l_n}] \end{array} \tag{7.14}$$

Therefore, the algorithm of similarity template learning using shallow parsing can be organized as shown in Algorithm 11.

Learning Shallow Difference translation templates

For convince, let $M^1_{a,b}$ and $M^2_{a,b}$ be the left side and the right side of a match sequence $M_{a,b}$ respectively. Let $M_{a,b}WSC$ is a match sequence in which all similarities were replaced with their corresponding chunking labels. Suppose that $M_{a,b}WSC$ are in the following form.

$$M_{a,b}WSC = LS^1_0 D^1_0 ... LS^1_n D^1_n \leftrightarrow LS^2_0 D^2_0, ... LS^2_m D^2_m \qquad for \qquad n, m \geq 1 \tag{7.15}$$

Since $D^1_i = (D^1_{i,a}, D^1_{i,b})$ and $D^2_i = (D^2_{i,a}, D^2_{i,b})$, so we define M_aSVC and M_bSVC by following formulas.

$$M_aWSC = LS^1_0 D^1_{0,a} ... LS^1_n D^1_{n,a} \leftrightarrow LS^2_0 D^2_{0,a} ... LS^2_m D^2_{m,a} \qquad for \qquad n, m \geq 1 \tag{7.16}$$

$$M_bWSC = LS^1_0 D^1_{0,b} ... LS^1_n D^1_{n,b} \leftrightarrow LS^2_0 D^2_{0,b} ... LS^2_m D^2_{m,b} \qquad for \qquad n, m \geq 1 \tag{7.17}$$

Algorithm 11 The Shallow Similarity Translation Template Learning Algorithm

1: Let the match sequence $M_{a,b}$ for the pair of translation examples E_a and E_b and a sequence of chunking symbol be:

$$S_0^1 D_0^1 S_1^1 ... D_{n-1}^1 S_n^1 \leftrightarrow S_0^2 D_0^2 S_1^2 ... D_{m-1}^2 S_m^2$$
and
$$LS_0^1, LD_0^1, LS_1^1, ..., LD_{n-1}^1, LS_n^1 \leftrightarrow LS_0^2, LD_0^2, LS_1^2, ..., LD_{m-1}^2, LS_m^2$$
$1 \leq \mathrm{n}, \mathrm{m}$

2: **if** $n = m = 1$ **then**

3: Infer the following templates:
$S_0^1 LD_0^1 \leftrightarrow S_0^2 LD_0^2 S_1^2$ if $LD_0^1 \leftrightarrow LD_0^2$

$$D_{0,a}^1[LD_0^1] \leftrightarrow D_{0,a}^2[LD_0^2]$$
$$D_{0,b}^1[LD_0^1] \leftrightarrow D_{0,b}^2[LD_0^2]$$

4: **else**

5: **if** $1 \leq n = m$ and $n-1$ corresponding differences can be found in $M_{a,b}$ **then**

6: Assume that the unchecked corresponding difference is
$((D_{k_n,a}^1, D_{k_n,b}^1), (D_{l_n,a}^2, D_{l_n,b}^2))$
Assume that the list of corresponding differences is
$((D_{k_1}^1, D_{l_1}^1), ..., (D_{k_{n-1}}^1, D_{l_{n-1}}^2))$

7: **for** each corresponding difference $(D_{k_i}^1, D_{l_i}^2)$ **do**

8: Replace $D_{k_i}^1$ with $LD_{k_i}^1$ and $D_{l_i}^2$ with $LD_{l_i}^2$

9: Get a new match sequence $M_{a,b}WDC$ by inferring the following template
$M_{a,b}WDC$ if $LD_{k_1}^1 \leftrightarrow LD_{l_1}^2 ...$ and $LD_{k_n}^1 \leftrightarrow LD_{l_n}^2$

$$D_{k_n,a}^1[LD_k^1] \leftrightarrow D_{l_n,a}^2[LD_k^2]$$
$$D_{k_n,b}^1[LD_k^1] \leftrightarrow D_{l_n,b}^2[LD_k^2]$$

10: **end for**

11: **end if**

12: **end if**

If there exits only a single non-empty similarity in both sides of a match sequence $M_{a,b}$, then these similar constituents must be translations of each other. In this case, each side of the match sequence can contain one or two differences, and they may contain different number of differences. On the other hand, each side ($M_{a,b}^1 \leftrightarrow M_{a,b}^2$) can be one of the following:

S_0^i, D_0^i, S_1^i where S_0^i is non-empty and S_1^i is empty,
S_0^i, D_0^i, S_1^i where S_1^i is non-empty and S_0^i is empty,
$S_0^i, D_0^i, S_1^i, D_1^i, S_2^i$ where S_1^i is non-empty and S_2^i is empty.

In these cases, we replace the non-empty similarity in $M_{a,b}^1$ a chunking label and separate difference pairs in the match sequence to get two match sequences with similarity variables, namely

$$\begin{aligned} M_a^1 WSC &\leftrightarrow M_a^2 WSC \\ M_b^1 WSC &\leftrightarrow M_b^2 WSC \end{aligned} \tag{7.18}$$

Assume that the number of non-empty similarities in both source language and target

language are the same and they are equal to $n(n \geq 1)$. Since in $M_{a,b}$ the non-empty similarities in $M^1_{a,b}$ only corresponds with non-empty similarities in $M^2_{a,b}$. Thus, if $n-1$ corresponding non-empty similarities were located in $M_{a,b}$,that are:

$(S^1_{k_1,a}, S^2_{l_1,a}), (S^1_{k_2,a}, S^2_{l_2,b}), ...(S^1_{k_{n-1},a}, S^2_{l_{n-1},b})$ and a unlocated difference is $S^1_{k_n}$ and $S^2_{l_n}$, then we can say that $S^1_{k,n}$ is located with $S^2_{l_n}$.. Thus, we can generate template rules for n case as follows:

M_aWSC if $LS^1_{k_1} \leftrightarrow LS^2_{l_1}...$ and $LS^1_{k_n} \leftrightarrow LS^2_{l_n}$

M_bWSC if $LS^1_{k_1} \leftrightarrow LS^2_{l_1}...$ and $LS^1_{k_n} \leftrightarrow LS^2_{l_n}$

In addition, the following atomic translation template is learned from the last corresponding similarities.

$S^1_{k_n}[LS^1_{k_n}] \leftrightarrow S^2_{l_n}[LS^2_{l_n}]$

To summary the description mentioned above, we describe a formal description of the shallow difference TTL algorithm in Algorithm 12.

Algorithm 12 The Shallow Difference Template translation Learning Algorithm

Input: A match sequence $M_{a,b}$

1: **if** the number of similarity $M^1_{a,b}$ = the number of similarity $M^2_{a,b} = n \geq 1$ and $n-1$ corresponding similarities can be found in $M_{a,b}$ **then**
2: Assume that unchecked corresponding similarities are $(S^1_{k_n}, S^2_{l_n})$.
Assume that the list of corresponding similarities is $(S^1_{k_1}, S^2_{l_1})...(S^1_{k_i}, S^2_{l_i})...(S^1_{k_n}, S^2_{l_n})$ including unchecked ones.
3: **for** each corresponding difference $(S^1_{k_i}, S^2_{l_i})$ **do**
4: Replace $S^1_{k_i}$ with $LS^1_{k_i}$ and $S^2_{l_i}$ with $LS^2_{l_i}$ to get the new match sequence $M_{a,b}WSV$
5: Divide $M_{a,b}WSV$ into M_aWSV and M_bWSV by separating differences.
6: Infer the following templates:
M_aWSV if $LS^1_{k_1} \leftrightarrow LS^2_{l_1}...$,and $LS^1_{k_n} \leftrightarrow LS^2_{l_n}$
M_bWSV if $LS^1_{k_1} \leftrightarrow LS^2_{l_1}...$,and $LS^1_{k_n} \leftrightarrow LS^2_{l_n}$
$(LS^1_{k_n}, LS^2_{l_n})$: $S^1_{k_n} \leftrightarrow S^2_{l_n}$
7: **end for**
8: **end if**

7.4.3 Template Translation Using Shallow Parsing

This translation method is similar to the original method as mentioned in the previous section. However, it is extended by using the shallow information which are obtained from a shallow parsing. We summary the ideal behind this algorithm as follows:

The given input sentence $e_1e_2...e_m$ was chunked by using the shallow parsing [118]. Assuming that it was chunked into $LC_1LC_2...LC_k$, in which LC_j $(j = 1, k)$ is a chunking label. Each template rule r_i in a set of template rules $r_1, r_2, ..., r_d$ is in the following form:

$r_i = LS_1LS_2..LS_n \leftrightarrow LT_1LT_2...LT_k$, Here LS_j $(j = 1, n)$ and LT_j $(j = 1, k)$ are chunking labels.

The shallow template translation tries to find all possible ways to replace the variables with phrases in SL so that the input sentence $e_1e_2...e_m$ can be produced from this rule. This process is the same as that of the template translation algorithm. But for each variable LS_j, a phrase in SL is now a candidate for replacing the variable LS_j if its

chunking label is LS_j. Next, it finds each corresponding phrase in TL within the set of lexical rules with a phrase in SL so that the chunking label of a lexical rule is the same as that of the variable. Then, the translation results are obtained using these lexical rules. Therefore, the behavior of template translation using sallow parsing is the same with that of the original algorithm, so we are able to apply both the original translation template and the translation template based HMM.

7.4.4 The combination of CEBMT and RBMT

We have developed a Rule Based Machine Translation system (RBMT) for English and Vietnamese languages namely LVTrans which uses a large of linguistic knowledge encoded manually by humans and it obtained a good translation performance[119]. However, the linguistic rules encode by human could not cover almost every cases and using human manually editing uncovered cases sometime affect to other rules and this tends to obtain wrong translations for other examples. An sufficient way is to incorporate rule based translation system with example based machine translation system. A given English sentence firstly using CEBMT to translate it, we then obtain a translation output together with its score. If the score given by CEBMT is lower than a threshold obtaining by experiments, LVTrans will be used to translate it into Vietnamese language. We can also use the following simple strategy to enhance the performance of LVTrans by enriching more example translations. An input sentence could not work well in translating using LVTrans will be considered to enhance its translation performance. Instead of adding or modifying linguistic rules, we find all relevant examples to that input and using translation template learning algorithms to generate template rules for CEBMT system.

7.5 Example Based Machine Translation for Sentence Reduction

7.5.1 Template Learning for Sentence Reduction

Template learning algorithm has been applied to machine translation [23]. In order to apply the algorithm to sentence reduction, some definitions are described as follows.

Template reduction rules

Although in this paper we consider the sentence reduction problem on one language, we here discuss the sentence reduction problem in the general case of reducing a sentence from a source language (SL) to a reduced sentence in a target language (TL). A template reduction rule is defined in the form $S_1S_2...S_i...S_m \leftrightarrow T_1T_2...T_j...T_k$ in which S_i are either constants or variables in SL, and T_j are either constants or variables in TL. A constant can be a phrase or a word, while a variable can be substituted by constants. Each variable S_i in the left side of the rule is aligned with a variable T_j in the right side of the rule.

Figure 7.6 depicts an example of a template reduction rule where S_1 and T_1 are variables, and the phrase "is very good and includes a tutorial to get you started" is reduced to the phrase "is very good".

We call a template reduction rule having no variable as a lexical rule. A lexical rule can be used as a value of a variable in a template reduction rule in order to reduce a

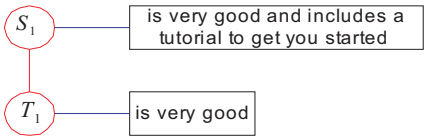

Figure 7.6: Template reduction rule example

long sentence into a short sentence. For example, if the lexical rule "The document" $\leftrightarrow$ "Document" is in the set of template reduction rules, then the input sentence "The document is very good and includes a tutorial to get you started" can be reduced to the sentence "Document is very good" by using the template reduction rule in Figure 7.6.

Learning template reduction rules

We apply the TTL algorithm [23] to infer template reduction rules using similarities and differences between two examples taken from a corpus of pairs of long sentences and their corresponding reduced sentences.

Formally, a reduction example $E_a : E_a^1 \leftrightarrow E_a^2$ is composed of a pair of sentences, E_a^1 and E_a^2, where E_a^1 is the original sentence in SL and E_a^2 is reduced sentence in TL. A similarity between two sentences of a language is a non-empty sequence of common items (root words or morphemes) in both sentences. A difference between two sentences of a language is a pair of sub-sequences having no common items, one is sub-sequence of the first sentence and the other of the second sentence. Given two reduction examples (E_a, E_b), our problem is to find similarities between the constituents of E_a and E_b. A sentence is considered as a sequence of lexical items. If no similar constituents (viewed as subsequences of lexical items) can be found, then no template reduction rule is learned from these examples. If there are similar constituents then *a match sequence* $M_{a,b}$ in the following form is generated

$$M_{a,b} = S_0^1 D_0^1 ... D_{n-1}^1 S_n^1 D_n^1 \leftrightarrow S_0^2 D_0^2 S_1^2 ... D_{m-1}^2 S_m^2 D_m^2 \tag{7.19}$$

Here $n, m \geq 1$, S_k^i represents a similarity between E_a^i and E_b^i, and $D_k^i = (D_{k,a}^i, D_{k,b}^i)$ represents a difference between E_a^i and E_b^i, where $D_{k,a}^i$ and $D_{k,b}^i$ are non-empty subsequences of items between two similar constituents. For instance, consider the following reduction examples:

(1)
E_a="The document is very good and includes a tutorial to get you started" $\leftrightarrow$ "Document is very good".
E_a^1="The document is very good and includes a tutorial to get you started" and E_a^2="Document is very good".
(2)
E_b="This paper is very good and includes a tutorial to get you started" $\leftrightarrow$ "Paper is very good". Where E_b^1 ="This paper is very good and includes a tutorial to get you started" and E_b^2 ="Paper is very good".

For these reduction examples, the matching algorithm obtains the following match sequence.
$M_{a,b}$=(The document, This paper) "is very good and includes a tutorial to get you started" $\leftrightarrow$ (Document, Paper) "is very good".

That is,
S_0^1 = "", D_0^1 = (The document, This paper), $D_{0,a}^1$ = (The document), $D_{0,b}^1$ = (This paper)
S_1^1 = "is very good and includes a tutorial to get you started",
S_0^2 = "", D_0^2 = (Document, Paper), $D_{0,a}^2$ =(Document), $D_{0,b}^2$=(Paper), S_1^2 = "is very good"

Intuitively, in the above example the similar elements and the different elements in the left hand side should be aligned with the similar elements and the different elements in the right hand side, respectively. Thus, in this case "(The document, This paper)" is aligned with "(Document, Paper)", and "is very good and includes a tutorial to get you started" is aligned with "is very good". We consider "(The document, This paper)" and "(Document, Paper)" as variables, and we can generate the template reduction rule in Figure 7.6. We also obtain two lexical rules
$D_{0,a}^1 \leftrightarrow D_{0,a}^2$, or "The document" $\leftrightarrow$ "Document"
$D_{0,b}^1 \leftrightarrow D_{0,b}^2$, or "This paper" $\leftrightarrow$ "Paper".

We have illustrated that the problem of sentence reduction can be solved by using translation template learning algorithm. The advantage of translation template learning method is that it does not need to parse the input sentence. Obviously, one can use some preprocessing techniques such as morphological analysis and shallow parsing to enrich linguistic information to the given input sentence. For the simplicity of representation, in this paper we only used the morphological analysis to represent an input sentence.

Figure 7.7 depicts the framework of sentence reduction using template learning. The corpus of reduction examples is used to generate template rules in the translation template learning algorithm. In the reduction process, a given long sentence will be represented in the surface level by preprocessing such as pos tagging, morphological analysis, or chunking, etc. The reduction using template rules will be then performed in order to generate the reduced sentence. This process will be presented in next subsections.

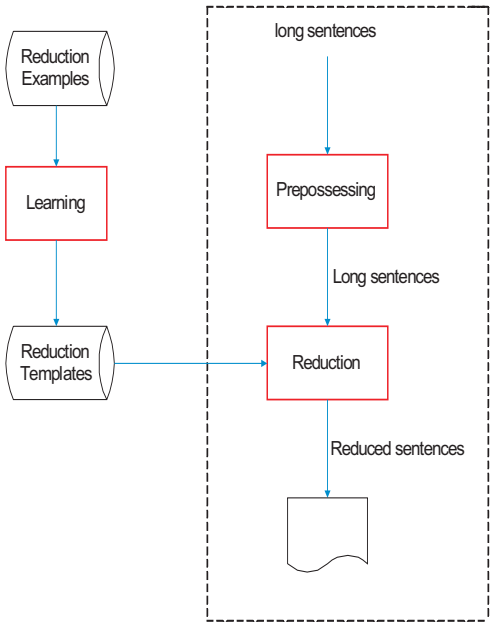

Figure 7.7: Framework of sentence reduction using template learning

7.5.2 Sentence Reduction using Template Rule

The main idea of using template rules for sentence reduction is the same as that of using it on translation. We summary two algorithms for sentence reduction using template reduction rules as follows. In the first one, the original method of translation template learning is applied to reduction problem. In the second algorithm, we proposed a novel method using Hidden Markov Model that is efficient when the input sentence is long and the number of template reduction rules is big.

Sentence reduction using template rules

To illustrate the behavior of the sentence reduction using template rules, let us consider the reduction example of the sentence "It is likely that two companies will work on integrating multimedia with database technology" using the template rule[1] in Figure 7.8.

Two phrases "It is likely that" and "will work on" within the input sentence are matched with the template rule. The reduction algorithm then tries to find all possible choices to replace variables S_2 and S_4 using lexical rules. For variables S_2 and S_4 it finds all lexical rules whose left side is matched with "two companies" and "integrating multimedia with database technology", respectively. Figure 7.8 shows three choices for S_2 and S_4, from which we have six reduction results. Intuitively, the best reduction output

[1]The template rule is learned using two following examples:
It is likely that he will work on through storm. ↔ He will work on through the storm.
It is likely that she will work on this book. ↔ She will work on this book.

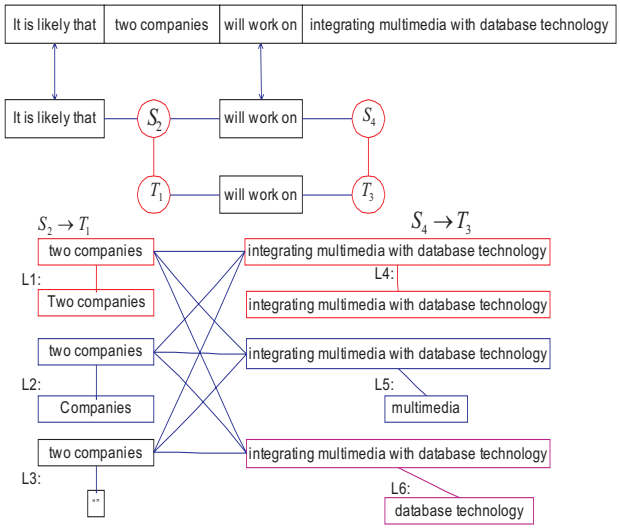

Figure 7.8: Example of reduction based HMM

is "Two companies will work on integrating multimedia with database technology".

As the problem of translation, there are two obstacles when using the original template reduction:

- How to determine the best reduced outputs when using template reductions.
- Suppose that a template rule has t variables and each variable has l matched lexical rules, so we have l^t choices for reduction. How can we deal with this exponential calculation?

To solve the problems, the HMM-based method for template translation as described in subsection 7.3.1 can be applied. The difference of applying method for sentence reduction in comparison with machine translation is that of estimating HMM models. We estimated HMM models for sentence reduction on the reduction corpus which consists of a set of long sentences and their reductions.

7.6 Experimental Results

7.6.1 Translation results

In order to assert that our method can enhance the translation accuracy with a low complexity. We implemented an English Vietnamese translation system and tested it on a corpus of 1,200 bilingual sentences collected manually from some text books and newspapers. All English sentences within the corpus were chunked by using the shallow parser [118] and Vietnamese sentences were chunked manually by three annotators.

Figure 7.9 shows the relation of the number of template rules and lexical rules respectively, which are obtained by using the template learning algorithm. The result shows

how the size of the template rules for a bilingual corpus of English-Vietnamese language changes.

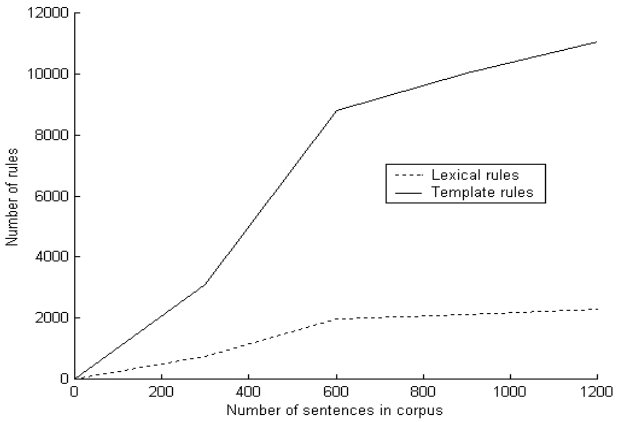

Figure 7.9: The relation of the number of lexical rules and the number of template rules with the number of sentences within the corpus.

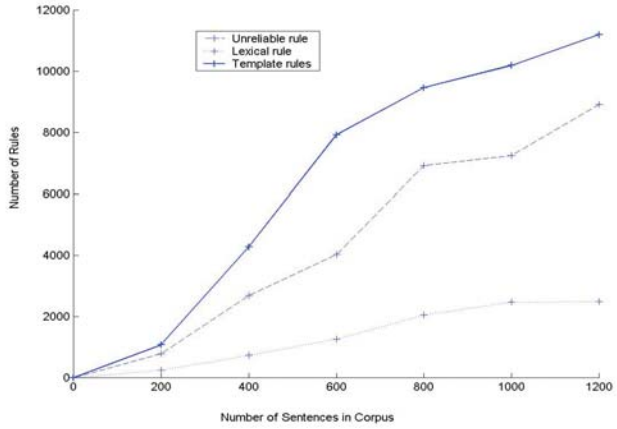

Figure 7.10: The relation of lexical rules, template rules and unreliable rules with the size of corpus

The number of sentences for one language in our corpus is from 300 to 1,200 sentences. The solid line and doted line show the relation between the number of template rules and the number of lexical rules with the number of sentences within the corpus, respectively. Figure 7.10 depicts the relation between the number of template rules and lexical rules when applying the shallow template learning algorithm. It also shows the number of unreliable rules when performing the shallow template learning algorithm. The unreliable rules mean those rules has no chunking labels. The frequency of a chunking label which appears on reliable rules are shown in figure 7.10. This result shows that the number of NP and VP are the highest in the various sizes of corpus. This motives that recognizing NP and VP are very important in our chunking based examples translation method. We suspect that only use of two chunks: NP and VP are enough for CEBMT system. This problem will be addressed in future work.

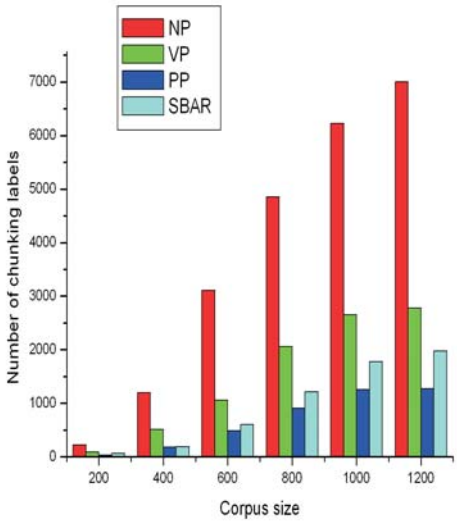

Figure 7.11: The distribution of chunking label in the corpus.

The number of template rules and the number of lexical rules generated by the template translation learning is 11,034 rules and 2,287 rules, respectively. Using the template rules and the data corpus, we obtained a set of observed sequences for estimating HMM model described in section 7.3.2, then the initialized parameters for the HMM model is estimated by performing the Algorithm 10 in which the number of hidden states is equal to the number of lexical rules. In addition, the training data for estimating such HMM model consists of 1,200 observed sequences, where each of which is a substrings corresponding to a lexical rule. Algorithm 10 is used to initialize HMM mode by using 1,100 observed sequences. After that, the final model are estimated by performing the Forward and Backward algorithm [58] on the remaining sequences.

The template rules and shallow template rules are trained by using the template translation learning and shallow template translation learning, respectively. These template rules are estimated by using the forward backward learning as described in previous sections. After obtained these template rules, we are now comparing translation results of the proposed systems and the original systems. We conduct four translation methods as follows: The original translation method, the translation method using HMM model, the translation method using the shallow template learning and the combination of using HMM and shallow template learning.

Finally, we tested the translation accuracy by using the sentences within the corpus with an evaluation method as follows. The translation accuracy is calculated by the rate of the number of correct translations among the total translation outputs. This formula is given bellow.

$$Accuracy = \frac{X}{Y} \tag{7.20}$$

where X and Y be the number of correct translations and the total translation outputs, respectively.

Using the formula (7.19), we obtained the results as shown in Table 7.2.

Table 7.2: Accuracy of four translation algorithm: Translation template learning, Shallow template translation learning, Template translation learning using HMM, and shallow translation template learning using HMM.

Method	Translation Accuracy
TTL	0.34
STTL	0.52
TTL-HMM	0.81
STTL-HMM	0.87

Table 7.2 shows the translation results of the template translation learning, shallow template learning, the template learning using HMM and the shallow template learning combining with the HMM, respectively. The shallow template learning improved the original algorithm. This was due to the fact that performing a translation in the reliable rules is better than the original rules. The combining of the shallow template learning and the HMM model achieved the best result. This indicates that the combined algorithm inherits the advantage of two algorithms, the shallow template learning and the HMM model.

7.6.2 Reduction Results

The corpus for sentence reduction is collected from Vietnam agency web-site(http:// www.vnagency.com.vn), these sentences in the corpus are then used to generate template rules for our reduction methods. The number of template rules and the number of lexical rules using the translation template learning are 11,034 rules and 2,287 rules, respectively. Using the template rules and the data corpus, we obtained the training data for estimating the HMM model for sentence reduction. The training data for estimating the HMM model consists of 1,500 observed sequences, in which each sequence is correspond to a sequence of lexical rules. We obtained other 1,200 sentences from the same web-site, in which the number of sentences which cannot be recognized by the template rules takes 10% percent. We selected randomly 32 sentences among 1,200 sentences for testing and the remained sentences are used to extract observed sequences for training the HMM model by the Forward-Backward algorithm[58].

It is difficult to compare our methods with previous methods using parsing approach. This is due to the fact that there was no reliable syntax parsing for Vietnamese language. However, we manually parsed all sentences in our corpus in order to use the decision tree based reduction described in [14]. After performing the C4.5 training program [98] on the corpus above, We are able to test the reduction based decision tree model.

We implemented five sentence reduction methods as follows.

- The baseline method is the one that obtains a reduced sentence with the highest word-bigram score.
- The sentence reduction based decision tree model (Decision-Based).
- The proposed reduction method using TTL algorithm (EBSR).

- The reduction method using HMM-based reduction algorithm (EBSR-HMM).
- The EBSR-HMM using the n-best of Viterbi algorithm.

We used the same evaluation as [14] by showing each original sentence in the test corpus to four judges who are Vietnamese, together with five sentence reductions of it and compares with human reduction. The judges were told that all outputs were generated automatically. The order of the outputs was scrambled randomly across test cases. The judges participated in two experiments. In the first experiment, they were asked to determine on a scale from 1 to 10 how well the systems did with respect to selecting the most important words in the original sentence.

In the second experiment, they were asked to determine on a scale from 1 to 10 how grammatical the outputs were. Table 7.3 shows compression ratios in the first column. It means that the lower the compression ratio the short the reduced sentence. Table 7.3 also shows *mean* and *standard deviation* results across all judges, for each algorithm and human.

The results show that the reduced sentences produced by both algorithms are more grammatical and contain more important words than the sentences produced by the baseline. T-test experiments showed these differences to be statistical significant at 95% confident interval for average scores across all judges and the performance of the proposed algorithms are much closer to human performance than the baseline algorithm.

Table 7.3: Experimental Results

Method	Compression	Grammaticality	Importance
Baseline	57.19	4.78 ± 1.19	4.34 ± 1.92
EBRS	65.20	6.80 ± 1.30	6.49 ± 1.80
Decision-Based	60.25	7.40 ± 1.32	7.12 ± 1.73
EBRS-HMM	65.15	7.70 ± 1.20	7.30 ± 1.60
EBRS-HMM (n-best)	68.40	8.20 ± 1.32	7.90 ± 1.45
Human	53.33	9.05 ± 0.30	8.50 ± 0.80

The results shown on Table 7.3 also indicate that the proposed algorithms are closer to and better than the Decision-Based algorithm in term of the grammaticality and the importance measure. Especially, the EBSR-HMM using the n-best of Viterbi algorithm shows the significant improvements in comparison with other algorithms.

Figure 7.12 shows some examples of our reduction methods in testing on Vietnamese language. Each reduction example is attached an English translation. The left hand side and the right hand shows the template rules and the reduction results using these template rules, respectively. There are three examples for reduction using template rules. The results of EBSR and EBSR-HMM in the first example are identical and they are closed to the human reduction. The result of EBSR in the second example is wrong because it did not used a correct lexical rule. Reduction results of EBSR-HMM are good for both example 2 and example 3.

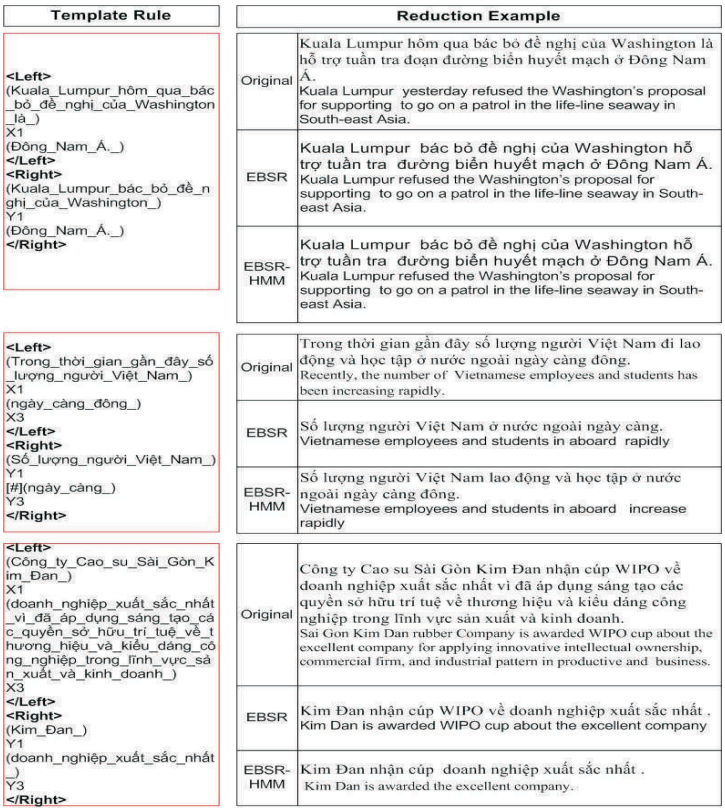

Figure 7.12: Examples of reduction using example-based approach. The template rules are generated by TTL algorithms. Reduction results are obtained using EBSR and using EBSR-HMM.

7.7 Conclusions

We have present a chunking based example-based machine translation system. Our machine translation method is extended from the translation template learning method in both aspects: The learning phase and the translation phase.

In the learning phase, we present a shallow-translation template learning algorithm in order to produce the comprehensive and reliable rules. Experiments show that those comprehensive rules dramatically improved the translation accuracy.

In the translation phase, we propose a translation template using a Hidden Markov Model which can avoid the exponential calculation problem and efficiently improve the translation accuracy in comparison with previous work. Interestingly, it draws a new perspective for applying statistical machine learning theory on example based machine translation. The use of other statistical machine learning models are discussed in future work.

We believe that the combination of the proposed system with the rule based machine translation system is promising since our translation methods are able to evaluate the translation result by obtaining scores from the HMM model.

Furthermore, we discuss a novel application of our translation methods for sentence reduction without syntactic parsing. The advantage of the proposed algorithms is that it

can be implemented for any language without using parsing and achieved acceptable results in comparison with human reduction. We believe that with the same technique using both for translation and for reduction, the performance of cross language summarization could be high.

Chapter 8

Evaluation of Cross Language Text Summarization System

In this chapter, we describe a framework of CLTS using statistical machine learning models which are estimated from training data available. The proposed system consists of three main parts: Sentence Extractions, Sentence Reduction, and Machine Translation. To evaluate the performance of the proposed CLTS system we used human evaluation and ROUGE-evaluation methods in two domains: The scientific domain, and news domain.

8.1 System Architecture

Figure 8.1 shows a framework of cross language text summarization system which describes the relations of the components in the CLTS system and the process of summarizing a given text document to a summary in other languages.

First, a given text document is parsed into a surface level by using a shallow parsing. Second, the sentence extraction phase extracts from the original text document a set of important sentences which mostly reflect the gist meaning of the original. Finally, each long sentence will be reduced to a shorter ones by the sentence reduction phase and the short sentence will be then translated to a sentence in another language by performing the chunking-based machine translation system. All the translation outputs will be combined in order to obtain a summary in another language. The proposed CLTS system mainly uses statistical learning models (HMM, MEM, and SVM) which are estimated automatically on training data. In addition, the CLTS also incorporates with some knowledge databases including Comlex[97], WordNet[92], and LVTRule [119].

8.2 Implementation

We have developed a full-fledged cross language summarization system using statistical machine learning models. The program were mostly written in the C++ language. The main components of the system include:

- **Training data Generation** The training data generation is an off-line process in our CTLS system. It is used to generate training data for both sentence reduction and sentence extraction task.

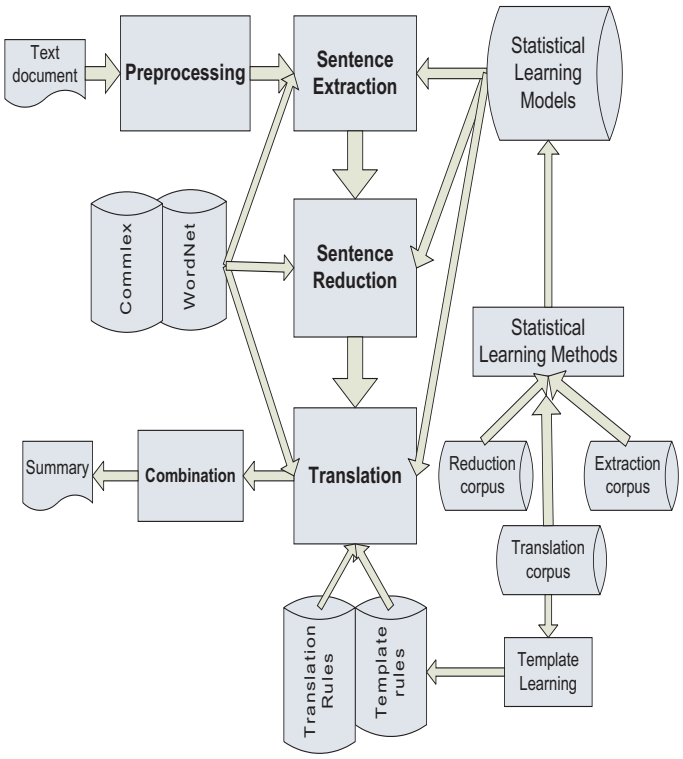

Figure 8.1: A framework of cross language text summarization system

- **Sentence Extraction** The sentence extraction extracts importance sentences from the original document.
- **Sentence Reduction** Each importance sentence is then reduced to a shorter sentence so that the gist meaning of the original one is the same as that of the short sentence. The sentence reduction components used in our implementation are the sentence reduction using maximum entropy models along with our rule based sentence reduction [19]. We used this combination because it can reduce any long sentence which it is not similar to those sentences within the training corpus.
- **Translation** The translation method in our CLTS is the CEBMT method and its combination with the LVTrans machine translation system [119].
- **Fusion of MT and Text Summarization** We also propose a fusion strategy of machine translation and mono-language text summarization to CLTS. The key technique is that we used the same chunking based template learning for machine translation and sentence reduction (the details are described in the previous chapter), this leads to a fusion strategy of sentence reduction and translation as follows. A sentence reduction phases is designed to obtain multiple reduction outputs by using our probabilistic sentence reduction based on statistical machine learning models. In implementation, we applied the probabilistic sentence reduction based on maximum entropy model because its computational times are faster than those of other methods. After obtaining multiple reduction outputs, the given long sentence and these outputs are considered as an input for a chunking based example based

translation and reduction phase. This phase generates all reduction results along with their scores by firstly translating all short sentences and the long sentence, then reducing all previously translation outputs. Afterward, the reduction outputs with highest score are selected as the result for the CLTS system.

For implementation, we also need some other tools for preparing data which include: The Named Entity Recognition[94], The Chunking Parsing[118], and the syntactic parsing[91]. Since our chunking-based example based machine translation system is a part of the LVTrans machine translation system [119], our cross language summarization also uses the LVTrans software for experiments.

To illustrate, we show an example of our reduction and translation results. Figure 8.2 depicts that a long English sentence will be reduced to a shorter sentence and translated it into Vietnamese language.

While dye-polymer optical drives are not yet available commercially , two companies -- Pioneer and Panasonic -- now sell phase-change rewritable optical disk drives .
Pioneer and Panasonic now sell phase-change rewritable optical disk drives .
Người đi tiên phong và Panasonic bây giờ giai đoạn lừa gạt - sự thay đổi rewritable những ổ đĩa quang học.

The files are stored in a temporary directory on the VAX disks , where they are converted to VMS Backup format .
The files are stored in a temporary directory on the VAX disks.
Những hồ sơ được cất giữ trong một thư mục tạm thời trên những đĩa VAX.

President Tran Duc Luong on Thursday wrapped up his official visits to three European countries, including Russia, Hungary and the United Kingdom.
President Tran Duc Luong visits Russia, Hungary and the United Kingdom.
Tổng thống Tran Duc Luong thăm Nga, Hungary và United Kingdom.

Figure 8.2: An running example of reduction and translation.

Figure 8.3 shows an example from beginning to the end to illustrate how a document was summarized to another language. The input document was extracted, then reduced to a shorter sentences. The shorter sentences were translated to the other language. Figure 8.3 also shows three processes of the CLTS system, in which we only extract one sentence from the original text document then our reduction is used to reduce it to a condensation form. The short sentence is translated to Vietnamese language, and it is also a result of the CLTS system.

To this end, Figure 8.4 introduce our cross language text summarization system for summarizing an English text document to Vietnamese language.

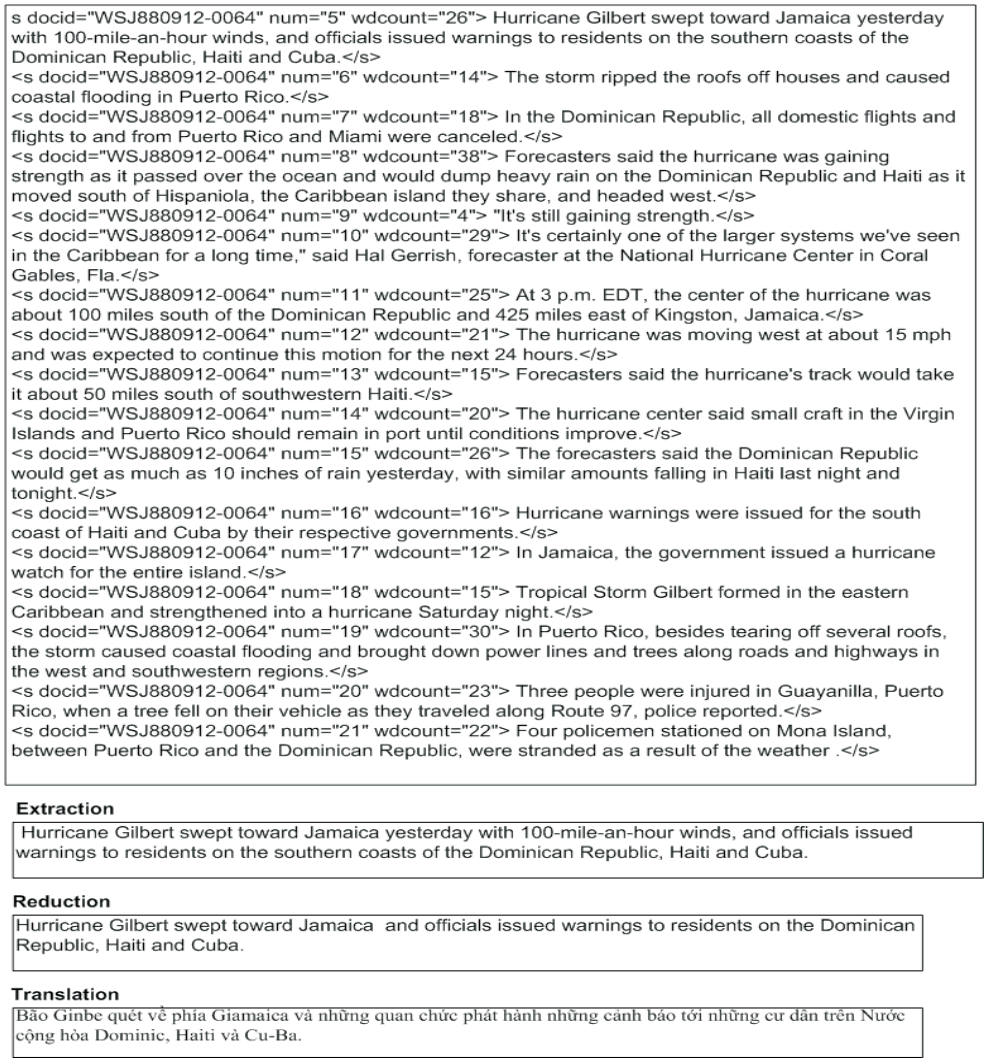

Figure 8.3: A running example of the cross language text summarization for English and Vietnamese language.

8.3 Evaluation of the Overall System

Evaluation is the one of the hardest problem in natural language processing. For cross-language text summarization, there is no consensus on what is good way to evaluate summaries. Fortunately, cross-language summarization is similar to machine translation in term of producing outputs in another language than the language of the given text document. Thus, we can utilize those evaluation methods available in machine translation for cross-language text summarization. Among those evaluation methods, human judgments are commonly applied and it is also used to evaluate the proposed CLTS system.

Recently, some automatic evaluation methods are issued for both machine translation and text summarization [51], [53]. In fact, these evaluations are more suitable for machine translation and text summarization system which their languages are English, whilst in our CLTS system, its target language is Vietnamese and the word orders and function words are two most often used grammaticality facilities in Vietnamese. Despite of these

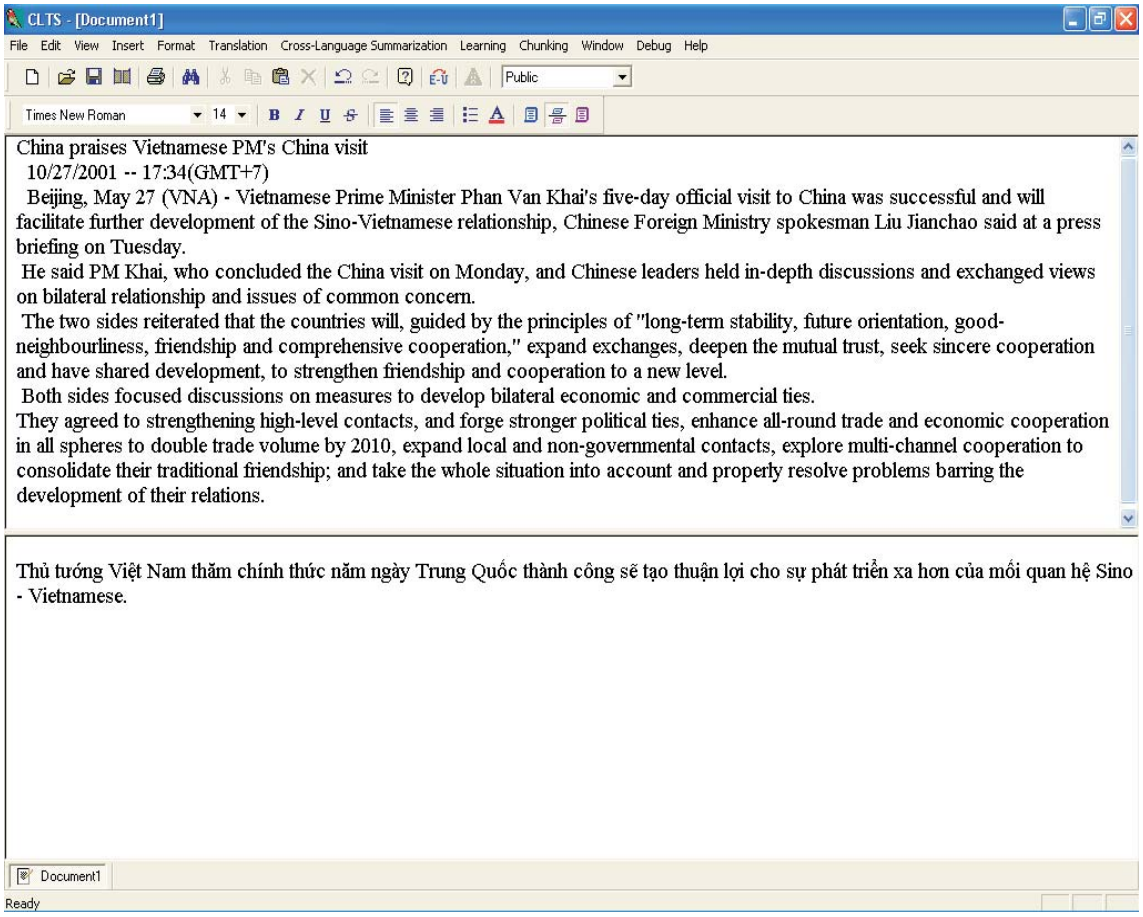

Figure 8.4: A running example of the cross language text summarization for English and Vietnamese language.

limitations, the ROUGE evaluation is used for evaluating the performance of our CLTS system.

8.3.1 Human Judgments

We selected 20 documents from DUC and cmlp data to test our text summarization system in which a subset of 10 text document are used for evaluation our CLTS performance. For DUC data we obtained the DOC61-J and DOC62-J which consists of 10 text documents to test. Since there were no manually gist translation of the DUC corpus and the cmlp-corpus mentioned above, so we could not apply the ROUGE - an automatic text summarization evaluation to measure the performance of the proposed CLTS. Instead, we used the human judgments evaluation in order to measure whether or not a summarization result of a given document performed by CLTS is good. The evaluation is based on the two following criteria measures:

- Content score gives a good idea of what the articles are about. It can be used to determine the gist meaning of an article in its summary.
- Clarity score represents the clarity of the summary obtaining from our system. This

Table 8.1: The performance result of the CLTS system on a test of 10 text documents on cmlp-data set.

Methods	Content score	Clarity score	Computational times (second)
MT-late-gold	6.0	5.25	10.5
MT-before-lead	4.0	3.25	195.6
CLTS	5.5	5.00	56.5

Table 8.2: The performance result of the CLTS system on a test of 10 text documents of DUC data DOC61-J and DOC62-J.

Methods	Content score	Clarity score	Computational times (second)
MT-late-gold	5.00	4.25	8.5
MT-before-lead	4.50	3.75	100.5
CLTS	5.25	4.50	39.6

evaluation aim at determining how well structured of a summary itself. It includes the organization, coherence, and readability of a summary.

Theses scores above are in the scale of 0(poor) to 10(excellent). The evaluation results for cmlp-data and DUC data are shown in Table 8.1 and Table 8.2, respectively.

We do not compare the performance of our system with another because there was no cross language summarization system for the language pairs English-Vietnamese other than our system. Instead, the summary of each document were taken from the corpus (using gold data) and translated to Vietnamese language by using our LVTrans machine translation software[119]. This method is called MT-late-gold. The second method so called MT-before-lead that we use LVTrans to translate the whole English document and then performing an extraction method to obtain a summary document in Vietnamese language. For convenience, we use the Lead-based method (extracting the leading sentences) to extract a summary in Vietnamese language.

Our CLTS results will be then compared with the translation gold-data (MT-late-gold) and the translation of the whole text document method (MT-before-lead). The LVTrans machine translation system [119] have had a good performance in testing on both conversation data and news data. For the news data, we obtained 9.94% for the very good translation, 45.22% for understandable, and 12% for understandable with minor modification. For the conversation data we obtained 78.2% for the very good translation, 18.5% for understandable, and 4.3% for wrong translation.

Table 8.1 and 8.2 show the evaluation score of the proposed system on 20 text documents. Their average scores show that our system can be produced acceptable results for summarizing an English document to Vietnamese language. The results also indicate that even we used the MT-late-gold method, the results also not in high quality. This was because the performance of MT-software is currently not high. Despite of these problems, the CLTS achieves an acceptable result in comparison with MT-late-gold and outperforms MT-before-lead. In fact, the MT-late-gold is based on the assumption that the output

Table 8.3: The performance result of the CLTS system on a test of 20 text documents obtained from webs.

Methods	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4
MT-late-gold	0.295	0.082	0.028	0.0090
MT-before-lead	0.138	0.029	0.010	0.0020
MT-EBSR	0.197	0.032	0.013	0.0024
CLTS	0.227	0.053	0.015	0.0030

of a text summarization is perfect since we obtained the summary of each original document. This is the reason why the results of CLTS system are slightly smaller than those of MT-late-gold.

Table 8.1 and 8.2 also show the computational times of three methods in which the result of the CLTS system is lower than MT-late-gold, but it is efficiently faster than MT-before-lead.

8.3.2 ROUGE Evaluation

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation [53]. It is a method to automatically determine the quality of a summary by comparing it to other (ideal) summaries created by humans. This measure is computed by counting the number of overlapping words between the computer-generated summary to be evaluated and the ideal summaries created by humans.

$$ROUGE - N = \frac{\sum_{S \in \{\text{referencesummaries}\}} \sum_{gram_n \in s} Count_{match}(gram_n)}{\sum_{S \in \{\text{referencesummaries}\}} \sum_{gram_n \in s} Count(gram_n)} \tag{8.1}$$

Equation (8.1) shows the ROUGE-N score where n stands for the length of the n-gram, $gram_n$, and $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in candidate summary and a set of reference summaries.

For testing, we collected 20 English news articles and they gist meanings in Vietnamese and English from the web (http://www.vnagency.com.vn/). We then experiment our CLTS system on this test set by using the ROUGE evaluation. In addition, we also use our example based sentence reduction technique (EBSR) as described in 7.5 to reduce the Vietnamese long sentence after translating 20 English news sentences. We called the method MT-EBSR.

The evaluation results in Table 8.3 explain that the CLTS system outperforms MT-before-lead in almost every extraction tasks. Table 8.3 also indicates that our second method (MT-EBSR) outperforms MT-before-lead. In comparison with MT-late-gold, the results of CLTS were smaller but the difference is acceptable.

8.4 Conclusion

We have designed a cross language text summarization framework and show its potential when testing on a subset of DUC corpus, Cmlp corpus, and a news article corpus for

summarizing English document to Vietnamese language. Human and ROUGE evaluation both show that the proposed system achieved acceptable results and they are significantly better than that of MT-before-lead. Although our results are good in testing on a small corpus there are some problems to obtain a reliable cross language text summarization system. To make it applicable, we focus on improving the performance of the mono-language text summarization and machine translation tasks further. Currently, the initial results of CLTS system is not high but in the future its result will be improved when the training data corpus is completely revised and enriched more.

We believe that with a larger corpus, these problems can be solved and the system's performance will be further improved.

Chapter 9

Conclusion

In this thesis we have described statistical machine learning for cross language text summarization. We show that the major limitation of previous work on CLTS is likely to treat machine translation and mono-lingual text summarization separately. To overcome this problem, we first propose a new method which allows adapting translation to mono-lingual summarization. We then apply statistical machine learning models to CLTS in order to improve both the performance of text summarization and machine translation.

9.1 Summary of the Contributions

The main contributions of this thesis include:

- *Decomposition of human-written summary sentences*
  Chapter 4 presents a new method of enhancing the accuracy of a decomposition task by using position checking and a semantic measure for each word within a summary document. The proposed model is an extension of the Hidden Markov Model for the human written decomposition problem. Experimental results using DUC data and Telecommunication Corpus shows that the proposed method improves the accuracy of decomposition of human-written summary sentences. Although our generation training data method are suitable as well for experimenting on the corpus of original texts and their abstracts, it still needs human correction for the generated data in order to use for training sentence extraction and sentence reduction task.

- *Co-MEM for Sentence Exaction*
  Chapter 5 discusses the use of unlabeled data to improve the sentence extraction using machine learning, we propose a Co-MEM training algorithm that is a variant of the co-training algorithm based on two different views. Experiments shows that the proposed algorithm improved the conventional algorithms on the sentence extraction task.

- *Probabilistic sentence reduction*
  Chapter 6 investigates a novel application of support vector machines in sentence reduction. Furthermore, we propose a new probabilistic sentence reduction methods based on support vector machine learning and maximum entropy models. In contrast to previous methods, the proposed methods have the ability to produce

multiple best results for a given sentence, which is useful in text summarization because information in full text document can be utilized to summarize the document. Experimental results show that the proposed methods outperform earlier methods in term of sentence reduction accuracy.

- *Machine translation in Cross-language summarization*
  Chapter 7 addresses a new example-based machine translation system based on template translation learning method. The proposed system improves the template translation system in both the learning phase and the translation phase. The learning phase is extended by incorporating linguistic information in order to produce more comprehensive and reliable rules. The translation phase is extended to enhance translation's performances in term of computational times and accuracy by establishing a Hidden Markov Model on a set of template rules that estimates from translation examples. Experiments show that the comprehensive and reliable rules improved translation results. Furthermore, establishing a Hidden Markov Model on a set of template rules dramatically outperforms the original system. The proposed system also incorporated with a rule-based machine translation system with a larger number of translation rules for using in real application. To this end, we introduce an example based sentence reduction method which can achieve a good reduction result without using any syntactic parser.

- *A new Cross-Language Text Summarization System*
  Chapter 8 shows the implementation of the cross language text summarization system for English and Vietnamese language. In which, we have designed a road map and built a framework of a cross language text summarization for any pair of languages. In addition, we show its potential by testing on a small corpus.

9.2 Further Research Direction

We have shown a new approach to cross-language text summarization system using statistical machine learning models. Although our CLTS system shows a promising results, there are still several open problems for future works which can be listed as follows.

- Our generating training data algorithm is mainly applied for a pair of text document and its summary in the same language. So it is essential to adapt it to wherever the abstract and the original document are two different languages.

- Since we treat the reduction phase and the translation phase separately, it would be interesting to find a common kennel for reduction and translation phase in our CLTS system. When we obtain a corpus of long sentences and short sentences, in which a long sentence and a short one are in two different languages, then our reduction model should change to new one so that it can utilize these corpus. In the translation phase, we draws a new perspective for applying statistical machine learning to this domain, but only HMM models are experimented. To support our idea further, using other statistical learning models such as Maximum Entropy Makov Model and Conditional Random Fields seem to be very promising.

- Reformulating the text for other summarization purpose. In this thesis, we have looked at how to reformulate the text from a document into a text summary that

can be read by reader. In future work, we focus on applying our technique to multimedia summarization. Typically, the structure of multimedia and text are very different, therefore discovering new suitable statistical machine learning models for multimedia summarization are interesting works.

- We have designed a cross language text summarization system and show its summarization performance. However, it would be very nice if we could incorporate it with other applications. For example, we can use our CLTS system in digital library or cross language information retrieval.

Bibliography

[1] I. Mani and M. Maybury, "Advances in Automatic Text summarization", The MIT Press, 1999.

[2] E. Neggemeyer et al "Summarizing Information", Springer, Berlin (1998).

[3] E.H. Hovy and C.Y. Lin: "Automated text summarization in summarist", in I. Mani and M. Maybury, editor, Advances in Automated Text Summarization, chapter 8. MIT Press.

[4] C.Y. Lin: "Machine translation for information access across the language barrier: the must systerm", In Machine Translation Summit VII, Stemper, 1999.

[5] W. Ogden, J. Cowie, M. Davis, E. Ludovik, H.M. Salgado, and H. Shin: "Getting information from documents you cannot read: An interactive cross-language and summarization system", In SIGIR/DL Workshop on Multilingual Information Dvsvovery and Acess (MIDAS), August, 1999.

[6] H.H. Chen and C.J. Lin, "A multilingual news summarizer", In Proceeding of the 18th International Conference on Computational Linguistic", pp. 159-165, 2000.

[7] D. Kirk Evans and J.L. Klavans: "A Platform for Multilingual News Sumamrization", Columbia University, Report, Apri, 2003.

[8] A. Leuski, C.Y. Lin, L. Zhou, U. Germann, F.J. Och, and E. Hovy: "Cross-language C*ST*RD: English Access to Hindi Information", ACM Transactions on Asian Language Information Proceesing", Vol. 2, No.3, September, pp 245-269, 2003.

[9] B. Dorr and D. Zajic: "Cross-Language Headline Generation for Hindi", ACM Transactions on Asian Language Information Proceesing", Vol. 2, No.3, September, pp 270-289, 2003.

[10] Jing and R. McKeown: "Cut and Paste text summarization", In Proceeding of the 1st Conference of the North American Chapter of the Association for Computational Linguistic,(2000) pp. 178-185.

[11] H.P. Edmunson: " New methods in automatic extracting", Journal of the Association for Computing Machinery 16(2): 264-285 (1958).

[12] G. Salton, C. Buckley, and A. Singhal: "Automatic analysis theme generation and summarization of machine-readable texts", Science, 264: 1421-1426, June 3, 1994.

[13] M. Osborne: "Using Maximum Entropy for Sentence Extraction", In Proceeding on ACL Workshop on Automatic Summarization (Including DUC 2002), Philadelphia, Pennsylavania, USA, July 11-13, 2002.

[14] K. Knight and D. Marcu, "Summarization beyond sentence extraction: A Probabilistic approach to sentence compression", Artificial Intelligence 139: 91-107, 2002.

[15] G. Grefenstette, "Producing intelligent telegraphic text reduction to provide an audio scanning service for the blind", In Working notes of the AAAI Spring Symposium on Intelligent Text summarization, pp.111-118, 1998.

[16] S.H. Corston-Olivers and W.B. Dolan, "Less is more; eliminating index terms from subordinate clauses", In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistic, pp.349-356, 1999.

[17] H. Jing, "Sentence reduction for automatic text summarization", In Proceeding of the First Annual Meeting of the North American Chapter of the Association for Computational Linguistics NAACL-2000

[18] M.L. Nguyen and S. Horiguchi, "A new sentence reduction Based on Decision tree model", Proceedings of 17th Pacific Asia Conference on Language, Information and Computation, Page 290-297, 2003

[19] M.L. Nguyen and S. Horiguchi, "A sentence reduction Using Syntax Control", Information Retrieval With Asian Language, pp 139-146, 2003.

[20] C.Y. Lin, "Improving Summarization Performance by Sentence Compression — A Pilot Study", Proceedings of the Sixth International Workshop on Information Retrieval with Asian Languages, pp.1-8, 2003.

[21] J.M. Withbrock and O.v. Mital, "Ultra-summarization: A statistical approach to generating highly-Condensed Non-Extractive Summaries", Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGR'99), Poster Secsion, 315-316, Berkeley, CA.

[22] S.Rezler, et al, "Statistical Sentence Condensation using Ambiguti Packing and Stochastic Disambiguation Method for Lexical-Functional Grammar", Proceeding of the Human Language Technology Conference and the 3rd Meeting of the North American Chapter of the Association for Computational Linguistic (HTL-NAACL'03), Edomton, Canada, CA.

[23] I. Cicekli and H.A. Güvenir: "Learning Translation Templates from Bilingual Translation Examples", Applied Intelligence 15: 57-36, 2001.

[24] H.A. Güvenir and I.Cicekli, "Learning translation templates from examples", Information System, vol.23,no.6, pp.353-363, 1998.

[25] H. Jing: "Using Hidden Markov Modeling to Decompose Human-Written Summaries", Computational Linguistic 28: 527-543, 2002.

[26] G. Salton, J. Allan: "Selective text utilization and text traversal", International Journal of Human-Computer Studies, 43: 483-497, 1995.

[27] M. Mitra, A. Singhal, and C. Buckley: "Automatic text summarization by paragraph analysis", In Proceeding of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization, pp 39-46, 1997.

[28] C.D. Paice: "Constructing literature abstracts by computer: techniques and prospects", Information Proceesing Management, 26(1): 171-186. 1990.

[29] K.R. Mckeown and D.R. Radev: "Generating summaries of multiple news articles", In Proceedings of the Eighteenth Annual internal ACM SIGIR Conference on Research and Development in Information Retrieval, pages 74-82, 1995.

[30] J. Morris and G. Hirst, "Lexical cohesion computed by thesaural relations as an indicator of the structure of text", Computational Linguistics, 17(1): 21-48, March. 1991.

[31] R. Barizly and M. Elhadad, "Using lexical chains for text summarization", In proceeding of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization, pp 10-17, 1997.

[32] M. Benbrahim and K. Ahmad: "Text summarization: The role of lexical cohesion analysis", The New Review of Document and Text Management, 1:321-335. 1995.

[33] B. Baldwin and T. Morton: "Coreference-based sumamrization", In Proceeding of the TIPSTER Text Phase III Workshop, Washing-ton, 1998.

[34] D. Marcu: "From discourse structures to text summaries", In Proceedings of ACL/EACL'97 Workshop on Intelligent Text summarization, pp 82-88, Madrid, Spain, 1997.

[35] U. Reimer and U. Hahn: "A formal model of text summarization based on condensation operators of a terminological logic", In Proceedings of the ACL/EACL Workshop on Intelligent Scalbale Text Summarization, pp 97-104, 1997.

[36] J. Kupiec, Jan O. Pedersen, and F. Chen:"A trainable document summarizer", In Research and Development in Information Retrieval", pp 68-73 (1995).

[37] S. Teufel and M. Moens: "Sentence extraction as a classification task", Proceeding ACL/EACL-97 Workshop on Intelligent and Scalable Text Summarization, 1997.

[38] T. Hirao, H. Isozaki, E. Maeda, and Y. Matsumoto: "Extracting important sentences with support vector machines", In Proceeding of COLING 2002, pages 342-348, 2002.

[39] H. Halteren: "New Feature Sets for Summarization by Sentence Extraction", IEEE Intelligent systems, pp.34-42, 2003.

[40] J.M. Conroy et al: "Using HMM and Logis-tic Regression to Generate Extract Summaries for DUC", Proceeding Document Understanding Conference, 2001.

[41] A. Berger and V. Mittal: "OCELOT: A system for summarizing web pages", In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2000.

[42] C.K. Chuah:"Linguistic Processes for Content Condensation in Abstracting Scientific Texts", PhD thesis, Universit'e de Montréal, 2001.

[43] R. Barizlay: "Information Fusion for Multidocument Summarization: Paraphrasing and Generation", Ph.D thesis, Columbia Unviersity, 2003.

[44] M. Banko, V. Mittal, and M. Witbrock: "Headline generation based on statistical translation", In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000), pp 318-325, 2000.

[45] A. Berger and V. Mittal: "Query-relevant summarization using FAQs", In Proceedings of the 38 Annual Meeting of the Association for Computational Linguistic, pp 294-301, 2000.

[46] R. Jing and A. Hauptmann: "Title generation for machine translated documents", In Proceedings of the Seventeenth International Conference on Artifical Intelligence, pp 4-10, 2001.

[47] R. Brandow, K. Mitze, and L.F. Rau: "Automatic condensation of electronic publications by sentence selection", Information Proceesing and Management, 31(5):675-685, 1995.

[48] R.D. Radev, "Generating natural language summaries from multiple on-line source: language resuse and regeneration", Ph.D. thesis, Columbia University.

[49] J. Carbonell, Y. Geng, and J. Goldstein, "Automated query-relevant summarization and diversity-based reranking", Proceeding of the IJCAI-97 Workshop. Sandiego, USA, 1997.

[50] C.Y. Lin: "Summarization evaluation environment". http://www.isi.edu/SEE.

[51] K. Papineni, S. Roukos, T. Ward, and W-J. Zhu: "BLEU: A method for automatic evaluation of machine translation. Reserch Report RC2217, IBM, 2001.

[52] C.Y. Lin and E.H. Hovy: "Manual and automatic evaluation of summaries", In proceedings of the Document Understanding Conference (DUC-02), pp 45-51, 2002.

[53] C.Y.Lin and E.H. Hovy, "Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics", In Proceeding of 2003 Language Technology Conference (HLT-NAACL 2003), Edmonton, Canada. 2003.

[54] http://www-nlpir.nist.gov/projects/duc/guidelines/2004.html.

[55] E. Mjolsness and D. DeCoste: "Machine Learning for Science: State of the Art and Future Prospects", Science vol 293 (14), pp 2051-2055, 2001.

[56] S.E. Levinson, L.R. Rabiner and M.M. Sondhi. 1985. "An introduction to the application of the theory of probabilistic functions of Markov process to automatic speech recognition". Bell System Technical Jounal, 62:1035-1074.

[57] A.J. Viterbi , "Error bounds for convolution codes and an asymptotically optimal decoding algorithm", IEEE Trans on Information Theory.(1967) 260-269.

[58] L.E. Baum and J.A. Eagon. 1967. "An inequality with application to statistical for probabilistic functions of Markov processes and to a model of ecology". Bull. Amer. Math. Soc, 73:360-363.

[59] E.T. Jaynes, "Information Theory and Statistical Mechanics", Physical Review (106), pp.620-630,1957.

[60] A. Berger, S.A. Dell Pietra and V.J. Dell Pietra, "Maximum Entropy Approach To Natural Language Processing", Computational Linguistic, 22(1), 33-71. 1996.

[61] S. Benson and J.J. Moré, " A limitted memory variable matric method for bound contraint minimization", Technical Report ANL/MCS-P909-0901, Argone Natinal Laboratory.

[62] R. Malouf,"A comparison of algorithms for maximum entropy parameter estimation", In Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002). Pages 49-55, 2002.

[63] V. Vapnik, "The Natural of Statistical Learning Theory", New York: Springer-Verlag, 1995.

[64] B. Scholkopf et al, "Comparing Support Vector Machines with Gausian Kernels to Radius Basis Function Classifers", IEEE Trans. Signal Procesing, 45, pp. 2758-2765, 1997.

[65] F. Brown, J.C. Lai and R.L. Mercer: "Aligning sentences in parallel corpora". Proceeding of the 29th Annual of the Association for Computational Linguistic,(1991) pp 169-176.

[66] A. Gale and Keneth , "A Program for aligning sentences in parallel corpora". In Proceeding of the 29th Annual meeting of the Association for Computational Linguistic,(1991) pp.77-184.

[67] D. Marcu, "The automatic construction of large-scale corpora for summarization research". In Proceedings of the 22nd International Conference on Research and Development in Information Retrieval, University of California, Berkeley, (1999).

[68] H. Jing and R. McKeown: "The decomposition of human-written summary sentences", in Proceeding of the 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, (1999) pp. 129-136.

[69] M.L. Nguyen and S. Horiguchi: "An efficient Decomposition of Human - Written Summary Sentence", Proceeding of 9th International Conference on Neural Information Processing,(2002) vol.2, pp. 705-710.

[70] A. Budanitsky and G. Hirst , "Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures", Workshop on WordNet and Other Lexical Resources, Second meeting of the North American Chapter of the Association for Computational Linguistics. Pittsburgh, PA, 2001.

[71] J. Jiang and D. Conrath: "Semantic similarity based on corpus statistics and lexical taxonomy", Proceedings of International Conference on Research in Computational Linguistics, Taiwan, 1997.

[72] G. Hirt and D. St-Onge: "Lexical Chains as representations of context for the detection and correction of malapropisms", Fellbaum pp. 305-332. 1998.

[73] C. Leacock and M. Chodorow: "Combining local context and WordNet similarity for word sense identification", Fellbaum pp. 265-283, 1998.

[74] http://www-nlpir.nist.gov/projects/duc/.

[75] Manber, Udi and Gene Myers, "Suffix arrays: A new method for online-string searches", Proceeding of The first Annual ACM-SIAM Symposium on Discrete Algorithms, pages 319-327.

[76] Aone, Chinatsu, Mary Ellen Okurowski, Jame Gorlinsky, and Bjornar Larsen:"A Trainable summarizer with knowledge acquired from robust NLP techniques", In I. Mani and M. T. Maybury, editors, Advances in Automatic Text Summarization. MIT Press, Cambiridege, pp. 71-80.

[77] P.B. Baxendale: "Man-made index for technical literature- An experiment", IBM Journal of Research and Development, vol. 2(4), pp.354-361 (1958).

[78] A. Blum and T. Mitchell:"Combining labeled and unlabeled data with co-training", In Proceedings of the 11th Annula Conference on Learning Theory", pp 92-100, 1998.

[79] H.P. Luhn, "The automatic creation of literature abstracts", IBM Jouranal of Research Development, vol. 2(2): 1959-165 (1958).

[80] C. Buckely and C. Cardie, "Using empire and smart for high-precision IR and summarization.", In proceeding of the TIPSTER Text Phase III 12-Month Workshop, Sandiego, CA. Octorber.

[81] Stralkowski, Tomek, G. Stein, J. Wang, and B. Wise, "A robust practical text summarizer", In I. Mani and M. T. Maybury editor, Advances in Automatic text summarization. MIT Press, Cambridge, pp. 137-154 (1999).

[82] H. Jing, K. McKeown, R. Barzilay, and M. Elhadad, "Summarization evaluation methods: Experiments and analysis", In Intelligent Text Summarization: Paper from the 1998 AAAI Spring Symposium, Standford, CA, 23-25 March. Technical Report SS-98-06. AAAI Press, pp. 60-68 (1998).

[83] C.Y. Lin:"Training a selection function for extraction", In Proceeding of the Eighteenth ACM conference on Information and Knowldge Management (CKIM), Kansas City, 6 November. ACM, pp. 55-62 (1997).

[84] C.Y. Lin and E. Hovy:"Indentifying topics by position", In Fifth Conference on Applied Natural Language Processing", Association for Computational Lingusitic, 32 March-3 Apirl, pp 55-62 (1999).

[85] D. Yarowsky: "Unsuprvised word sense disambiguation rivaling supervised methods", In Proceedings of the 33rd Annual Meeting of the Association for Compuational Lingustic, pp. 189-196 (1995).

[86] M. Collins and Y. Singer:"Unsupervised model for named entity classification", In Proceedings of the Empirical Methods in NLP Conference", pp. 100-110, 1999.

[87] D. Pierce and C. Cardie: "Limitations of co-training for natural language learning from large datasets", In Proceedings of the Empriical Methods in NLP conferences,Pittsburgh, PA (2001).

[88] A. Sarkar: "Applying co-training methods to statistical parsing", In Proceedings of the EMNLP Conferences, pp. 133-142, Philadelphia, PA (2001).

[89] D. Radev, H. Jing and M. Budzikowska: "Centroid-based summarization of multiple documents: Sentence extraction, utility-based evaluation, and user studies.", In ANLP/NAACK Workshop on Summarization, Seatle, April 2000.

[90] R. Tibshirani, "Classification by pairwise coupling", The Annals of Statistics, 26(1): pp. 451-471, 1998.

[91] E. Charniak, "A Maximum entropy inspired parser", In Proceedings of the first Annual Meeting of the North American Chapter of the Association for Computational Linguistic NAACL, pp.132-139, 2000.

[92] C. Fellbaum, "WordNet: An Electronic Lexical Database", Mit Press(1998).

[93] A. Ratnaparkhi, "A Maximum Entropy Part of Speech Tagger", In Proceeding Conference on Empirical Method in Natural Language Processing , 1996.

[94] A. Borthwick, "A Maximum Entropy Approach to Named Entity Recognition", Ph.D thesis, Computer Science Department, New York University (1999).

[95] R. Koeling, "Chunking with Maximum Entropy Models", Proceeding of CoNLL-200 and LLL-2000, Lisbon, Portugal, pp. 139-141.

[96] Y. Zhou and F. Weng, "A Fast Algorithm for Feature Selection in Conditional Maximum Entropy Modeling",Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, pp.153-159,2003.

[97] C. Macleod and R. Grishman, "COMLEX syntax Reference Manual", Proteus Project, New York University (1995).

[98] J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman, San Mateo, CA (1995).

[99] J. Platt, " Probabilistic outputs for support vector machines and comparison to regularized likelihood methods," in Advances in Large Margin Classifiers, Cambridege, MA: MIT Press, 2000.

[100] T.T. Hastie and R. Tibshirani, "Classification by pairwise coupling", The Annals of Statistics, 26(1): pp. 451-471, 1998.

[101] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines", Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[102] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multi-class support vector machines", IEEE Transactions on Neural Networks, 13, pp. 415-425, (2002).

[103] A. Ratnaparkhi, "A Maximum Entropy Part of Speech Tagger", In Proceeding Conference on Empirical Method in Natural Language Processing , 1996.

[104] A. Borthwick, "A Maximum Entropy Approach to Named Entity Recognition", Ph.D thesis, Computer Science Department, New York University (1999).

[105] R. Koeling, "Chunking with Maximum Entropy Models", Proceeding of CoNLL-200 and LLL-2000, Lisbon, Portugal, pp. 139-141.

[106] Y. Zhou and F. Weng, "A Fast Algorithm for Feature Selection in Conditional Maximum Entropy Modeling",Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, pp.153-159,2003.

[107] M. Nagao: "Framework of a mechanical translation between Japanese and English by analogy principle", Artificial and Human Intelligence, edited by A. Elithorn and R Banerji, NATO publication, North-Holland, Edinburgh, pp. 173–180 (1984).

[108] S. Sato and M. Nagao: "Tword memory-based transation. Proceeding of the 13th International Conference on Computational Linguistic, Helsinki Filand, vol. 3, pp. 247–252 (1990).

[109] E. Sumita and H. Iida: "Experiments and prospects of example-based machine translation", Proceeding of the 29th Annual Meeting of the Association for Computational Linguistics, 185-192.

[110] S. Nirenbug, S. Beale and C. Domashnew: "A full-text experiment in Example-Based Machine Translation." New Methods in Language Proceesing, Studies in Computational Linguistic, Manchester, England.

[111] D.R. Brown, "Transfer-rule induction for example based translation", Proceeding of the Workshop on Example-Based Machine Translation, http://www.eamt.org/summitVIII/papers/brown.pdf.

[112] H. Somers, "Review Article: Example-Based Machine Translation", Machine Translation 14: 113-157,1999.

[113] I. Cicekli and H.A. Guvenir. Learning Translation Rules From A Bilingual Corpus. Proceeding of the 2nd International Conference on New Method in Language Processing, Ankara, Turky, September, pp: 90-97 (1996).

[114] R.D. Brown: "Adding Linguistic Knowledge to a Lexical Example-Based Translation system". Proceeding of the 8th International Conference on Theoretical and Methodological Issue in Machine Translation (TMI 99), Chester, England, 22-32, (1999).

[115] Öz and I. Cicekli: (1998). "Ordering Translation Templates by Assignning Confidence Factors", Proceeding of the 3rd Conference of Association for Machine Translation in the Mericas, Langhorne, PA, pp. 51-61.

[116] F.Sha and F.Pereira (2003). "Shallow Parsing with Conditional Random Fileds", Proceedings of Human Language Technology-NAACL 2003, Edmonton. Canada.

[117] F.S. Chen and J. Goodman, "An emprical study of somoothing techiques for language modeling", Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University, 1998.

[118] A. Monila, "Shallow parsing using Specialized HMM", Journal of Machine learning Research, Vol.2 595-613. (2002).

[119] N.H. Pham, M.L. Nguyen, A.C. Le, N.P. Thai, N.V. Vinh, and H.S. Dam, "LVT: An English Vietnamese Translation System", First National Conference on Fundamental and Applied Research in Information Technology FAIR-03, Hanoi, October 2003.

Publications

Journal Papers

[1] M.L. Nguyen, A. Shimazu, and S. Horiguchi, "A New Template Translation Learning Based on Hidden Markov Modeling", **WSEAS Transactions on Computers**, Issue 1, Volume 3, pp. 256-262, 2004.

[2] M.L. Nguyen, S. Horiguchi, A. Shimazu, T.B. Ho, " Example Based Sentence Reduction Using Hidden Markov Model", to appear, **ACM Transactions on Asian Language Information Processing**, Issue 3, Vol 3, September, 2004.

[3] M.L. Nguyen and S. Horiguchi, "Accuracy Enhancement for the Decomposition of Human-Written Summary", to be published **International Journal of Computer Processing of Oriental Languages (IJCPOL).**

[4] M.L. Nguyen, M. Fukushi, and S. Horiguchi, "A Probabilistic Sentence Reduction Using Maxium Entropy Model", **IEICE Transactions on Information System** (accepted).

[5] M.L. Nguyen and S. Horiguchi, "A New Sentence Reduction Learning Technique Based on Decision tree model", Submitted to **International Journal of Artificial Intelligent Tool (IJAIT)**.

[6] M.L. Nguyen and S. Horiguchi, "A Maximum Entropy Markov Model for the Decomposition of Human-Written Summary", to be submitted, **Information Processing and Managements**.

[7] M.L. Nguyen, A. Shimazu, and S. Horiguchi, "A Chunking Based Example Based Machine Translation System", to be submitted, **Machine Translation**.

[8] M.L. Nguyen, A. Shimazu, S. Horiguchi, T.B. Ho, "Statistical machine learning for sentence reduction", to be submitted, **Computational Intelligence**.

Refereed Conference Papers

[9] M.L. Nguyen, A. Shimazu, S. Horiguchi, T.B. Ho, and M. Fukushi, "Probabilistic Sentence Reduction Using Support Vector Machines", The 20th International Conference on Computational Linguistics COLING 2004, 23-27 August, Geneva, pp. 743-749, 2004.

[10] M.L. Nguyen, A. Shimazu, and S. Horiguchi, "Translation Template Learning Using Hidden Markov Modeling", Proc. of 17th Pacific Asia Conference on Language, Information and Computation, pp. 269-276, 2003.

[11] M.L. Nguyen and S. Horiguchi, "A New Sentence Reduction Based on Decision Tree Model", Proc. of 17th Pacific Asia Conference on Language, Information and Computation, pp. 290-297, 2003.

[12] M.L. Nguyen and S. Horiguchi, "A Sentence Reduction Using Syntax Control", Proc. of 6th Information Retrieval with Asian Language, pp. 139-146, 2003.

[13] M.L. Nguyen and S. Horiguchi, "Sentence Reduction and Query Translation for Cross Language Information Retrieval: An Internet Search Application", Proc. of The International Symposium on Towards Peta-Bit Ultra-Network, pp. 103-107, 2003.

[14] M.L. Nguyen and S. Horiguchi, "An Efficient Decomposition of Human-Written Summary Sentence", Proc. of 9th International Conference on Neural Information Processing, November, Singapore, Vol.2, pp.705-710, 2002

[15] M.L. Nguyen, S. Horiguchi, T.B. Ho, A.C. Le, V.V. Nguyen and P.T. Nguyen, " Sentence Reduction Using Semantic Parsing with Rich Knowledge Base", Proc of Joint Third International Conference on Intelligent Technologies and Third Vietnam-Japan Symposium on Fuzzy Systems and Application, Hanoi, pp. 376-379, 2002.

Non-refereed Workshop Paper

[16] P.H. Nguyen, M.L. Nguyen, A.C. Le, N.P. Thai, N.V. Vinh, and H.S. Dam, "LVT: An English Vietnamese Translation System", First National Conference on Fundamental and Applied Research in Information Technology FAIR-03, Hanoi, October 2003.

[17] M.L. Nguyen and S. Horiguchi, "Approach to scalable statistical text summarization", JAIST Research Report, IS-RR-2002-016, 2002, April. pp.1-16.