Cut-and-Paste Text Summarization

Hongyan Jing

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2001

ABSTRACT

Cut-and-Paste Text Summarization

Hongyan Jing

Automatic text summarization provides a concise summary for a document. In this thesis, we present a *cut-and-paste approach* to addressing the text generation problem in domain-independent, single-document summarization.

We found that professional abstractors often reuse the text in an original document for producing the text in a summary. But rather than simply extracting the original text, as in most existing automatic summarizers, humans often edit the extracted sentences. We call such editing operations "revision operations". Our summarizer simulates two revision operations that are frequently used by humans: sentence reduction and sentence combination. Sentence reduction removes inessential phrases from sentences and sentence combination merges sentences and phrases together. The sentence reduction algorithm we propose relies on multiple sources of knowledge to decide when it is appropriate to delete a phrase from a sentence, including linguistic knowledge, probabilities trained from corpus examples, and context information. The sentence combination module relies on a set of rules to decide how to combine sentences and phrases and when to combine them. Sentence reduction aims to improve the conciseness of generated summaries and sentence combination aims to improve the coherence of generated summaries. We call this approach "cut-and-paste" since it produces summaries

by excerpting and combining sentences and phrases from original documents, unlike the extraction technique which produces summaries by simply extracting sentences or passages.

Our work also includes a Hidden Markov Model based sentence decomposition program which analyzes human-written summaries. The decomposition program identifies where the phrases of a summary originate in the original document, producing an aligned corpus of summaries and articles that we use to train and evaluate the summarizer. We also built a large-scale, reusable lexicon by combining multiple, heterogeneous resources. The lexicon contains lexical, syntactic, and semantic knowledge. It can be used in many applications.

Contents

List of Figures

List of Tables

Acknowledgments

A number of people deserve special thanks for guiding, encouraging, and supporting me to complete this thesis.

My advisor, Kathy McKeown, has been wonderful: she encouraged me to explore on my own while constantly providing insightful advice, she taught me about research, and she patiently corrected my papers even when she had millions of other things waiting. I am most grateful to her.

The other members of my dissertation committee, Luis Gravano, Graeme Hirst, Aravind Joshi, and Judith Klavans, also played an important role in the development of this thesis. I am very grateful to them for their valuable comments at the thesis proposal and the defense and also their feedback on the draft of this dissertation.

I had the opportunity to work with some great people outside of Columbia University. Evelyne Tzoukermann was a wonderful mentor during my summer internship at Bell Laboratories and has been a mentor and friend ever since. Karen Kukich provided me the first summer internship in my graduate school years and is one of the most caring people I have met. Oliviero Stock, Emanuele Pianta, and the other members of the natural language processing group at Istituto per la Ricerca Scientifica e Tecnologica gave me the opportunity to venture out of my main research area. It was a pleasure to work with such a wonderful group of people. I not only learned about machine translation, but also had the most pleasant summer ever. I thank Michael Elhadad and his students at Ben-Gurion University for the inspiring conversations during our collaboration. I also thank Yael Ravin for being on my candidacy exam committee.

I am indebted to the members of the natural language group at Columbia University for their help and support. They participated in my evaluations, they sat through my presentation dry-runs, and they gave me interesting ideas. I want to thank particularly Shimei Pan and James Shaw for helping me every step along the way, Vasileios Hatzivassiloglou and Dragomir Radev for interesting discussions over many years, Pascale Fung and Rebecca Passonneau for helping me in my early graduate student years, and Min-Yen Kan and Barry Schiffman for being great office mates.

Finally, and most importantly, I would like to thank my family and friends for their encouragement and unwavering support over the years spent on this work.

To Mom, Dad, Hongwei, and Ling

Chapter 1

Introduction

The automatic construction of abstracts from the texts of documents has gained increasing interest in recent years. Abstracts can help a reader to grasp the central subject matter of a document without looking at the full document. High-quality text summarization would improve document search and browsing in a variety of contexts. For example:

- *Over the Internet.* With such an overwhelming amount of information on the Internet, retrieving and browsing relevant documents in an efficient manner becomes extremely important to Internet users. Abstracts could help users to quickly judge the relevance of documents, and therefore, not to waste time accessing and browsing documents that are not at all interesting to them.

- *Over Digital Libraries.* As documents become increasingly available in electronic form, the effort of storing and indexing them leads to the construction of digital libraries. To provide abstracts for documents in digital libraries so that the documents can be efficiently organized and retrieved, we need to explore automatic techniques since the number of documents that are to be included in digital libraries is enormous and manually constructing abstracts for all documents is impossible.

- *From hand-held devices.* Due to the limited display space on hand-held devices such as Personal Digital Assistants, document condensation is very desirable.

This chapter is organized as follows. First, we give an overview of automatic text summarization, touching on such matters as types of abstracts, and other aspects that are used by researchers in the summarization community to describe automatic summaries and automatic summarization systems. Section 1.2 gives a critical survey of summarization approaches, informing readers of the state of the art of the field. Section 1.3 is the major part of this introduction, in which we introduce the problem that this thesis focuses on — the generation problem in summarization research. This problem has been relatively neglected by the summarization research community. We reinforce the importance of investigating this problem, and contemplate possible solutions and the most difficult issues involved in the problem. Section 1.4 gives an overview of this thesis, outlining the approach we propose to address the generation problem in summarization.

1.1 An Overview of Automatic Text Summarization

According to the American Heritage Dictionary of the English Language (Fourth Edition, 2000), an *abstract* is *"a statement summarizing the important points of a text"*. Abstracts may contain *indicative* material, helping a reader to decide whether it will be worthwhile to look at the full document. Many abstracts are also *informative*, including material such as main results and conclusions so that a reader can gather the most important information without having to refer to the full document. The indicative function is regarded as essential and the informative function is desirable. In practice, most automatically constructed abstracts contain at least some informative material. Although some researchers state that their aim is to produce indicative summaries while others state that they aim to produce informative summaries, the techniques used are very similar.

Abstracts that are produced manually almost always summarize the important points of the central subject matter of a text; such abstracts are called *generic* abstracts. Many automatic summarization systems can also construct *user-focused* or *query-based* summaries, which do not concentrate on the central topic of a text but contain material that is most important based on a user's particular interests. A user's queries to a search engine or to a question-answering system are often regarded as the indication of the user's particular interests. In practice, many techniques developed for generic summarization are also used for query-based summarization, but with slight modifications.

We might consider a wide variety of text types to construct abstracts from. Much of the reported work has been concerned with producing summaries for newspaper articles or scientific papers. Some summarization systems are *domain-dependent*, meaning that they can only handle texts from a specific domain. Such systems often rely on the knowledge of texts in that domain (for example, text structures, keywords or key phrases) to find the important information. Some systems are *domain-independent*, meaning that they do not have strict restrictions on the domain of documents.

Abstracts might be constructed from a *single document*; they might also be constructed from *multiple documents*. In the latter case, the abstract will summarize the important points in *all* the source documents (in practice, the source documents are often a cluster of documents that report on the same event). When multi-document summarization is concerned, the source documents can be in a single language (*monolingual*), or they can be in different languages (*multilingual* or *translingual*).

The majority of automatic systems produce summaries by extracting informative sentences from documents. We call such summaries *extracts*, meaning that they consist of sentences extracted from the full documents. Summaries that are constructed by extracting sentences from the beginning of documents are called *lead-based summaries*, and are often used in the evaluation experiments as a baseline with which to compare automatically constructed summaries. In quite a few studies, lead-based summaries were actually found to outperform “sophisticated” or “intelligent” summarization systems.

1.2 Previous Approaches

Much of the reported work has been concerned with producing extract-like summaries. Therefore, the key problem addressed by most previous literature is how to identify the most important, salient, or informative sentences in the text.

Earlier work (from 1950s to early 1990s) can be roughly divided into two categories in terms of their approaches: the statistical or surface-clue approach, and the artificial intelligence (AI) approach. The systems that use the first approach try to find clues that could possibly indicate the informativeness of sentences. The clues that have been investigated are many, including the frequency of words, the location of sentences, cue phrases, and the syntactic structure of sentences, among others. Based on these clues, the systems compute the informativeness of each sentence, usually indicating it with a score. The sentences with the highest scores are then regarded as most important and extracted. In contrast to the surface-clue approach, the AI approach tries to understand the meaning of the text using natural language processing techniques. With current language processing technology, deep understanding of a text is possible only if the text is in a specific domain, in which fixed text structures or patterns are often used. Such structures or patterns aid the understanding of the text. Rather than cite individual papers, we refer readers to [Paice, 1990] for a very good overview of the work between the 1950s and 1990.

The Dagstuhl Seminar on *"summarizing text for intelligent communication"* held in Dagstuhl, Germany in 1993 [Endres-Niggemeyer, Hobbs, and Sparck Jones, 1993], can be seen as the beginning of a new phase in summarization research. There has been a tremendous surge of publications and interest since then. The reported work has taken four different paths: *statistical*, *knowledge-based*, *shallow understanding based*, and *hybrid approaches*.

Many *statistical models*, most of which have already been successfully used in other language applications, are applied to summarization. One of the earliest works is by Salton and his colleagues [Salton et al., 1994], who use the Vector Space Model in Information Retrieval to measure the similarities between paragraphs and find important paragraph(s). Another representative work is [Kupiec, Pedersen, and Chen, 1995], which formulates summarization as a statistical classification problem — dividing sentences into two categories: important and unimportant — and deploys the Bayesian classification algorithm for summarization. Recently, [Knight and Marcu, 2000] applies statistical machine translation techniques for compressing sentences. Similar techniques are used for summarizing web pages [Berger and Mittal, 2000]. In our work, we use Hidden Markov Models to decompose human-written summary sentences (see Chapter 3). Other statistical techniques for summarization include position-based [Lin and Hovy, 1997] , and signature word based [Brandow, Mitze, and Rau, 1995].

The statistical approach is fast, robust, scalable, and domain-independent. The disadvantages of this approach include the following: feature selection is experimental, sentence scoring is heuristic and empirical, a large training corpus is needed, and there is a performance upper bound due to no real understanding of the text. A particular model may not possess all of the above properties but only a subset of them.

An alternative to the statistical approach is the *knowledge-based approach*. This approach is often adopted to summarize text in a specific domain. It relies on rich knowledge about the domain to understand the text and decide what should be included in the summaries. Domain knowledge is acquired either through automatic training or through manual encoding. For example, [Paice and Johns, 1993] uses stylistic clues and constructs to identify important concepts in highly structured technical papers. [McKeown and Radev, 1995, Radev, 1999] use the result of information extraction systems as input to construct fluent summaries for a cluster of documents using natural language generation techniques.

The knowledge-based approach uses domain knowledge effectively and adapts to the special requirements of the application. The disadvantages of the approach include the following: it is expensive to port to a new domain, it is unscalable and knowledge-intensive, it only summarizes predefined interests, and it may need large training corpora.

The *shallow understanding based approach* uses shallow linguistic knowledge to understand the text to some degree; understanding is not as deep as in knowledge-based systems but is stronger than that in statistical models. This shallow understanding might depend on lexical cohesion and coherence of the text [Morris and Hirst, 1991, Barzilay and Elhadad, 1997, Benbrahim and Ahmad, 1995, Baldwin and Morton, 1998, Mani, Bloedorn, and Gates, 1998], discourse structure [Marcu, 1997], or communicative functions.

The shallow understanding based approach has more understanding of the text than the statistical method, and is more robust and less expensive than the knowledge-based. The disadvantages of this approach include: it is often fragile, it is not robust enough to deliver good results consistently, and it might be computationally expensive compared to statistical approaches.

In practice, a summarization system might combine and use more than one of the above techniques. We refer to any such system as a *hybrid system*). For example, a primarily shallow understanding based approach might also adopt statistical techniques. The hybrid connectionist-symbolic model [Aretoulaki, 1997] is another example of the hybrid approach; in this case, even more diverse methods are combined.

Evaluation of summarization is a difficult and controversial matter. We defer this topic until Chapter 7, where we present the evaluation of our summarization system.

1.3 The Generation Problem in Summarization

Conceptually, summarization should include at least two processes: first, identifying the most important information that ought to be included in the summary, and second, generate the summary based on the information that has been identified in the first process. We refer the first process as the *understanding* process and the second process as the *generation* process. As we have indicated in section 1.2, much of the previous work in summarization is only concerned with the understanding process, specifically extraction techniques. The generation problem has been a relatively neglected corner in the field of summarization.

1.3.1 The Problems with Extraction

Automatic summaries are typically generated by extracting sentences (occasionally phrases or paragraphs) from the original documents. When sentences are removed from their context and strung together to construct summaries, many problems may arise. Problems with extraction-based summaries include:

(1) **Extraneous phrases**

Extracted sentences can be very long, containing material that does not need to be included in a summary.[1] For instance, the following is a sentence extracted from a newspaper article:

"The five (men) were apprehended along Interstate 95, heading south in vehicles containing an array of gear including paramilitary uniforms, a stun gun, knives, two-way radios and smoke grenades, authorities said."

Instead of including all the details in the original sentence, we might want to include in the summary only the fact that *"The five (men) were apprehended along Interstate 95, authorities said."* Simple extraction, however, cannot remove extraneous information from extracted sentences.

(2) **Dangling pronouns and noun phrases**

Extracted sentences could contain anaphora, definite noun phrases, and logical and rhetorical connectives, which may be unresolved when the sentences are removed from their original context. As a result, the automatically generated summaries might be incoherent, or even worse, incomprehensible. When the above example sentence is extracted without its original context, it is unclear to readers who "the five" (the subject of the sentence) refers to.

(3) **Misleading information**

When sentences are taken out of context and placed one after another in automatic summaries, they may convey meanings that are not at all intended in the original text, presenting misleading or false information. Consider the following example, which consists of two sentences extracted from an editorial on human cloning:

[1] Longer sentences typically have higher statistic weights and are more likely to be extracted by automatic summarizers.

> *"And who will we clone?"*
>
> ** "It should include a high percentage of women, since women are so greatly affected by reproductive issues of all kinds."*

The summary seems to convey that the author of the editorial believes that we should clone a high percentage of women. But from the text in the original article, as shown below, it is clear that the author did not advocate the cloning of women:

> *"And who will we clone? Who will have the resources to order up a replica? Mainly rich white guys — unless some company orders up a brace of Michael Jordans or Tiger Woodses."*
>
> *... ...*
>
> *"It (the committee) should include a high percentage of women, since women are so greatly affected by reproductive issues of all kinds."*

Moreover, extraction is not an option for all types of documents. Take the example of documents in patent databases. Quite often, the documents in patent databases contain sentences that have as many as hundreds of words (especially in patents that describe the mechanical components of a device). In such cases, simply extracting sentences without any condensation will not yield acceptable summaries.

Much of the reported work is concerned with how to extract informative sentences to construct summaries; very little research is concerned with the generation problem in summarization. But given the fact that we have no better summary generation technique than simple extraction and the very negative effects that extraction has on the quality of automatic summaries, it is clearly time to investigate better techniques for generating summaries.

The research in this thesis focuses on the generation problem in summarization. In particular, we are interested in developing techniques that are domain-independent, robust, and scalable. Our research is primarily concerned with single-document summarization instead of multiple documents, and we aim to generate generic summaries instead of query-based summaries.

1.3.2 Traditional Natural Language Generation

One possible solution to the generation problem in summarization is to divide the summarization task into two steps: (1) interpret the text using natural language understanding techniques and represent its meaning in some form of semantic representation, and (2) generate the summary from the semantic representation by using traditional natural language generation techniques. With this approach, the generation problem in summarization can be solved using existing techniques in traditional natural language generation.

While this solution might look plausible in theory, it has many difficulties in practice. First, we are interested in domain-independent summarization, but traditional natural language generation systems work in specific domains [Swartout, 1983, McKeown, 1985, McCoy, 1986, Meteer et al., 1987, Miller and Rennels, 1988, Meteer, 1989, Elhadad, 1992, Robin, 1994, McKeown, Kukich, and Shaw, 1994]. Since there are usually limited types of messages in a specific domain, generation systems can rely on the deep understanding of the semantic input and comprehensive domain knowledge to decide what to say and how to say it. Such a method, however, cannot be scaled up to a general domain.

Second, there is a wide gap between what a language interpretation module currently can provide and the semantic input that a traditional generation system expects.

A generation system requires sophisticated semantic information and extensive domain knowledge, which is impossible to provide by a summarizer that works in a general domain and relies on text analysis tools to extract information.

For the above reasons, we cannot rely only on available generation techniques to tackle the generation problem in summarization, but need to develop new techniques that are particularly suitable for summarization.

1.4 This Thesis

This thesis presents a *cut-and-paste approach* for the generation problem in summarization. On the one hand, this approach allows the reuse of original text to construct summaries; on the other hand, it modifies the selected text so that the sentences or phrases that are selected from disconnected context in the original text can fit together. The research presented in this thesis is related to three research areas: summarization, information retrieval, and lexical resources. The next chapter gives an overview of our cut-and-paste summarization system and it also includes an introduction of the remaining chapters.

Chapter 2

System Overview

In this chapter, we give a high level overview of our cut-and-paste summarization system. We begin by introducing the *cut-and-paste approach* for generating summaries. Then, we present the architecture and major components of our system.

As discussed in the previous chapter, the focus of this dissertation is on the text generation problem in summarization. We propose a *cut-and-paste approach*, which produces summaries by *excerpting and combining sentences and phrases from original documents*. Unlike simple extraction, the cut-and-paste approach edits the extracted sentences and phrases. The cut-and-paste approach reuses the text in the original documents to generate summaries. This is a practice often used by professional abstractors, as we show in Section 2.1. We also identified operations that are frequentely used by professionals for editing the sentences and phrases extracted from the original text into summaries. These operations are explained in details in Section 2.1.2.

2.1 Reusing Document Text for Summary Generation

To develop an automatic system that can generate fluent, concise, and coherent summaries, we first studied how humans write summaries, hoping that the human summarization experience can shed some light on developing an automatic system.

2.1.1 Abstracting Guidelines

We found that abstracting guidelines give inconsistent advice on how a summary should be generated. One school of scholars believes that writers should use their own words rather than heavily rely on the original wording. For example, an early book on abstracting for American high school students states "(use) your own words... Do not keep too close to the words before you" [Thurber, 1924]. In contrast, another study showed that professional abstractors actually rely on cutting and pasting the original text to produce summaries: "Their professional role tells abstractors to avoid inventing anything. They follow the author as closely as possible and reintegrate the most important points of a document in a shorter text" [Endres-Niggemeyer and Neugebauer, 1995, Endres-Niggemeyer et al., 1998]. Some studies are somewhere in between: "summary language may or may not follow that of author's" [Fidel, 1986]. Other guidelines, books, or studies on abstracting [ANSI, 1997, Cremmins, 1982] do not discuss the issue. Generally, we found that the abstracting guidelines we could find in books or other literatures are at a very high level.

2.1.2 Revision Operations

Given that abstracting guidelines are unfortunately not very helpful in building an automatic system, we analyzed by ourselves a set of articles to observe how they were summarized by human abstractors, hoping in this process to find useful techniques used by humans that can perhaps be applied in an automatic system as well. The set of articles we analyzed includes 15 news articles on telecommunications, 5 articles on medical issues, and 10 articles in the legal domain. Although the articles are related to particular domains, they cover a wide range of topics, so they are actually quite general; even the articles in the same domain do not possess the same structure or writing style. The Telecommunications articles were collected using the free daily news service, "Communications-related Headlines", provided by the Benton Foundation (http://www.benton.org). These articles came from various newspapers and their abstracts were written by staff writers at Benton. The medical news articles were collected from *HIV/STD/TB Prevention News Update*, provided by the Center for Disease Control (CDC) (http://www.cdcnpin.org/news/prevnews.htm). The CDC provides synopses of key scientific articles and lay media reports on HIV/AIDS as a public service. The synopses are written by staff writers daily. The articles in the legal domain were from the newspaper *New York Law Journal*. These articles describe court's decisions on law suits and the summaries were written by the editors of the newspaper.

We found that, in the corpus we studied, reusing the text in the original document for producing the text in the summary is an almost universal practice by human abstractors. This is consistent with the finding in [Endres-Niggemeyer and Neugebauer, 1995, Endres-Niggemeyer et al., 1998], which stated that professional abstractors often rely on cutting and pasting the original text to produce summaries.

Based on careful analysis of the human-written summaries, we defined six operations that can be used alone, sequentially, or simultaneously to transform a sentence in an article into a summary sentence in a human-written abstract. We call these *revision operations*. The operations are:

(1) sentence reduction

Remove extraneous phrases from a sentence, as in the following example:[1]

Document Sentence: *When it arrives sometime next year in new TV sets,* the V-chip will give parents a *new and potentially revolutionary* device to block out programs they don' t want their children to see.

Summary Sentence: The V-chip will give parents a device to block out programs they don' t want their children to see.

The deleted material can be at any granularity: a word, a phrase, or a clause. Multiple components can be removed from a single sentence.

(2) sentence combination

Merge material from a few sentences. It is typically used together with sentence reduction, as illustrated in the following example, which also uses paraphrasing:

Document Sentence 1: But it also raises serious questions about the privacy of such highly personal information *wafting about the digital world.*

Document Sentence 2: The issue thus fits squarely into the broader debate about privacy and security on the Internet, *whether it involves protecting credit card number or keeping children from offensive information.*

Summary Sentence: But it also raises the issue of privacy of such personal information *and* this issue hits the nail on the head in the broader debate about privacy and security on the Internet.

[1] All the examples in this section were taken from the 30 articles we analyzed; the summary sentences are real examples found in human-written abstracts.

(3) syntactic transformation

Change the syntactic structure of a sentence. In both sentence reduction and combination, syntactic transformations may be involved. For example, the subject of a sentence may be moved from the end of the sentence to the front.

(4) lexical paraphrasing

Replace phrases with their paraphrases. For instance, in the previous example, the summary sentences substituted *fit squarely into* with a more picturesque description *hit the nail on the head*.

(5) generalization or specification

Replace phrases or clauses with more general or specific descriptions. Examples of generalization and specification include:

> Generalization: *"a proposed new law that would require Web publishers to obtain parental consent before collecting personal information from children"* → *"legislation to protect children's privacy on-line"*
>
> Specification: *"the White House's top drug official"* → *"Gen. Barry R. McCaffrey, the White House's top drug official"*

(6) reordering

Change the order of extracted sentences. For instance, place an ending sentence in an article at the beginning of an abstract.

In human-written abstracts, there are, of course, sentences that are not based on cut-and-paste, but were completely written from scratch. The criteria we used in distinguishing a sentence based on cut-and-paste and a sentence written from scratch are the following: if more than half of words in a summary sentence are composed by phrases borrowed from the original document, then the sentence is considered as constructed by cut-and-paste; otherwise, it is considered written from scratch. There are also other cut-and-paste operations not listed here due to their infrequent occurrence. Note that it is often the case that multiple revision operations are involved in order to produce a single summary sentence.

2.2 The Cut-and-Paste Summarization

2.2.1 The Cut-and-Paste Approach

We decided to use a *cut-and-paste approach* to addressing the text generation problem in domain-independent, single-document summarization. This approach goes beyond simple extraction, to the level of simulating the revision operations to edit the extracted sentences. *In particular, we simulate two revision operations: sentence reduction and sentence combination.* Since this approach generates summaries by extracting and combining sentences and phrases from the original text, we call it the *cut-and-paste approach*.

While extraction-based approaches mostly operate at the sentence level, and occasionally at the paragraph or clause level, the cut-and-paste approach often involves extracting and combining phrases. This cut-and-paste approach addresses only the text generation problem in summarization; it does not address the document understanding problem in summarization.

2.2.2 The Conservative Strategy in Editing

Automatic summaries can potentially mislead the users. The extraction-based summaries mislead the users when the sentences that are taken out of context and placed one after another in an automatic summary happen to convey meanings that are not at all intended in the original text. We have shown such an example in Chapter 1.

By automatic editing of reused text, the cut-and-paste summarization can even potentially mislead the user in a way that a simple list of extracted sentences cannot, for the edits are transparent to the users. In order to avoid misleading users, we adopted a conservative strategy in editing the reused text. The system reduces or combines sentences and phrases only when it is quite confident in doing so. For cases that it is less confident at, no editing is performed. The goal of sentence reduction and sentence combination is to resolve the problems caused by sentence extraction, such as the existence of inessential phrases in extracted sentences and the incoherence caused by dangling anaphora. We want to avoid as much as possible introducing new problems while trying to solve the problems caused by sentence extraction. It would be useful to inform users in advance that the summaries are automatically generated and the reused text has been edited.

2.2.3 System Architecture

We now present the architecture of our cut-and-paste summarization system, focusing on the relations between different modules and the role of each module.

Figure 2.1 shows the system architecture. This is a domain-independent, single-document summarization system; therefore, the input to the system is a single document from any domain, and the output is a summary of the input document. There are two stages in the summarization process. The first stage, *extraction*, identifies the most important sentences in the input document. The second stage, *cut-and-paste generation*, is the focus of our research. The cut-and-paste generation component edits the extracted sentences, particularly by simulating two revision operations: sentence reduction and sentence combination. The cut-and-paste generation is designed to be a portable component; it can be integrated with any extraction-based, single-document summarization system, serving as its generation component.

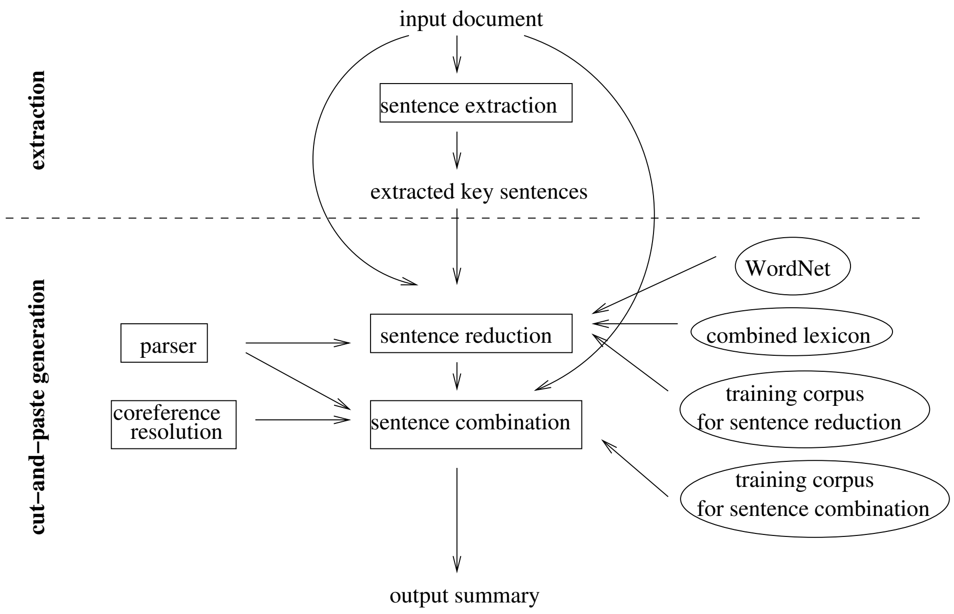


Figure 2.1: System architecture.

The *sentence extraction* module identifies the most important sentences in the input document. The input of the extraction module is the input document, and the output is a list of key sentences that have been selected by the extraction module. Our algorithm for sentence selection is primarily based on the lexical relations between words, and it also incorporates some other types of information, such as statistical measures used in information retrieval, sentence positions and cue phrases.

The *sentence reduction* module removes inessential phrases from an extracted key sentence, resulting in a shortened version of the extracted sentence.[2] This shortened text can be used directly in a summary, or it can be fed to the sentence combination module to be merged with other sentences. Reduction can significantly improve the conciseness of automatic summaries. Our reduction program uses multiple sources of knowledge to decide which phrases in an extracted sentence should be removed, includ-

[2]It is actually also possible that the reduction program decides no phrase in a sentence should be removed, thus the result of reduction is the same as the input.

ing lexical information, syntactic information from linguistic databases, and statistical information from corpus.

The reduction module acquires its lexical information from the WordNet lexical database [Miller et al., 1990]. It also uses syntactic knowledge from a large-scale lexicon that we have constructed by combining multiple resources. A training corpus, which consists of example sentences that were constructed by humans using sentence reduction, is also used by the reduction module. The module also uses a syntactic parser (the ESG parser licensed from IBM [McCord, 1990]).

The *sentence combination* module merges the resulting sentences from sentence reduction with other phrases or reduced sentences together as coherent sentences. The rule-based combination module can apply different combination operations based on the phrases and sentences to be combined.

The combination module uses a training corpus that consists of example sentences that were constructed by humans using sentence combination. Besides the syntactic parser, it also uses a coreference resolution system (the coreference module in Deep Read, a reading comprehension system licensed from the MITRE Corporation [Hirschman et al., 1999]).

The *combined lexicon* used by the sentence reduction module was constructed by merging multiple, large-scale, heterogeneous linguistic resources, including the WordNet lexical database [Miller et al., 1990], the COMLEX syntax dictionary [Grishman, Macleod, and Meyers, 1994], English Verb Classes and Alternations (EVCA) [Levin, 1993], and the Brown Corpus tagged with WordNet senses [Miller et al., 1993]. It provides necessary syntactic and semantic knowledge for the reduction operation. The lexicon can also be used in many other applications, such as traditional natural language generation, word sense disambiguation, and machine translation.

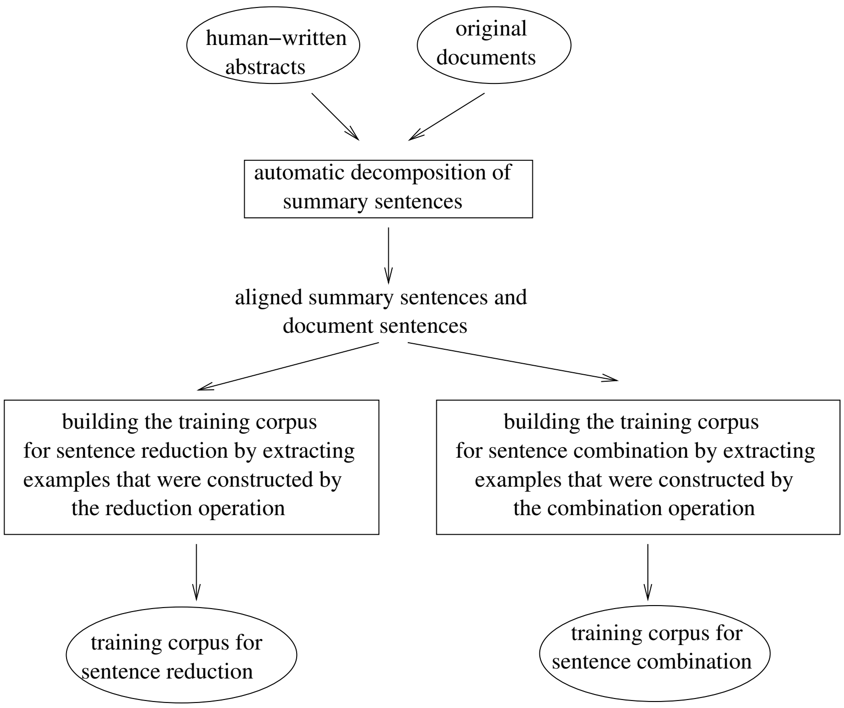


Figure 2.2: Building training corpora for sentence reduction and combination.

The training corpora for sentence reduction and sentence combination are constructed by an automatic summary sentence decomposition program. The process for building the training corpora is illustrated in Figure 2.2.

The *decomposition module* is an automatic program for analyzing human-written abstracts. The statistically based decomposition program can identify where the phrases of a summary originate in the original document, producing an aligned corpus of summaries and articles that we then used to train and evaluate the sentence reduction and sentence combination module.

The remainder of this dissertation is organized as follows. We begin in Chapter 3 with decomposing human-written abstracts. In Chapter 4, we present the sentence reduction algorithm. In Chapter 5, we describe the sentence combination method. In Chapter 6, we show the large-scale lexicon combined from multiple resources. In Chapter 7, we investigate the evaluation and portability issues. Finally, we conclude in Chapter 8 with a summary of research contributions and a number of suggestions for future work.

Chapter 3

Decomposition of Human-Written Summary Sentences

We stated in the previous chapter that professional abstractors often reuse the text in the original documents to produce summaries. The task of *summary sentence decomposition* is to deduce whether a summary sentence is constructed by reusing the original text and identifying the reused phrases. To be more specific, we define the decomposition problem as follows: *Given a human-written summary sentence, the decomposition program needs to answer three questions: (1) Is this summary sentence constructed by reusing the text in the original document?; (2) If so, what phrases in the sentence come from the original document?; (3) And where in the document do the phrases come from?*

The benefits of solving the decomposition problem are two-fold. First, large corpora for training and evaluating our cut-and-paste summarizer can be built from the decomposition result. By linking human-written summaries with original texts, we can mark exactly what phrases humans cut from the original document and how the phrases were pasted together to produce the summary. By doing it automatically, we can afford to mark up a large set of documents, therefore providing valuable training and testing data sets for our system. Second, the decomposition result also provides large corpora for extraction-based summarizers. By aligning summary sentences with original document sentences, we can automatically annotate the most important sentences in an input document, therefore constructing large corpora for training and evaluating the extraction-based summarizers.

While decomposition is useful, it is also difficult. The phrases coming from the original document can be at any granularity, from a single word to a complicated verb phrase to a complete sentence. Therefore, identifying the boundary of phrases is a complex issue. Determining the origin of a phrase is also hard since the component may occur multiple times in the document in slightly different forms. Moreover, multiple revision operations may have been performed on the reused text. As a result, the resulting summary sentence can be significantly different from the document sentences it comes from. All these factors add to the difficulty of the decomposition problem.

In the reminder of this chapter, we first present our Hidden Markov Model solution to the decomposition problem. Section 3.1 discusses the design of the Hidden Markov Model for decomposition purpose and the characteristics of the model. Section 3.2 describe a number of evaluation experiments. In Section 3.3, we show the applications of decomposition. In Section 3.4, we compare with related work.

3.1 Using Hidden Markov Model for Decomposition

We proposed a Hidden Markov Model [Baum, 1972] solution to the decomposition problem. There are three steps in this process. First, we formulate the decomposition problem to an equivalent problem; that is, for each word in a summary sentence, we find a document position that it most likely comes from. This is an important step since only after this transformation are we able to apply the Hidden Markov Model to solve the problem. Second, we build the Hidden Markov Model based on a set of general heuristic rules that we have observed from the text reusing practice of humans. This is actually quite unconventional in applications that use Hidden Markov Models since Hidden Markov Models usually require a training corpus to compute transition probabilities, but we believe it is appropriate in our particular application. The evaluations show that this unconventional Hidden Markov Model is effective for decomposition. In the last step, a dynamic programming technique, the Viterbi algorithm [Viterbi, 1967], is used to efficiently find the most likely document position for each word in a summary sentence and finally find the best decomposition for a summary sentence.

3.1.1 Formulating the problem

We first mathematically formulate the summary sentence decomposition problem. An input summary sentence can be represented as a word sequence: $(I_1, \cdots, I_N)$, where I_1 is the first word of the sentence and I_N is the last word. The position of a word in a document can be uniquely represented by the sentence position and the word position within the sentence: *(SNUM, WNUM)*. For example, (4, 8) uniquely refers to the 8th word in the 4th sentence. Multiple occurrences of a word in the document can be represented by a set of word positions: $\{(SNUM_1, WNUM_1), \cdots, (SNUM_m, WNUM_m)\}$.

Using the above notation, we formulate the decomposition problem as follows: *Given a word sequence* $(I_1, \cdots, I_N)$ *and the positions* $\{(SNUM_1, WNUM_1), \cdots, (SNUM_m, WNUM_m)\}$ *for each word in the sequence, determine the most likely document position for each word in the sequence.*

Through this formulation, we transform the difficult tasks of identifying component boundaries and determining component origins into the problem of finding a most likely document position for each word. As shown in Figure 3.1, when each word in the summary sequence chooses a position, we get a sequence of positions. For example, ((0,21), (2,40), (2,41), (0,31)) is the position sequence we get when every summary word chooses its first occurrence of the same word in the document. ((0,26), (2,40), (2,41), (0,31)) is another position sequence. Every time a summary word chooses a different position, we get a different position sequence. The word "the" in the sequence occurs 44 times in the document, "communication" occurs once, "subcommittee" occurs twice, and "of" occurs 22 times. For this 4-word sequence, there are a total of 1,936 ($44 \times 1 \times 2 \times 22$) possible position sequences.[1] Morphological analysis or stemming can be performed to associate morphologically related words, but it is optional.

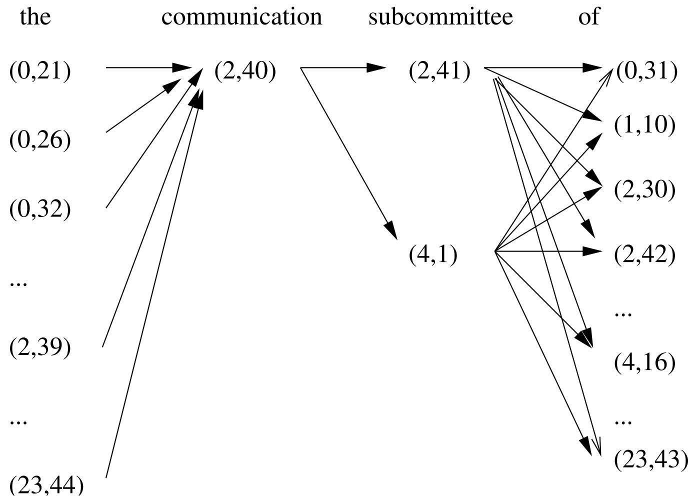


Figure 3.1: The sequences of positions in summary sentence decomposition.

Finding a most likely document position for each word is equivalent to finding the most likely position sequence among all possible position sequences. For the example in Figure 3.1, as we can see, the most likely position sequence should be ((2,39), (2,40),

[1]Given an N-word sequence $(I_1, \cdots, I_N)$, supposing I_i occurs F_i times in the document, for $i = 1..N$, then the total number of possible position sequences is $F_1 \times F_2 \times \cdots \times F_N$.

(2,41), (2,42)); that is, the fragment comes from document sentence 2 and its position within the sentence is word number 39 to word number 41. However, how can we automatically find this sequence among 1,936 possible sequences?

3.1.2 Hidden Markov Model (HMM)

Exactly what document position a word comes from depends on the positions of the words surrounding it. We simply the problem using the bigram model and assume that the probability a word comes from a certain position in the document only depends on the word directly before it in the sequence. Suppose I_i and I_{i+1} are two adjacent words in a summary sentence and I_i is before I_{i+1}. We use *PROB (I_{i+1} = (S_2, W_2) | I_i = (S_1, W_1))* to represent the probability that I_{i+1} comes from sentence number S_2 and word number W_2 of the document when I_i comes from sentence number S_1 and word number W_1.

To decompose a summary sentence, we must consider how humans are likely to generate it; we draw here revision operations we noted in Section 2.1. There are two general heuristic rules we can safely assume: first, humans are more likely to cut phrases than cut single, isolated words; second, humans are more likely to combine nearby sentences into a single sentence than combine sentences that are far apart. These two rules are our guidance in the decomposition process.

We translate the heuristic rules into the bigram probability *PROB (I_{i+1} = (S_2, W_2) | I_i = (S_1, W_1))*, where I_i, I_{i+1} represent two adjacent words in the input summary sentence, as noted earlier. The probability is abbreviated as *PROB($I_{i+1}|I_i$)* in the following discussion. The values of *PROB($I_{i+1}|I_i$)* are assigned in the following manner:

- IF (($S_1 = S_2$) and ($W_1 = W_2 - 1$)) (i.e., the words are in two adjacent positions in the document), THEN *PROB($I_{i+1}|I_i$)* is assigned the maximal value P1. For example, *PROB((subcommittee = (2,41)* $|$ *communications = (2,40))* in the example of Figure 3.1 will be assigned the maximal value. *(Rule: Two adjacent words in a summary are most likely to come from two adjacent words in the document.)*

- IF (($S_1 = S_2$) and ($W_1 < W_2 - 1$)), THEN *PROB(*$I_{i+1}|I_i$*)* is assigned the second highest value P2. For example, *PROB(of = (4,16)* $|$ *subcommittee = (4,1))* will be assigned a high probability. *(Rule: Adjacent words in a summary are highly likely to come from the same sentence in the document, retaining their relative precedent relation, as in the case of sentence reduction. This rule can be further refined by adding restrictions on distance between words.)*

- IF (($S_1 = S_2$) and ($W_1 > W_2$)), THEN *PROB(*$I_{i+1}|I_i$*)* is assigned the third highest value P3. For example, *PROB(of = (2,30)* $|$ *subcommittee = (2,41)). (Rule: Adjacent words in a summary can come from the same sentence in the document but change their relative order. For example, a subject can be moved from the end of the sentence to the front, as in syntactic transformation.)*

- IF $(S_2 - CONST < S_1 < S_2)$, THEN *PROB(*$I_{i+1}|I_i$*)* is assigned the fourth highest value P4. For example, *PROB(of = (3,5)* $|$ *subcommittee = (2,41)). (Rule: Adjacent words in a summary can come from nearby sentences in the document and retain their relative order, such as in sentence combination. CONST is a small constant such as 3 or 5.)*

- IF $(S_2 < S_1 < S2 + CONST)$, THEN *PROB(*$I_{i+1}|I_i$*)* is assigned the fifth highest value P5. For example, *PROB(of = (1,10)* $|$ *subcommittee = (2,41)). (Rule: Adjacent words in a summary can come from nearby sentences in the document but reverse their relative orders.)*

- IF $(|S_2 - S_1| >= CONST)$, THEN *PROB($I_{i+1}|I_i$)* is assigned a small value P6. For example, *PROB(of = (23,43) | subcommittee = (2,41)). (Rule: Adjacent words in a summary are not very likely to come from sentences far apart.)*

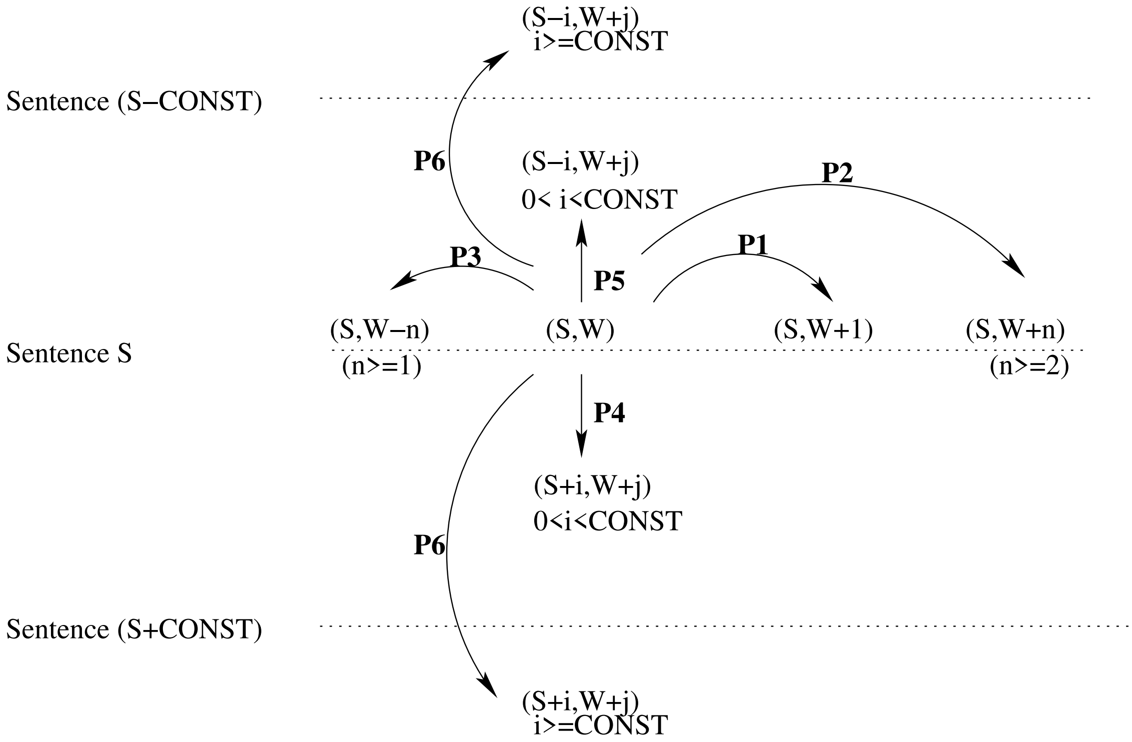


Figure 3.2: Assigning transition probabilities in the Hidden Markov Model.

Figure 3.2 shows a graphical representation of the above rules for assigning bigram probabilities. The nodes in the figure represent possible positions in the document, and the edges output the probability of going from one node to another. These bigram probabilities are used to find the most likely position sequence in the next step. Assigning values to P1-P6 is experimental. In our experiment, the maximal value is assigned 1 and others are assigned evenly decreasing values 0.9, 0.8 and so on. These values, however, can be adjusted for different corpora.

We consider Figure 3.2 as a very abstract representation of a Hidden Markov Model for decomposition. Each word position in the figure represents a state in the

Hidden Markov Model. For example, (S, W) is a state, $(S, W+1)$ is another state. Note here that (S, W) and $(S, W+1)$ are relative values; the S and W in the state (S, W) have different values based on the particular word position we are considering. However, this relative model can be easily transformed to an *absolute* model. We can substitute (S, W) with every possible word position in the document and add transition probabilities between every pair of positions. In Section 3.1.6, we give a formal description of our Hidden Markov Model for decomposition, in particular, defining the basic elements of the model using the terminology in formal Hidden Markov Model terms.

3.1.3 The Viterbi Algorithm

To find the most likely sequence we must find a sequence of positions that maximizes the probability ***PROB***$(I_1, \cdots, I_N)$. Using the bigram model, this probability can be approximated as:

$PROB(I_1, \cdots, I_N) = \prod_{i=0}^{N-1} PROB(I_{i+1}|I_i)$

PROB($I_{i+1}|I_i$***)*** has been assigned as shown earlier. Therefore, we have all the information needed to solve the problem. We use the Viterbi Algorithm [Viterbi, 1967] to find the most likely sequence. For an N-word sequence, supposing each word occurs M times in the document, the Viterbi Algorithm is guaranteed to find the most likely sequence using $k \times N \times M^2$ steps, for some constant k, compared to M^N for the brute force search algorithm.

The Viterbi Algorithm finds the most likely sequence incrementally. It first finds the most likely sequence for $(I_1 I_2)$, for each possible position of I_2. This information is then used to compute the most likely sequence for $(I_1 I_2 I_3)$, for each possible position of I_3. The process repeats until all the words in the sequence have been considered.

We slightly revised the Viterbi algorithm for our application. In the initialization step, equal chance is assumed for each possible document position for the first word in the sequence. In the iteration step, we take special measures to handle the case when a summary word does not appear in the document (thus has an empty position list). We mark the word as non-existent in the original document and continue the computation as if it had not appeared in the sequence.

3.1.4 Post-Editing

After the phrases have been identified, the program does *post-editing* to cancel mismatchings, annulling the results returned by the Viterbi program for some stop words and isolated content words in a summary sentence. The mismatchings are caused by the fact that the Viterbi program assigns each word in the input sequence a position in the document, as long as the word appears in the document at least once. The program deals with two types of mismatchings: wrong assignment of document positions for inserted stop words in a summary sentence and wrong assignment of document positions for isolated content words in a summary sentence. To correct the first type of mismatching, if any document sentence contributes only stop words for the summary, the matching is canceled since the stop words are more likely to be inserted by humans rather than coming from the original document. To correct the second type of mismatching, if a document sentence provides only a single non-stop word, we also cancel such matching since humans rarely cut single words from the original text to generate a summary sentence – they often cut phrases.

3.1.5 An Example

To demonstrate the program, we show an example from beginning to end. The sample summary sentence is as follows:

The input summary sentence (also shown in Figure 3.3):

Arthur B. Sackler, vice president for law and public policy of Time Warner Inc. and a member of the Direct Marketing Association, told the communications subcommittee of the Senate Commerce Committee that legislation to protect children's privacy online could destroy the spontaneous nature that makes the Internet unique.

We first index the document, listing for each word its possible positions in the document.[2] Stemming, a procedure to reduce morphologically related words to the same stem, can be performed before indexing, although it is not used in this example. Augmenting each word with its possible document positions, we therefore have the input for the Viterbi program, as shown below:

Input to the Viterbi Program (words and their possible document positions):

arthur	:	1,0			
b	:	1,1			
sackler	:	1,2	2,34	...	15,6
...					
the	:	0,21	0,26	...	23,44
internet	:	0,27	1,39	...	18,16
unique	:	0,28			

For this 48-word sentence, there are a total of 5.08×10^{27} possible position sequences. Using the bigram probabilities as assigned in Section 3.1, we run the Viterbi Program to find the most likely position sequence. After every word is assigned a most likely document position, we mark the components in the sentence by conjoining words

[2]The original document contains 25 sentences and 727 words in total.

coming from adjacent document positions.

Summary Sentence:

(F0:S1 **arthur b sackler vice president for law and public policy of time warner inc**) (F1:S-1 *and*) (F2:S0 **a member of the direct marketing association told**) (F3:S2 **the communications subcommittee of the senate commerce committee**) (F4:S-1 *that legislation*) (F5:S1**to protect**) (F6:S4 **children' s**) (F7:S4 **privacy**) (F8:S4 **online**) (F9:S0 **could destroy the spontaneous nature that makes the internet unique**)

Source Document Sentences:

Sentence 0: a proposed new law that would require web publishers to obtain parental consent before collecting personal information from children **(F9 could destroy the spontaneous nature that makes the internet unique) (F2 a member of the direct marketing association told)** a senate panel thursday

Sentence 1: **(F0 arthur b sackler vice president for law and public policy of time warner inc)** said the association supported efforts **(F5 to protect)** children online but he urged lawmakers to find some middle ground that also allows for interactivity on the internet

Sentence 2: for example a child's e-mail address is necessary in order to respond to inquiries such as updates on mark mcguire's and sammy sosa's home run figures this year or updates of an online magazine sackler said in testimony to **(F3 the communications subcommittee of the senate commerce committee)**

Sentence 4: the subcommittee is considering the **(F6 children's) (F8 online) (F7 privacy)** protection act which was drafted on the recommendation of the federal trade commission

Figure 3.3: Sample output of the decomposition program.

Figure 3.3 shows the final result for the sample input summary sentence. The components in the summary are tagged as *(FNUM:SNUM actual-text)*, where *FNUM* is the sequential number of the component and *SNUM* is the number of the document sentence where the component comes from. SNUM = -1 means that the component does not come from the original document. The borrowed components are tagged as *(FNUM actual-text)* in the document sentences.

In this example, the program correctly concluded that the summary sentence was constructed by reusing the original text, and it identified the four document sentences from which the summary sentence was combined; it correctly divided the summary sentence into phrases and pinpointed the exact document origin of each phrase. In this example, the phrases that were borrowed from the document range from a single word to long clauses. Certain borrowed phrases were also syntactically transformed. Despite these, the program successfully decomposed the sentence.

3.1.6 Formal Description of Our Hidden Markov Model

This subsection is intended for readers who have an interest in Hidden Markov Models. It can be skipped without affecting the understanding of the rest of the chapter.

Introduction to HMM[3]

Initially introduced and studied in the late 1960s and early 1970s by Baum and his colleagues [Baum and Petrie, 1966, Baum, 1972], statistical methods of Markov source or hidden Markov modeling have become increasingly popular in recent years. Since the models are very rich in mathematical structure, they can form the theoretical basis for use in a wide range of applications. They particularly work very well in practice for several important applications, including speech recognition.

To understand HMMs, one needs to first understand weighted finite-state automata. A weighted automaton augments a finite automaton by associating each arc with a probability, indicating how likely that path is taken. For a weighted automaton, the observation probability is either 1 or 0 (that is, a state can either generates a given symbol or it cannot). However, in an HMM, the observation probability can take on any value from 0 to 1. It also allows a set of observation symbols separate from the state set.

[3] The material in this section is based on the tutorial on Hidden Markov Models by Rabiner [Rabiner, 1989].

In Rabiner's tutorial, a Hidden Markov Model is characterized by the following:

(1) N, the number of states in the model. We denote the individual states as $S = \{S_1, S_2, \cdots, S_N\}$ and the state at time t as q_t. The states can represent different things in different applications. For example, a state can represent a phone in an HMM for speech recognition. It represents a document position in our HMM for decomposition.

(2) M, the number of distinct observation symbols per state. The observation symbols correspond to the physical output of the system being modeled. We denote the individual symbols as $V = \{V_1, V_2, \cdots, V_M\}$. The observation symbols can represent spectral features vectors in an HMM for speech recognition. In our model for decomposition, the observation symbols represent the words in a document.

(3) The state transition probability distribution $A = \{a_{ij}\}$ where a_{ij} represents the probability of transitioning from state i to state j:

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i], \qquad 1 \leq i, j \leq N.$$

(4) The observation symbol probability distribution in state j, $B = \{b_j(k)\}$, where $b_j(k)$ represents the probability of an observation V_k being generated from a state j:

$$b_j(k) = P[V_k \text{ at } t | q_t = S_j], \qquad 1 \leq j \leq N, 1 \leq k \leq M.$$

(5) The initial state distribution $\pi = \{\pi_i\}$ where π_i represents the probability that the HMM will start in state i:

$$\pi_i = P[q_1 = S_i], \qquad 1 \leq i \leq N.$$

Given appropriate values of N, M, A, B, and π, the HMM can be used as a generator to produce an observation sequence

$$O = O_1 O_2 \cdots O_T$$

where each observation O_t is one of the symbols from V and T is the number of observations in the sequence.

"A complete specification of an HMM requires specification of two model parameters (N and M), specification of observation symbols, and the specification of the three probability measures A, B, and π."

Description of Our Hidden Markov Model for Decomposition.

We have said in Section 3.1.2 that Figure 3.2 is an abstract representation of a relative HMM and that it can be transformed into an absolute model. Here, we illustrate how the absolute model can be created. Then we give a formal description of the model.

Suppose there are only two sentences in the original document and each sentence has two words, for simplicity. From the relative model in Figure 3.2, we can build an absolute model as shown in Figure 3.4.

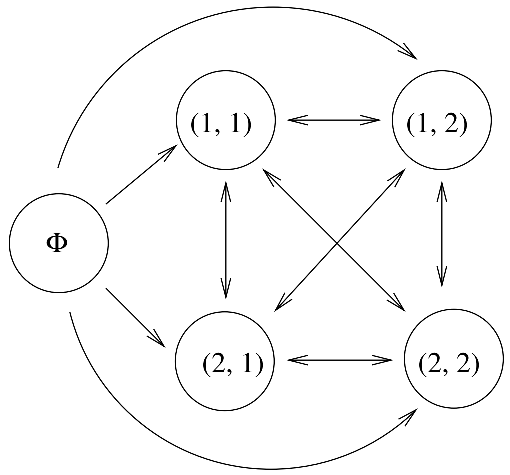


Figure 3.4: Example of the absolute Hidden Markov Model.

In the absolute model, there are four states, (1,1), (1,2), (2,1), and (2,2), each corresponding to a word position. Each state has only one observation symbol – the word in that position. Each state is interconnected with the other states in the model. The transition probabilities can be assigned following the rules shown in Figure 3.2. In this case, however, we need to normalize the values of $\{P_1, P_2, \cdots, P_6\}$ so that for each state the sum of the transition probabilities is 1, which is a basic requirement for an HMM. This normalization is needed in theory in order to conform our relative model to a formal model, but in practice it is not needed in the decomposition process since it does not affect the final result. The initial state distribution is uniform; that is, the initial state, labeled as Φ in Figure 3.4, has an equal chance to reach any state in the model.

We give a formal description of our HMM for decomposition as follows. For each original document, we can build an absolute model based on the relative model in Figure 3.2. In the absolute model, each state corresponds to a word position, and each word position corresponds to a state. The observation symbol set includes all the words in the document, and the observation symbol probabilities are defined as: $P(W_i|P_i) = 1$ if word W_i is in position P_i, and $P(W_i|P_i) = 0$ if word W_i is not in position P_i. The transition probabilities $P(P_j|P_i)$ are defined as we described in Figure 3.2, with every word position linked to every other word position, and state initial probabilities are uniform as we mentioned. This Markov model is hidden because one symbol sequence can correspond to many state sequences.[4]

3.2 Experimental Evaluations

We carried out three experiments to evaluate the decomposition module. In the first experiment, we evaluated decomposition in a task called summary alignment. This measures how successful the decomposition program can align sentences in the summary with the document sentences that are semantically equivalent to them. In the second experiment, we asked humans to judge whether the decomposition results are correct.

[4] We thank Jianying Hu from Avaya Research for an interesting discussion on HMMs. This paragraph mostly comes from a personal communication from her.

Compared to the first experiment, this is a more direct evaluation, and it also uses a larger collection of documents. The third experiment deals with the portability of the program.

The corpus used in the first experiment consists of 10 documents from Ziff-Davis corpus, which contains articles that are related to computer products. The corpus used in the second experiment consists of 50 documents that are related to telecommunication issues, provided by the Benton Foundation. The corpus used in the third experiment consists of legal documents on court cases, provided by the Westlaw Company.

3.2.1 Summary Alignment

In this experiment, we tested the decomposition program in a *summary alignment task*. The goal of the summary alignment task is to find sentences in the document that are semantically equivalent to the sentences in the summary. For this kind of alignment, we use human-written abstracts, which are abundant, to automatically derive their equivalent extracts, thus building large training and testing corpora for extraction-based summarizers.

We used a small collection of 10 documents, collected by Marcu [Marcu, 1999]. Marcu presented these 10 documents together with their human-written summaries from the Ziff-Davis corpus, which is a collection of newspaper articles announcing computer products, to 14 human judges. The human judges were instructed to extract sentences from the original document that are semantically equivalent to the summary sentences. Sentences selected by the majority of human judges were collected to build an *extract* of the document. This extract based on majority opinion of human judges is used as the gold standard in our evaluation.

Our program can provide a set of relevant document sentences for each summary sentence, as shown in Figure 3.3. Taking the union of the selected sentences, we can build an *automatic extract* for the document. We compared this automatic extract with the gold standard extract based on the majority of human judgments, using the recall/precision measure computed as follows:

$$\text{Precision} = \frac{\text{\# of sentences in the automatic extract and also in the gold standard extract}}{\text{total \# of sentences in the automatic extract}}$$

$$\text{Recall} = \frac{\text{\# of sentences in the automatic extract and also in the gold standard extract}}{\text{total \# of sentences in the gold standard extract}}$$

$$\text{F-measure} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$$

The program achieved an average 81.5% precision, 78.5% recall, and 79.1% F-measure for 10 documents. In comparison, the average performance of 14 human judges is 88.8% precision, 84.4% recall, and 85.7% F-measure. The detailed result for each document is shown in Table 3.1.

Docno	*Prec*	*Recall*	*F-measure*
ZF109-601-903	0.67	0.67	0.67
ZF109-685-555	0.75	1	0.86
ZF109-631-813	1	1	1
ZF109-712-593	0.86	0.55	0.67
ZF109-645-951	1	1	1
ZF109-714-915	0.56	0.64	0.6
ZF109-662-269	0.79	0.79	0.79
ZF109-715-629	0.67	0.67	0.67
ZF109-666-869	0.86	0.55	0.67
ZF109-754-223	1	1	1
Average	0.815	0.785	0.791

Table 3.1: Evaluation of decomposition program using the Ziff-Davis corpus.

Further analysis indicates that there are two types of errors by the program. The first is that the program missed finding semantically equivalent sentences that have very different wordings. For example, it failed to find the correspondence between the summary sentence "*Running Higgins is much easier than installing it*" and the document sentence "*The program is very easy to use, although the installation procedure is somewhat complex*". This is not really an "error" since the program is not designed to find such paraphrases. For sentence decomposition, the program only needs to indicate that the summary sentence is not produced by cutting and pasting text from the original document. The program correctly indicated it by returning no matching sentence.

The second problem is that the program may identify a non-relevant document sentence as relevant if it contains some common words with the summary sentence. This typically occurs when a summary sentence is not constructed by cutting and pasting text from the document but does share some words with certain document sentences. Our post-editing steps are designed to cancel such false matchings although we can not remove them completely.

It is worth noting that the extract based on human judgments, which is considered the gold standard in this evaluation, is not perfect. For example, suppose there are two document sentences that express the same information (i.e., they are semantic paraphrases), and all the human subjects consider this information important enough to be included in the summary, but half of the human subjects selected one of the sentences and half of the subjects selected the other sentence, then both sentences will be included in the extract although they are semantic paraphrases. This is exactly what happened in the extract of document ZF109-601-903. The program picked up only one of the paraphrases. However, this correct decision is penalized in the evaluation due to the mistake in the gold standard.

The program won perfect scores for three out of ten documents. We checked the three summaries and found that their texts were largely produced by cut-and-paste, compared to other summaries that might have sentences written by humans completely from scratch. This indicates that when only the decomposition task is considered, the algorithm performs very well.

3.2.2 Human Judgments of Decomposition Results

Since the first experiment does not assess the program's performance for the decomposition task directly, we conducted another experiment to evaluate the correctness of the decomposition result produced by the system. First, we selected 50 summaries from a telecommunication corpus we collected and ran the decomposition program on the documents. The telecommunication corpus contains documents on telecommunication related issues. They are articles from different newspapers and cover a wide range of topics related to telecommunication.

A human subject was asked to read the decomposition results to judge whether they are correct. The program's answer is considered correct when all three questions we posed in the decomposition problem are correctly answered. As we stated at the beginning of Section 4, the decomposition program needs to answer the following 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) And where in the document do the phrases come from?. There are a total of 305 sentences in 50 summaries. 18 (6.2%) sentences were wrongly decomposed, so we achieved an accuracy of 93.8%. Most of the errors occur when a summary sentence is not constructed by cutting and pasting but has many overlapping words with certain sentence in the document. The accuracy rate here is much higher than the precision and recall results in the first experiment. An important factor for this is that here we do not require the program to find the semantically equivalent document sentence(s) if a summary sentence uses very different wordings.

3.2.3 Portability

In the third and final evaluation, we tested the program on documents in the legal domain. This is a joint experiment with the Westlaw Company. Westlaw provides lawyers with documents on court cases. Such documents start with a "synopsis" of the case, written by attorneys. It is then followed by "headnotes", which are points of law that are also written by attorneys and are summarized from the discussions. The last part is the discussion in its entirety, called "opinion".

The task here is to match each headnote entry with the corresponding text in the opinion. When lawyers study a legal case document, they can not only see the important points of law, but also where these points of law are discussed in the opinion. Westlaw is interested in highlighting such an alignment between headnotes and opinions on a screen when a user, typically an attorney or an assistant doing research on a legal case, browses the document using Westlaw's legal document toolkit.

We applied our decomposition program to this task. We made no change to our HMM used on the telecommunication corpus in the second experiment; even the parameters were not adjusted. A sample decomposition result is shown in Figure 3.5. Similar to the notation used in Figure 3.3, the phrases in the headnote are tagged as *(FNUM:SNUM actual-text)*, where *FNUM* is the sequential number of the phrase and *SNUM* is the number of the document sentence where the phrase comes from. SNUM = -1 means that the phrase does not come from the original document. The borrowed phrases are tagged as *(FNUM actual-text)* in the opinion. Note that in this example we ignored the difference of the determiners ("a", "the", etc.) in the phrases, so the summary phrase "a motion for issuance of a peremptory writ" is marked the same as the document phrase “the motion for the issuance of the peremptory writ”.

HEADNOTE:

(F0:S0 **A motion for issuance of a peremptory writ**) (F1:S-1 *of mandamus*) (F2:S0 **notwithstanding**) (F3:S0 **the return**) (F4:S1 **operates as an admission by relator of truth of facts well pleaded**), (F5:S1 **but claims that in law the return presents no sufficient**) (F6:S-1 *ground*) (F7:S1 **why relief sought**) (F8:S1 **should not be granted**).

OPINION:

Sentence 0: As to the effect to be given **(F0 the motion for the issuance of the peremptory writ) (F3 the return)** of the respondents **(F2 notwithstanding)**, it is well to state at the outset that under our decided cases such a motion stands as the equivalent of a demurrer to a pleading in a law action.

Sentence 1: It **(F4 operates as an admission by the relator of the truth of the facts well pleaded)** by the respondent **(F5 but claims that in law the return presents no sufficient)** reason **(F7 why the relief sought)** in the alternative writ **(F8 should not be granted)**.

Figure 3.5: Sample output of legal document decomposition.

We received 11 headnotes from Westlaw and examined the decomposition results for all of them. The program found the correct source document sentences and identified the correct origins of the phrases for every headnote.

In summary, we performed three experiments in three different domains — computer, telecommunication news, and legal – and in each case achieved good results with no change or minimal parameter adjustment to the Hidden Markov Model. For the computer related articles, the program found semantically equivalent summary sentences and document sentences at an average of 81.5% precision and 78.5% recall. For the telecommunication articles, 93.8% of the decomposition results were considered correct by human judgment. For the legal documents, the program found correct document sentences and identified the correct origins of the phrases for every headnote we received. This demonstrates that the decomposition approach we have proposed is portable. The reason for this portability, we believe, is that the heuristic rules that we used to build the Hidden Markov Model are indeed very general, and they remain true for different humans abstractors involved and for articles from different domains.

3.3 Applications of Decomposition Results

Usage in our cut-and-paste based summarization system. We have used the decomposition results to build corpora for training and evaluating the sentence reduction and combination modules in our system. Using the decomposition program, we were able to collect a corpus of summary sentences constructed by humans using reduction operations. This corpus of reduction-based, human-written summary sentences was then used to train as well as evaluate our automatic sentence reduction module. Similarly, we collected a corpus of combination-based summary sentences, which reveals interesting techniques that humans use frequently to paste fragments in the original document into a coherent and informative sentence.

Usage in other summarization systems. As we have shown, in the summary alignment experiment in the evaluations, decomposition can automatically build extracts using the abstracts that have been written by humans. These extracts can be used to build large training and testing corpora for extraction-based summarizers.

Other use. The decomposition result can be also used in other applications. For example, in the experiment we did with Westlaw (see Section 3.2.3), linking summaries and original documents can potentially improve the user interface, helping users to browse and find relations between the text.

Corpus study. Using the decomposition program, we analyzed 300 human-written summaries of news articles. We collected the summaries from a free news service. The news articles come from various sections of a number of newspapers and cover broad topics. 300 summaries contain 1,642 sentences in total, ranging from 2 to 21 sentences per summary. The results show that 315 sentences (19%) do not have matching sentences in the document, 686 sentences (42%) match to a single sentence in the document, 592 sentences (36%) match to 2 or 3 sentences in the document, and only 49 sentences (3%) match to more than 3 sentences in the document. These results suggest that a significant portion (81%) of summary sentences produced by humans are based on cutting and pasting.

3.4 Related Work

Researchers have previously tried to align sentences in a summary with the sentences in the document, mostly by manual effort [Edmundson, 1968, Teufel and Moens, 1997, Kupiec, Pedersen, and Chen, 1995]. Given the cost of this manual annotation process, only small collections of text were annotated. Decomposition provides a way to perform this alignment automatically, building large corpora for summarization research.

[Marcu, 1999] presented an approach for automatic construction of large-scale corpora for summarization research, which essentially is an algorithm for aligning summary sentences with the semantically equivalent sentences in the document. It adopted an IR-based approach, coupled with discourse processing. Although our decomposition also aims to link summaries with the original documents, there are major differences between the two. While Marcu's algorithm operates at the sentence or clause level, our decomposition program deals with phrases, which are at various granularities and could be anything from a word to a complicated phrase to a complete sentence. Furthermore, the approaches used by the two systems are obviously distinct. Marcu's approach first breaks sentences into clauses, then uses rhetorical structures to decide what clauses should be considered, and finally uses an IR-based similarity measure to decide which clauses in the document are similar to those in the human-written abstracts. Our HMM solution first builds the HMM and then uses dynamic programming technique to find the optimal answer. Marcu reported the performance of 77.45%, 80.06%, and 78.15% for precision, recall, and F-measure respectively when the system was evaluated at the sentence level in the summary alignment task that we described in Section 3.2.1. When tested on the same set of test documents and for the same task, our system has an average of 81.5% precision, 78.5% recall, and 79.1% F-measure, as shown in Figure 3.1.

We transformed the decomposition problem to the problem of finding the most likely document position for each word in the summary, which is in some sense similar to the problem of aligning parallel bilingual corpora [Brown, Lai, and Mercer, 1991, Gale and Church, 1991]. While they align sentences in a parallel bilingual corpus, we align phrases in a summary with phrases in a document. [Brown, Lai, and Mercer, 1991] also used Hidden Markov Model in their solution to bilingual corpora alignment. The main difference between their model and our model include our choice of features and how the transition probabilities are assigned. The corpora alignment uses sentence length as a feature, and we use word position as a feature. While the corpora alignment has a training corpus so it can compute the transition probabilities based on the training data, our alignment techniques do not need annotated training data and we assigned the transition probabilities based on general heuristic rules we observed.

3.5 Conclusion

In this chapter, we defined the problem of decomposing a human-written summary sentence and proposed a Hidden Markov Model solution to the problem. The decomposition program can automatically determine whether a summary sentence is constructed by reusing text from the original document; it can accurately recognize phrases in a sentence despite the wide variety of their granularities; it can also pinpoint the exact origin in the document for a phrase. The algorithm is fast and straightforward. It does not need other tools such as a tagger or parser as preprocessor. It does not have complex processing steps. The evaluations show that the program performs very well for the decomposition task. The results from decomposition are used to build training and testing corpora for sentence reduction and sentence combination.

Chapter 4

Sentence Reduction

The decomposition program described in the previous chapter is used to build training corpora for the sentence reduction module that we describe in this chapter. Decomposition aligns summary sentences with document sentences, producing for each summary sentence a list of document sentences that have been used to generate that summary sentence. From the decomposition results, we can build a corpus for the sentence reduction module by selecting the summary sentences that were generated by removing phrases from document sentences. The corpus contains ¡summary sentence, document sentence¿ pairs. Each summary sentence is marked with the information of phrase origins that is provided by the decomposition output.

In a cut-and-paste summarization system, the role of sentence reduction is to perform the editing operation of removing extraneous phrases from an extracted sentence. It can remove material at any granularity: a word, a prepositional phrase, a noun phrase, a verb phrase, a gerund, a to-infinitive, or a clause. We use the term "phrase" here to refer to any of the above sentence components that can be removed in the reduction process.

Reduction improves the conciseness of automatically generated summaries, making it brief and on target. It can also improve the coherence of generated summaries, since extraneous phrases can potentially introduce incoherence if not removed. We collected 500 sentences and the corresponding reduced sentences written by humans (see Section 5.1.1 for the description of the corpus), and found that humans reduced the length of these 500 sentences by 44.2% on average (calculated in terms of number of words in the sentences). This indicates that a good sentence reduction system can improve the conciseness of generated summaries significantly.

We implemented an automatic sentence reduction system. Input to the reduction system includes extracted sentences, as well as the original document. Output of reduction are reduced forms of the extracted sentences, which can either be used to produce summaries directly, or be merged with other sentences. The reduction system uses multiple sources of knowledge to make reduction decisions: it uses the syntactic knowledge from a large-scale lexicon we have constructed to try to guarantee the syntactic correctness of the reduced sentence; it uses the context in which the sentence appears to determine the phrases that are of local focus so that they will not be deleted during reduction; and it uses statistics computed from a corpus of examples produced by humans to decide how likely a certain phrase is removed by humans.

In Section 4.1, we describe the sentence reduction algorithm in detail. Then, we introduce the evaluation scheme we have designed for assessing the performance of the system and present the evaluation results. In Section 4.3, we discuss other issues that relate to our reduction system. In particular, we discuss the extension of the program to query-based summarization and the factors that affect the performance of the system. In Section 4.4, we compare with related work.

4.1 Sentence Reduction Based on Multiple Sources of Knowledge

Our goal of sentence reduction is to “reduce without major loss”; that is, we want to remove as many extraneous phrases as possible from an extracted sentence so that it can be concise, but without detracting from the main idea the sentence conveys. Ideally, we want to remove a phrase from an extracted sentence *only if it is irrelevant to the main topic*. To achieve this, the system relies on multiple sources of knowledge to make reduction decisions. The reduction module can decide on its own an optimal reduction length for a sentence; it can also adjust its decisions based on the compression ratio requested by the user. We first introduce the resources used by the reduction module and then describe the reduction algorithm.

4.1.1 The Resources

The reduction program uses the following four resources:

(1) The corpus. One of the key features of our reduction module is that it uses a corpus consisting of original sentences and their corresponding reduced forms written by humans for training and testing purposes. This corpus was created using the decomposition program that we presented in the previous chapter. The human-written abstracts were collected from the free daily news service “Communications-related Headlines”, provided by the Benton Foundation (http://www.benton.org). The articles in the corpus are news reports on telecommunication related issues, but they cover a wide range of topics, such as law, labor, and company mergers.

(2) The lexicon. The system uses a large-scale lexicon we combined from multiple resources to identify the obligatory arguments of verb phrases. The resources that were combined include the COMLEX syntactic dictionary [Macleod and Grishman, 1995], English Verb Classes and Alternations [Levin, 1993], the WordNet lexical database [Miller et al., 1990], and the Brown Corpus tagged with WordNet senses [Miller et al., 1993]. The lexicon includes syntactic, lexical, and semantic information for over 5,000 verbs. The construction of the lexicon and its usage in natural language generation and other applications are presented in Chapter 6.

(3) The WordNet lexical database. WordNet [Miller et al., 1990] is the largest lexical database to date. It provides many types of lexical relations between words, including synonymy, antonymy, meronymy, entailment (e.g., *eat* → *chew*), or causation (e.g., *kill* → *die*). These lexical links are used to identify important phrases in the local context.

(4) The syntactic parser. We use the English Slot Grammar (ESG) parser developed at IBM [McCord, 1990] to analyze the syntactic structure of an input sentence. The ESG parser not only annotates the syntactic category of a phrase (e.g., "np" or "vp"), it also annotates the grammatical role of a phrase (e.g., "subject" or "object").

4.1.2 The Algorithm

There are five steps in the reduction program:

Step 1: Syntactic Parsing.

We first parse the input sentence using the ESG parser and produce the sentence parse tree. The operations in all other steps are performed based on this parse tree. Each following step annotates each node in the parse tree with additional information, such as syntactic or context importance, which is used later to determine which phrases (they are represented as subtrees in a parse tree) can be considered extraneous and thus removed. A discussion on the effect of parsing errors on the performance of the system is in Section 4.3.3.

Step 2: Grammar Checking.

In this step, we determine which components of a sentence *must not* be deleted to keep the sentence grammatical. To do this, we traverse the parse tree produced in the first step in top-down order and mark, for each node in the parse tree, which of its children are grammatically obligatory. We use two sources of knowledge for this purpose. One source includes simple, linguistic-based rules that use the grammatical role structure produced by the ESG parser. For instance, for a sentence, the main verb, the subject, and the object(s) are essential if they exist, but a prepositional phrase is not; for a noun phrase, the head noun is essential, but an adjective modifier of the head noun is not. The other source we rely on is the large-scale lexicon we described earlier. The information in the lexicon is used to mark the obligatory arguments of verb phrases. For example, for the verb "convince", the lexicon has the following entry:

convince

sense 1:	NP-PP :PVAL ("of")
	NP-TO-INF-OC
sense 2:	NP

This entry indicates that the verb "convince" can be followed by a noun phrase and a prepositional phrase starting with the preposition "of" (e.g., *"he convinced me of his innocence"*). It can also be followed by a noun phrase and a to-infinitive phrase (e.g., *"he convinced me to go to the party"*). This information prevents the system from deleting the "of" prepositional phrase or the to-infinitive that is part of the verb phrase.

At the end of this step, each node in the parse tree — including both leaf nodes and intermediate nodes — is annotated with a value indicating whether it is grammatically obligatory. Note that whether a node is obligatory is relative to its parent node only. For example, whether a determiner is obligatory is relative to the noun phrase it is in; whether a prepositional phrase is obligatory is relative to the sentence or the phrase it is in.

Step 3: Computing Contextual Importance.

In this step, the system decides which components in the sentence are most related to the main topic being discussed. To measure the importance of a phrase in the local context, the system relies on lexical links between words. The hypothesis is that the more connected a word is with other words in the local context, the more likely it is to be the focus of the local context. We link the words in the extracted sentence with words in its local context, if they are repetitions, morphologically related, or linked in WordNet through one of the lexical relations. The system then computes an importance score for each word in the extracted sentence, based on the number of links it has with other words and the types of links. The formula for computing the context importance score for a word w is as follows:

$$ContextWeight(w) = \sum_{i=1}^{9}(L_i \times NUM_i(w))$$

Here, i represents the different types of lexical relations the system considered, including repetition, inflectional relation, derivational relation, and the lexical relations from WordNet. We assigned a weight to each type of lexical relation, represented by L_i in the formula. Relations such as repetition or inflectional relation are considered more important and are assigned higher weights, while relations such as hypernymy are considered less important and assigned lower weights. In Section 4.3.3, we discuss how the weights were experimentally assigned. $NUM_i(w)$ in the formula represents the number of a particular type of lexical links the word w has with words in the local context. Here, the local context includes five sentences before and after the target sentence.

After an importance score is computed for each word, each phrase in the sentence gets a score by adding up the scores of its children nodes in the parse tree. This score indicates how important the phrase is in the local context.

Step 4: Learning from Examples by Humans.

In this step, the program uses the corpus probabilities that are computed beforehand using a training corpus and stored in a table to determine how likely a phrase is removed by human professionals.

The training corpus contains sentences reduced by human professionals and their corresponding original sentences, produced by the decomposition program. The system first parses the sentences in the corpus using the ESG parser. It then marks which subtrees in these parse trees (i.e., phrases in the sentences) were removed by humans, based on the decomposition result. Using this corpus of marked parse trees, we can compute how likely it is that a subtree is removed from its parent node. For example, we can compute the probability that the "when" temporal clause is removed when the main verb is "give", represented as $Prob($"when-clause is removed"|"v=give"$)$, or the probability that the to-infinitive modifier of the head noun "device" is removed, represented as $Prob($"to-infinitive modifier is removed"|"n=device"$)$. These probabilities are computed using Bayes's rule. For example, the probability that the "when" temporal clause is removed when the main verb is "give", $Prob($"when-clause is removed"|"v=give"$)$, is computed as the product of $Prob($"v=give"|"when-clause is removed"$)$ (i.e., the probability that the main verb is "give" when the "when" clause is removed) and $Prob($"when-clause is removed"$)$ (i.e., the probability that the "when" clause is removed), divided by $Prob($"v=give"$)$ (i.e., the probability that the main verb is "give"):

$$Prob(\text{“when-clause is removed”}|\text{“v=give”}) =$$
$$\frac{Prob(\text{“v=give”}|\text{“when-clause is removed”}) \times Prob(\text{“when-clause is removed”})}{Prob(\text{“v=give”})}$$

Besides computing the probability that a phrase is removed, we also compute two other types of probabilities: the probability that a phrase is reduced (i.e., the phrase is not removed as a whole, but some components in the phrase are removed), and the probability that a phrase is unchanged at all (i.e., neither removed nor reduced). These two types of probabilities are computed in a way similar to the above. The probability that the “when” temporal clause is reduced when the main verb is “give”, represented as $Prob$(“when-clause is reduced”|“v=give”), is computed as follows:

$$Prob(\text{“when-clause is reduced”}|\text{“v=give”}) =$$
$$\frac{Prob(\text{“v=give”}|\text{“when-clause is reduced”}) \times Prob(\text{“when-clause is reduced”})}{Prob(\text{“v=give”})}$$

Similarly, the probability that the “when” temporal clause is unchanged when the main verb is “give”, represented as $Prob$(“when-clause is unchanged”|“v=give”), is computed as follows:

$$Prob(\text{“when-clause is unchanged”}|\text{“v=give”}) =$$
$$\frac{Prob(\text{“v=give”}|\text{“when-clause is unchanged”}) \times Prob(\text{“when-clause is unchanged”})}{Prob(\text{“v=give”})}$$

These corpus probabilities help us capture human practice. For example, for sentences like “The agency reported that ...”, “The other source says that ...”, “The new study suggests that ...", the that-clause following the say-verb (i.e., report, say, and suggest) in each sentence is very rarely changed at all by professionals. The system can capture this human practice, since the probability that *that-clause* of the verb *say* or *report* being unchanged at all will be relatively high, which will help the system to avoid removing components in the that-clause.

Note that these probabilities are computed off-line. At running time, the system only loads such information and annotates the parse tree which these probabilities which indicate the likelihood that a subtree is removed, reduced, or unchanged by humans.

Step 5: Making the Final Decision.

The final reduction decisions are based on the results from all the earlier steps. To decide which phrases to remove, the system traverses the sentence parse tree, which now has been annotated with different types of information from earlier steps, in top-down order and decides which subtrees should be removed, reduced or unchanged. A subtree (i.e., a phrase) is removed only if it is not grammatically obligatory, not the focus of the local context (indicated by a low importance score), and has a reasonable probability of being removed by humans. The program can reduce the sentences into different lengths by adjusting the thresholds it uses for importance scores and phrase removal probabilities.

Figure 4.1 shows sample output of the reduction program. The reduced sentences produced by humans are also provided for comparison.

4.2 Evaluations of the Reduction Module

To evaluate the reduction program, we used summary sentences that were generated by human professionals as gold standard, and assessed the output of the system against the

Example 1:

Original sentence : *When it arrives sometime next year in new TV sets,* **the V-chip will give parents a new and potentially revolutionary device to block out programs they don't want their children to see.**

Reduction program: The V-chip will give parents a new and potentially revolutionary device to block out programs they don't want their children to see.

Professionals : The V-chip will give parents a device to block out programs they don't want their children to see.

Example 2:

Original sentence : **Som and Hoffman's creation would allow broadcasters to insert multiple ratings into a show,** *enabling the V-chip to filter out racy or violent material but leave unexceptional portions of a show alone.*

Reduction Program: Som and Hoffman's creation would allow broadcasters to insert multiple ratings into a show.

Professionals : Som and Hoffman's creation would allow broadcasters to insert multiple ratings into a show. *(the same as the result by the reduction program)*

Figure 4.1: Sample output of sentence reduction program.

gold standard. We do not believe that there is a single correct way to reduce a sentence. Just like there is no *unique* summary that is "ideal" for a document, there is no unique reduction output that is "ideal" for a sentence. A sentence can be reduced in many ways, and different human subjects may reduce a sentence differently. But for the evaluation purpose, we assume that the output by humans is ideal.

4.2.1 The Evaluation Scheme

We define a measure called *success rate* to evaluate the performance of our sentence reduction program. The success rate computes the percentage of the system's reduction decisions that agree with those of humans.

The *success rate* is computed as follows. The reduction process can be considered as a series of decision-making processes along the edges of a sentence parse tree. At each node of the parse tree, both the human and the program make a decision whether to remove the node or to keep it. If a node is removed, the subtree with that node as the root is removed as a whole, thus no decisions are needed for the descendants of the removed node. If the node is kept, we consider that node as the root and repeat this process.

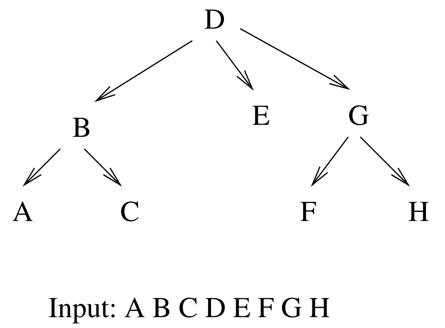


Figure 4.2: Sample sentence and parse tree.

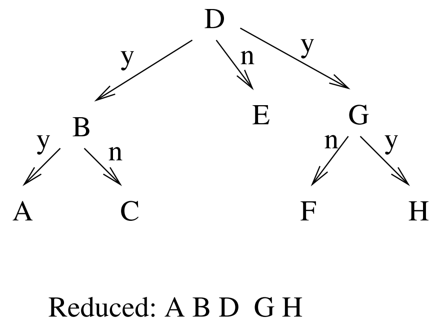
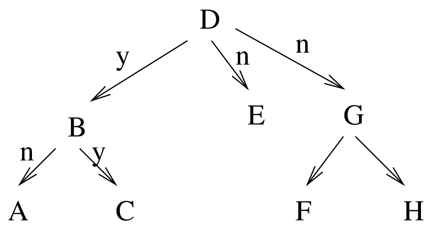


Figure 4.3: Reduced form by a human.

Suppose we have an input sentence *(ABCDEFGH)*, which has the parse tree shown in Figure 4.2. The parse tree representation we used in this figure is consistent with that of ESG. It is a dependency tree with each non-terminal node represents the head word of a phrase. Suppose a human reduces the sentence to *(ABDGH)*, which



Reduced: B C D

Figure 4.4: Reduced form by the program.

can be translated to a series of decisions made along edges in the sentence parse tree as shown in Figure 4.3. The symbol "y" along an edge means the node it points to will be kept, and "n" means the node will be removed. Suppose the program reduces the sentence to *(BCD)*, which can be translated similarly to the annotated tree shown in Figure 4.4.

We can see that along five edges (they are D→B, D→E, D→G, B→A, B→C), both the human and the program have made decisions. Two out of the five decisions agree (they are D→B and D→E), so the success rate is 2/5 (40%). The *success rate* is defined as:

$$\text{success rate} = \frac{\text{\# of edges along which the human and the program made the same decision}}{\text{total \# of edges along which both the human and the program made decisions}}$$

Note that the edges for which no decision needs to be made by the human or the program because they are covered by other decisions (e.g., G→F and G→F in Figure 4.3 and Figure 4.4) are not considered in the computation of success rate, since there is no agreement issue in such cases.

Another alternative to the success rate measure we proposed above for evaluating the reduction results is the recall/precision measure. We can presumably use the reduced sentences generated by human professionals as gold standards and compute recall and precision based on the number of words overlapping between the gold standards and the output by the program. The main problem with this alternative is that the recall and precision results will be greatly affected by the number of words in the sentence and the number of words in the removed phrases. For example, whether the program and the human agree on the removal of a relative clause that contains a large number of words will dramatically change the recall and precision result, while their agreement on a shorter phrase may not change the result as much. That is the reason why we decided against using the recall/precision measure but designed the success rate measure, which counts the reduction decision on any phrase, no matter its length, as one vote and treats each reduction decision equally.

4.2.2 Evaluation Results

In the evaluation, we used 400 sentences in the corpus to compute the probabilities that a phrase is removed, reduced, or unchanged. We tested the program on the remaining 100 sentences.

Using five-fold cross validation (i.e., choose 100 different sentences for testing each time and repeat the experiment five times), the program achieved an average success rate of 81.3%. If we consider the baseline as removing all the prepositional phrases, clauses, to-infinitives and gerunds, the baseline performance is 43.2%.

We also computed the success rate of the program's decisions on particular types of phrases. For the decisions on removing or keeping a clause, the system has a success rate of 78.1%; for the decisions on removing or keeping a to-infinitive, the system has a success rate of 85.2%. We found out that the system has a low success rate on removing adjectives of noun phrases or removing adverbs of a sentence or a verb phrase. One reason for this is that our probability model can hardly capture the dependencies between a particular adjective and the head noun since the training corpus is not large enough, while the other sources of information, including grammar or context information, provide little evidence on whether an adjective or an adverb should be removed. Given that whether or not an adjective or an adverb is removed does not affect the conciseness of the sentence significantly and the system lacks reliability in making such decisions, we decided not to remove adjectives and adverbs.

On average, the system reduced the length of the 500 sentences by 32.7% (based on the number of words), while humans reduced it by 41.8%.

The probabilities we computed from the training corpus covered 58% of instances in the test corpus. When the corpus probability is absent for a case, the system makes decisions based on the other two sources of knowledge.

Some of the errors made by the system result from the errors by the syntactic parser. We randomly checked 50 sentences, and found that 8% of the errors made by the system are due to parsing errors. There are two main reasons responsible for this relatively low percentage of errors resulting from mistakes in parsing. One reason is that we have taken some special measures to avoid errors introduced by mistakes in parsing. For example, PP attachment is a difficult problem in parsing and it happens often that a PP is wrongly attached. Therefore, we take this into account when marking the obligatory components using subcategorization knowledge from the lexicon (step 2) – we not only look at the PPs that are attached to a verb phrase, but also PPs that are next to the verb phrase but not attached, in case it is part of the verb phrase. We also wrote a preprocessor to deal with particular structures that the parser often has problems with, such as appositions. The other reason is that parsing errors do not always result in reduction errors. For example, the sentence "The spokesperson of the university said that ...", has a that-clause with a complicated structure which the parser gets wrong, the reduction system is not necessarily affected since it may decide in this case to keep the that-clause as it is, as humans often do, so the parsing errors will not matter in this example.

4.3 Discussion of Related Topics

In this section, we discuss three topics related to reduction, including adapting the program to query-based summarization, the interactive between the reduction module and other modules in the system, and several factors that affect the system's performance.

4.3.1 Reduction for Query-based Summarization

The reduction algorithm we have presented assumes generic summarization; that is, we want to generate a summary that includes the most important information in an article. We can tailor the reduction system to query-based summarization. In that case, the task of reduction is not to remove phrases that are extraneous in terms of the main topic of an article, but to remove phrases that are not very relevant to a user's query. We extended our sentence reduction program to query-based summarization by adding another step in the algorithm to measure the relevance of a user's query to phrases in the sentence. In the last step of reduction when the system makes the final decision, the relevance of a phrase to the query is taken into account, together with syntactic, contextual, and corpus information.

The method we use to measure the relevance between a user's query and phrases in the sentence are based on word similarities. For each phrase in the sentence (i.e., each subtree in the sentence parse tree), we tag it with the number of overlapping words (only non-stop words are considered) between the phrase and the query. A phrase is considered important if it has many words in common with the query. In the final decision making step, we keep a phrase that has a high percentage of words overlapping with the query, even when it is marked removable by the other three criteria (i.e., grammar, context, and statistics).

We cannot present any quantitative evaluation of the query-based reduction here due to the lack of corpora that can be used for such purpose.

4.3.2 Interaction between Reduction and Other Modules

Ideally, the sentence reduction module should interact with other modules in a summarization system. It should be able to send feedback to the extraction module if it finds that a sentence selected by the extraction module may be inappropriate (for example, having a very low context importance score). It should be able to request more sentences from the extraction module if the length of the summary is less than the required length after reduction. It should also be able to interact with the modules that run after it, such as the sentence combination module, so that it can revise reduction decisions according to feedback from these modules.

Currently, our reduction system can send feedback to the extraction module if it thinks an extracted sentence should be deleted as a whole (that happens when the context importance score of the selected sentence is very low, meaning that the sentence is not very related to other sentences). The reduction system is allowed to completely delete a sentence proposed by the sentence extractor; it can also request more sentences from the sentence extractor if the text after sentence reduction is shorter than the required length. The reduction module does not yet interact with the combination module, revising its decisions based on feedback from the combination module.

4.3.3 Factors that Affect the Performance

We discuss here some significant factors that affect the quality of the reduction result. In particular, we discuss the influence that the parser has on the syntactic correctness of a reduced sentence, the effectiveness of using lexical relations to identify the focus of the context, and the sparse data problem in statistical computation.

Parser Errors and Syntactic Correctness

The syntactic correctness of a reduced sentence largely depends on whether the output of the parser is right. If the parser analyzes the sentence correctly, it is extremely likely that the reduced sentence is grammatically correct — the only exception occurs when the combined lexicon misses indicating an obligatory argument of a verb phrase, which very rarely occurs given the wide coverage of the lexicon (combined from three large-scale resources). However, if the parser analyzes the sentence wrongly, the reduction result can be either grammatically correct or wrong, depending on the nature of the error(s) by the parser and what phrases the reduction program decides to remove. For example, if the parser misidentifies the subject of the sentence, it can happen that the real subject will be deleted during reduction. But as we mentioned in the evaluation section, parsing errors do not always result in reduction errors. For example, if reduction decides that nothing in the sentence should be removed, then even if the parse tree is wrong, the result is not affected. More detailed information on the parsing errors and the problems caused by them are presented in the evaluation results in Section 4.2.2.

Lexical Relations

We use lexical relations among words in the context as an approximation of semantic analysis for the purpose of identifying the focus of local context. The following questions arise immediately: Does such approximation produce good results? Out of the nine lexical relations we use, which ones are most important? What if WordNet misses some important relations between words? How can this affect the result?

In our analysis of the reduction output, we found that whether lexical relations can identify the important phrases in a sentence has much to do with the particular content and writing style of the document. Although we cannot precisely predict whether lexical relations will produce good results for a particular document, we did observe that the method tends to work well for a document that has a single, focused central topic and that has many repeated or related words (newspaper articles are more likely to be in this category), and the method tends to work less well for a document that does not have a single, focused central topic and whose words are less related (for example, a long narrative that describes someone's life story).

In our system, the nine types of lexical relations are considered in the following order of importance: repetition > inflection > derivation > synonymy > meronymy (part-of) > hypernymy > antonymy > entailment > causation. This importance is reflected in the assignment of weights in the computation of the context importance score; the more important types of lexical relations have higher weights. Deciding the order of importance of these lexical relations is experimental — we choose the above order partly based on linguistic theory [Hoey, 1991] and partly based on whether a particular type of lexical relation has many instances.

It is inevitable that WordNet misses certain relations between words, although it does encode many types of lexical relations and represents the largest number of lexical relations compared to other existing resources. Words such as "Gaza" and "Palestinians", "doctor" and "patient", "Bush" and "Republican" are related, but they are not represented in WordNet. One way to extract such relations is using lexical co-occurrence information. In another study we did in the field of Information Retrieval [Jing and Tzoukermann, 1999], we show related word pairs such as "Gaza" and

"Palestinians" can be found based on whether they frequently appear in the same documents in a corpus. However, such data computed from lexical co-occurrence information can also contain noise (for example, "today" and "government" happen to co-occur often in the corpus we used for computing lexical co-occurrence, so they are considered related although they are not). We experimented with adding this type of lexical co-occurrence based information to our program. We considered this co-occurrence relation as the tenth type of lexical relation, in addition to the nine types we have introduced before. We assigned its weight (representing its importance) as the lowest. The overall performance of the reduction system actually slightly dropped after adding this additional lexical relation. We do not believe that this is a conclusive indication that lexical co-occurrence is not useful — it is very possible that a different testing corpus will produce a contradictory result, since the words involved will be different then.

Sparse Data Problem

Our reduction program uses a corpus of reduced sentences written by humans to compute the probabilities that humans remove certain types of phrases. As in many statistical models, we also have a sparse data problem. If we compute the probabilities at a very "specific" level, there may be too few instances to give a reliable estimate, or worse, there may be zero instances, leaving the probability undefined. For this reason, when we compute the probability that a subtree in a parse tree (representing a phrase) is removed from its parent node, we represent the parent node using both its grammatical role and head word, but for the subtree node, we represent it using only its grammatical role. Hence, we compute the probability such as *PROB(adjective modifier is removed | n = "device")* (i.e., the probability that the adjective modifier is removed when the head noun is "device"), rather than *PROB(the word "new" as an adjective modifier is removed | n = "device")* (i.e., the probability that the word "new" as an adjective modifier is removed when the head word is "device"). We also decide against representing both parent node and the subtree node using only their grammatical roles, since a probability such as *PROB(adjective modifier is removed | n)* (i.e., the probability that the adjective modifier of a head noun is removed) is too "general" for our reduction purpose.

4.4 Related Work

4.4.1 Sentence Compression

[Knight and Marcu, 2000] discussed the sentence compression problem, which is very similar to our sentence reduction problem. They devised both noisy-channel and decision-tree approaches to the problem. The noisy-channel framework has been used in many applications, including speech recognition, machine translation, and information retrieval. Their system first parse the original sentence into a large parse tree, and then it hypothesizes and ranks various small trees that represent the compressed sentences using the stochastic model. The decision-tree method compresses sentences by learning the decisions for "rewriting" input parse trees into smaller trees, which correspond to the compressed versions of the original sentence.

The main difference between our sentence reduction program and the above compression algorithms include the following: (1) their compression algorithms do not consider the context of the sentences. They treat each sentence as isolated and independent. In contrast, in our reduction system, the context in which a sentence appears plays a very important role in determining how the sentence will be reduced.; (2) the compression algorithms can produce ungrammatical sentences, even when an input sentence is grammatical and the parser analyzes the sentence correctly. Both the noisy-channel model and the decision-tree model are stochastic model only. Our reduction program aims to guarantee the grammaticality of the reduced sentences, by relying on syntactic knowledge from a lexicon. It is unlikely to produce an ungrammatical sentence, if the input sentence is grammatical and the parser analyzes the sentence correctly.

The similarities between our reduction algorithm and their compression algorithms include the following: (1) both systems perform reduction or compression based on the parse trees of sentences; (2) both systems need training corpora.

[Knight and Marcu, 2000] evaluated the compression system by asking humans to assess the grammaticality and importance of the compressed sentences on a scale from 1 to 5 (larger values indicates better results). For the test corpus of 32 sentences from the Ziff-Davis corpus, the noisy-channel compressed the length of the sentences by 30% on average and the compressed sentences have an average score of 4.34 for grammaticality and a score of 3.38 for importance. For the same test corpus, the decision-tree model compressed the length of the sentences by 43% on average and the compressed sentences have an average score of 4.30 for grammaticality and a score of 3.54 for importance. The evaluation scheme we used, as shown in Section 4.2, is very different from the above evaluation method. We automatically computed the agreement between the reduction decisions made by humans and the reduction decisions made by our reduction system, using a measure defined as success rate. For a corpus of 500 sentences from news articles on Telecommunication issues, our reduction system reduced the length of the sentences by 33% on average while achieving an average success rate of 81%.

4.4.2 Reduction Based on Syntactic Roles Only

Some researchers suggested removing phrases or clauses from sentences for certain applications. [Grefenstette, 1998] proposed removing phrases in sentences to produce a telegraphic text that can be used to provide audio scanning services for the blind.

[Corston-Oliver and Dolan, 1999] proposed removing clauses in sentences before indexing documents for Information Retrieval. Both studies removed phrases based only on their syntactic categories. The reduction can be performed at different levels. For example, we can retain only subject nouns, head verbs, and object nouns and remove everything else in the original sentence, or we can retain subject nouns, head verbs, object nouns, and subclauses but remove the rest.

The fundamental difference between our reduction system and the above systems is that our focus is on deciding *when it is appropriate to remove a phrase*, based on such factors as the context in which a sentence appears and the syntactic properties of the words involved, whereas the reduction systems that are based on syntactic roles only perform reduction undistinguishly to each sentence guided only by the syntactic categories. Reducing sentences relying only on syntactic roles can produce ungrammatical sentences – for example, when an obligatory clause or to-infinitive or preposition phrase of a head verb is removed. It can also remove sentence components that are semantically important. In our evaluation of the reduction program shown in Section 4.2.2, we used the reduction based on syntactic roles as a baseline. The baseline method retained only the subject nouns, head verbs, and object nouns and undistinguishly removed everything else in the sentences. For a corpus of 500 sentences, the baseline method has a reduction success rate of 43%, in contrast to the success rate of 81% by our reduction system.

4.4.3 Text Simplification

Other researchers worked on the text simplification problem, which usually involves simplifying text but not removing any phrases. For example, [Carroll et al., 1998] discussed simplifying newspaper text by replacing uncommon words with common words, or replacing complicated syntactic structures with simpler structures to assist people with reading disabilities. [Chandrasekar, Doran, and Srinivas, 1996] discussed text simplification in general. The difference between these studies on text simplification and our system is that a text simplification system usually does not *remove* anything from an original sentence, although it may change its structure or words, but our system removes extraneous phrases from the extracted sentences.

4.4.4 Using Lexical Cohesion for Text Understanding

The way we use lexical relations is very close to the method proposed by Hoey in his book "Patterns of Lexis in Text" [Hoey, 1991]. As a linguist, he manually analyzed some documents and showed that lexical relations can possibly help to find most important sentences in the documents. He distinguished different types of lexical relations and gave them different weights in the computation, which we also did in our system. We made some modifications to his algorithms to suit our sentence reduction purpose. For instance, we consider only local context instead of the entire document while computing lexical relations.

Many other researchers have explored using lexical relations as an indicator of the structure of the text or the representation of the context. [Morris and Hirst, 1991] proposed using lexical cohesion computed by thesaural relations as an indicator of the structure of text. [Hirst and St-Onge, 1998] used lexical chains as representation of context for the detection and correction of malapropisms. [Barzilay and Elhadad, 1997] extended the lexical chains to the summarization purpose. [Benbrahim and Ahmad, 1995] adopted an approach similar to Hoey's method [Hoey, 1991] for extraction-based summarization.

4.5 Conclusion

We have presented a sentence reduction system that removes extraneous phrases from sentences that are extracted from an article in text summarization. The deleted phrases can be single words, noun phrases, verb phrases, preposition phrases, gerunds, to-infinitives, or clauses, and multiple phrases can be removed from a single sentence. The focus of this work is on determining, for a sentence in a particular context, which phrases in the sentence are less important and thus can be removed. Our system makes intelligent reduction decisions based on multiple sources of knowledge, including syntactic knowledge, context, and probabilities computed from corpus analysis. We also created a corpus consisting of 500 sentences and their reduced forms produced by human professionals, and used this corpus for training and testing the system. The evaluation shows that 81.3% of reduction decisions made by the system agreed with those of humans.

Chapter 5

Sentence Combination

Sentence combination merges the reduced sentences from sentence reduction with other sentences or phrases. For example, the combination operation can be performed on two sentences both of which have been compressed by sentence reduction, or it can be performed on a reduced sentence and a phrase selected from the original text.

Combination improves the coherence of generated summaries. As we indicated in Chapter 1, a main source of incoherence in extraction-based summaries is dangling pronouns and noun phrases present in extracted sentences. One type of combination operation that is performed by our combination system is to replace the dangling pronouns and phrases with the names of the entities they refer to in the original text. Thus, a main source of incoherence can be eliminated. Moreover, by grouping closely related sentences together and merging them as a single sentence, combination helps readers to understand the relations between sentences. Sentence combination is a technique widely used by expert summarizers.

We implemented an automatic sentence combination module. Input to the module includes the sentences that have already been reduced by the reduction module, as well as the original document. The output of the combination module is merged sentences, which are used to produce summaries. Our investigation of the sentence combination problem can be summarized into four steps. In the first step, we study what operations can be used to combine sentences. These operations are identified by analyzing combination examples produced by human experts. In the second step, we look into the restrictions that determine *when* to use *which* combination operation. Such restrictions are formalized as a set of rules. In the third step, we implement the combination operations and rules using a formalism based on Tree Adjoining Grammars (TAG) [Joshi, 1987, Joshi, Levy, and Takahashi, 1975, Joshi and Schabes, 1996]. Finally, we explored using machine learning techniques to automatically learn combination operations and combination rules.

The reminder of this chapter is organized as follows. In Section 5.1, we describe the operations that can be used to merge sentences and phrases. A total of 13 operations are identified and classified into 5 categories. In Section 5.2, we present the combination rules that are used to determine when to use which combination operation. The next section is on the implementation of the combination operations and rules, explaining why we chose the Tree Adjoining Grammar formalism for this purpose and how it is used. In Section 5.4, we describe our experiments of using machine learning techniques for automatically discovering combination operations and combination rules. In Section 5.5, we discuss the issues involved in the evaluation of sentence combination and show the evaluation results. Finally, we compare with related work and conclude the chapter with a short summary.

5.1 Combination Operations

Our first task in developing the sentence combination system is to investigate what operations can be used to combine sentences and phrases. One way to identify these operations is to rely on linguistic theories and list all the operations that we believe could be used to combine sentences and phrases. Another way is to analyze a corpus of example summary sentences that have already been produced by humans using the combination technique. We decide to use the latter approach, for the reasons that the examples in the corpus give a real sense of what operations are actually useful and widely used in combination, and more importantly, the examples give the actual context in which the combination operations are performed, which is very important later on when we construct combination rules to decide when to use which combination operation.

The corpus for sentence combination was built from the decomposition results. It contains sample summary sentences that were produced by humans using the combination technique, and the corresponding document sentences that were used to generate these summary sentences. For analysis purpose, each summary sentence is marked with the information of phrase origins, provided by the decomposition results.

After analyzing the examples in the above corpus, we identified 13 operations that can be used to combine sentences and phrases, and classified them into 5 categories based on their similarity. Table 5.1 shows the combination operations and their categories. We now elaborate in more details a couple of combination operations listed in this table and show some example summary sentences that were generated by human professionals using these combination operations.

One of the most frequently used operations is to *add names or descriptions for people or organizations*, as shown by the example in Figure 5.1.

In this example, the phrase *"Clayton at Columbia"* from sentence 77 was sub-

Categories	Combination Operations
Add descriptions or names	add descriptions for people or organizations
	add names for people or organizations
Aggregations	extract common subjects or objects of sentences
	transform a sentence to a clause of the merged sentence
	add connectives (e.g., *"and"*) to merge sentences
	add punctuations (e.g., *";"*) to merge sentences
Replace incoherent phrases	replace a dangling pronoun with the entity it refers to
	replace a dangling noun phrase with the entity it refers to
	replace a location adverb (e.g., *"here"*) with the name of the place it refers to
	remove connectives (e.g., *"moreover"*) if the previous sentence is not included
Replace with more general or specific information	replace a phrase with a shorter phrase containing more general description
	replace a phrase with a longer phrase containing more specific information
Mixed operations	combination of any of above operations

Table 5.1: Combination operations.

stituted by the more complete description *"Paul Clayton, Chairman of the department dealing with computerized medical information at Columbia"* from sentence 34. We can think of the scenario under which the human summarizer produced this example: the human considered sentence 77 as important so he extracted it, he noticed that the short description *"Clayton at Columbia"* in the extracted sentence did not give enough background information for the person that was quoted, he looked in the original document to search for the more detailed description of the person, he found it in sentence 34 in which the person was first mentioned, and then he replaced the short description of the person in sentence 77 with the more detailed information in sentence 34.

This is a very simple operation, but it has a very positive impact on the quality of generated summaries. Examples in our corpus indicate that this operation is extremely widely used by expert summarizers.

Combination Operation: add names and descriptions for people or organizations

Source Document Sentences:

Sentence 34: *"We're trying to prove that there are big benefits to the patients by involving them more deeply in their treatment",* **said Paul Clayton, Chairman of the department dealing with computerized medical information at Columbia.**

Sentence 77: **"The economic payoff from breaking into health care records is a lot less than for banks",** *said Clayton at Columbia.*

The output by a human professional:

"The economic payoff from breaking into health care records is a lot less than for banks", said Paul Clayton, Chairman of the department dealing with computerized medical information at Columbia.

Figure 5.1: An example sentence produced by adding names and descriptions for people or organizations.

Another frequently used operation is to *add connectives* to merge two reduced sentences into a single sentences, as shown by the example in Figure 5.2.

Combination Operation: add connectives

Source Document Sentences:

Sentence 8: But it also raises serious questions about the privacy of such highly personal information *wafting about the digital world.*

Sentence 10: The issue thus fits squarely into the broader debate about privacy and security on the Internet, *whether it involves protecting credit card number or keeping children from offensive information.*

The output by a human professional:

But it also raises the issue of privacy of such personal information *and* this issue hits the nail on the head in the broader debate about privacy and security on the Internet.

Figure 5.2: An example sentence produced by adding connectives.

This operation is performed on two sentences that have both been compressed by sentence reduction. The sentences that are merged by this operation are often close to each other in the original text and they are also semantically related. As we can see, this is very different from the operation of adding names and descriptions for people and organizations, which performs combination between a sentence and a phrase, and the source sentences can be very far apart in the original text.

Note that the second category of combination operation listed in Table 5.1 — *aggregation* — is also used in traditional natural language generation. The first three operations in that category — *extracting common subjects or objects, changing one sentence to a clause, and adding connectives* — have been well studied in generation [Shaw, 1995, Dalianis and Hovy, 1993], while the last operation — *adding punctuations* — has not been mentioned before. The fundamental difference between aggregation in traditional natural language generation and aggregation in our work is that, in summarization, we do not have the deep understanding of the semantic meanings of the input as in traditional generation systems. Therefore, we have to make aggregation decisions based on whatever understanding we have of the input sentences and phrases.

There are some combination operations performed by expert summarizers that we do not attempt to simulate in our automatic systems, for the reason that they are too complicated and need such deep semantic understanding that we do not yet have means to acquire such understanding automatically. For instance, an expert summarizer may change the subject of a sentence to a semantically equivalent substitute in order to merge sentences, as shown by the example in Figure 5.3.

In this example, the human summarizer produced a parallel structure in the combined sentence by substituting *"but the big ticket items in the proposal are requirements stipulating that ... ..."* with the roughly semantically equivalent expression *"BSA is asking ... ..."*. We cannot yet automatically decide such semantic equivalence in a reliable way.

Combination operation: replace phrases with semantic paraphrases

Source Document Sentences:

Sentence 1: The Business Software Alliance is asking, among other things, that the school district pay $300,000 — the approximate value of the copies — to the group's anti-privacy fund.

Sentence 2: But the big ticket items in the proposal are requirements stipulating that any unlicensed software running in other Los Angeles schools be replaced with licensed copies and that the district establish a team to train staff members in copyright do's and don'ts — efforts that district officials say would cost about $4.5 million.

The output by a human professional:

BSA is asking the school district pay $300,000 — the approximate value of the copies — to the group's anti-privacy fund, replace any unlicensed software with licensed copies, and that the district establish a team to train staff members in copyright regulations — efforts that the LA school district say would cost about $4.5 million.

Figure 5.3: An example sentence produced by replacing phrases with semantic paraphrases.

5.2 Rules for Applying Combination Operations

Once we have identified the operations that can be used to combine sentences and phrases, the next step is to decide *when to use which combination operation*. We experimented with two different approaches. The first approach is manual: we analyzed examples by humans and manually wrote a set of combination rules. In the second approach, we explored using various machine learning techniques to automatically learn combination rules from examples in the corpus. This section focuses on the manual approach, while Section 5.4 focuses on the automatic learning approach.

Each combination operation identified in Table 5.1 should be performed only on certain types of sentences and phrases and only under certain conditions. The role of *combination rules* is to state the restrictions for applying a combination operation. By analyzing the examples produced by humans and generalizing the conditions under which a combination operation was performed, we manually constructed a set of combination rules.

A sample combination rule is shown in Figure 5.4. This rule was constructed to decide when to perform the combination operation *adding names or descriptions for people or organizations*. Using this rule, the system can produce exactly the same result as that of the human summarizer for the example shown in Figure 5.1.

Operation: add names and descriptions for people or organizations:
IF: ((a person or an organization is mentioned the first time in a summary) and (the complete name or the complete description of the person or the organization is not used) and (there exists in the original document the complete name and description of the person or the organization))

THEN: replace the short phrase with the more complete name and description (or descriptions, if there are more than one)

Figure 5.4: Combination rule for adding names and descriptions for people or organizations.

Figure 5.5 shows the rule for extracting common subject of two closely related sentences. An example sentence produced using this combination rule is presented in Figure 5.6. The output by a human professional is also presented for comparison. As we can see, the expert actually combined three source document sentences into one, by first extracting common subject of two sentences and merging them into one, and then merging the result with the third sentence by adding the connective *while*. Our combination system performed the first operation, extracting common subject, but did not perform the second. Although the final result is not exactly the same as that of the human professional, it is quite similar to it, and we consider it clear and correct enough to be acceptable.

Operation: extract common subject of two related sentences:
<u>IF:</u> ((two sentences are close to each other in the original document) and (their subjects refer to the same entity) and (at least one of the sentences is reduced by sentence reduction))
<u>THEN:</u> merge the two sentences by removing the subject in the second sentence, and then combining it with the first sentence using connective *and*.

Figure 5.5: Combination rule for extracting common subject of two related sentences.

5.3 Implementing Combination Operations and Rules

After we have identified combination operations and constructed combination rules, the next task is to implement them. As we recall from previous chapters, the extracted sentences are first sent to the sentence reduction module, which parses them and marks which phrases (represented by subtrees in a parse tree) can possibly be deleted. These pruned parse trees — representing the reduced sentences — are then sent to the combination module. The actual combinations are performed on top of these parse trees.

5.3.1 Implementing Combination Operations

To implement the 13 combination operations presented in Table 5.1, we first defined and implemented two primitive operations, *adjoining and substitution*, to manipulate the parse trees, and then implemented all the combination operations by expressing them using these two primitive operations.

Substitution replaces a node of a parse tree. Let *Tree1* be a tree with nodes $\{A, B, C, D, E\}$, and *Tree2* be a tree with nodes $\{R, S, W\}$, as shown in Figure 5.7. The substitution function *Replace([Tree1, node_D]), [Tree2, node_S])* results in the substitution of node *D* in *Tree1* by node *S* in *Tree2*, as shown in Figure 5.7.

Adjoining merges two nodes into a new node. The adjoining function *Combine([Tree1, node_D]), [Tree2, node_S], [new_parent_node]* merges node *D* from *Tree1*

Operation: extract common subject of two related sentences

Source Document Sentences: Sentence 1: The new measure is an echo of the original bad idea, blurred just enough to cloud prospects both for enforcement and for court review.

Sentence 2: Unlike the 1996 act, this one applies only to commercial Web sites — thus sidestepping 1996 objections to the burden such regulations would pose for museums, libraries and freewheeling conversation deemed “indecent” by somebody somewhere.

Sentence 3: The new version also replaces the vague “indecency” standard, to which the court objected, with the better-defined one of material ruled ”harmful to minors.”

The output by the system:
The new measure is an echo of the original bad idea.

The new version applies only to commercial web sites and also replaces the vague “indecency” standard with the better-defined one of material ruled ”harmful to minors.”

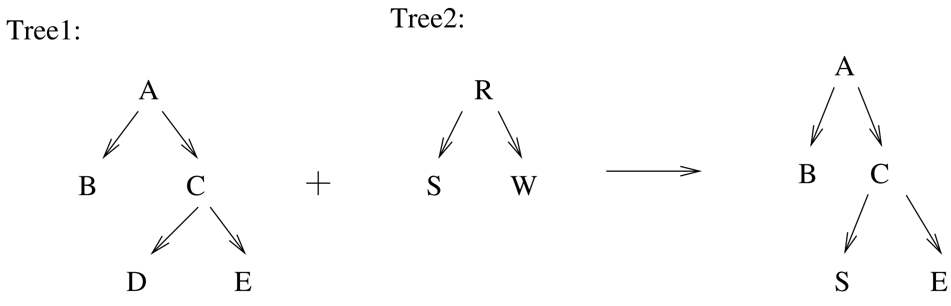
The output by a human professional:
While the new law replaces the “indecency” standard with “harmful to minors” and now only applies to commercial Web sites, the ”new measure is an echo of the original bad idea.”

Figure 5.6: Sample combination output produced by extracting common subject of two related sentences.

and node *S* from *Tree2* into a new tree and uses the new parent node as the root of the new tree, as shown in Figure 5.8.

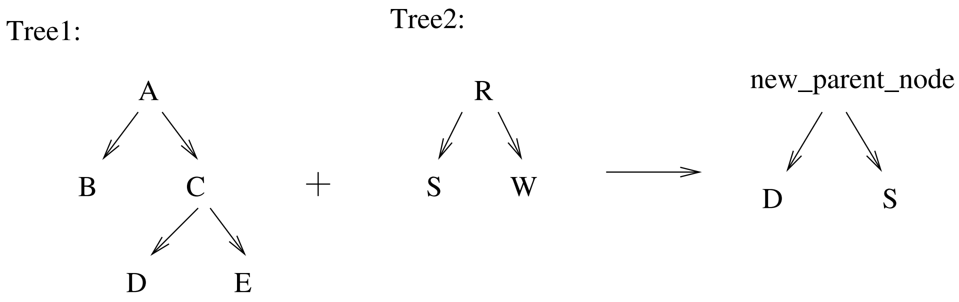
The above two functions — *Replace(old_node, new_node)* and *Combine(node1, node2, new_parent_node)* — are the only two functions we define in our system; all the combination operations are expressed and realized using these two primitive functions. Some of the combination operations can be implemented using these two primitive functions directly, while others need to be implemented as a series of primitive functions. We show a couple of examples here.

For instance, the combination operation *adding names and descriptions for people or organizations* can be realized using the *Replace* function directly. Suppose that



Substitution: Replace ([Tree1, node_D], [Tree2, node_S])

Figure 5.7: Tree substitution.

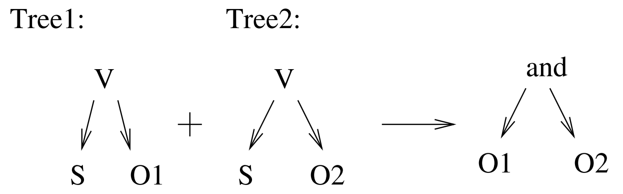


Adjoining: Combine([Tree1, node_D], [Tree2, node_S], [new_parent_node])

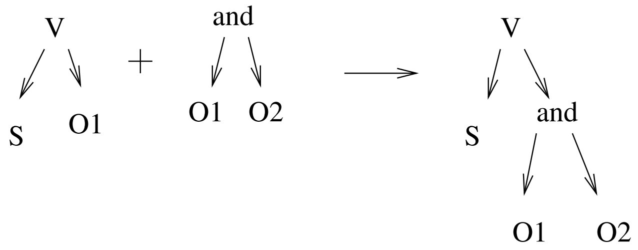
Figure 5.8: Tree adjoining.

in Figure 5.7, node *D* in *Tree1* represents a phrase that is a short name of a person, and node *S* in *Tree2* represents a phrase that is the detailed description of the same person. Then the function *Replace([Tree1, node_D]), [Tree2, node_S])* realizes the operation of adding descriptions for that person.

The combination operation *extracting common object of two related sentences* can be realized by first calling the adjoining function and then calling the substitution function, as shown in Figure 5.9. Suppose there are two sentences: $\{S, V, O_1\}$ and $\{S, V, O_2\}$, and S, V, O_1 and O_2 represent subject, verb, and objects respectively. The result after *extracting common object* should be $\{S, V, O_1 \textit{ and } O_2\}$. This involves first combining the two objects, O_1 and O_2, and adding the connective *and*, and then substituting the object O_1 in sentence $\{S, V, O_1\}$ with the result after adjoining.[1] We can express the above two steps using the following function: *Replace([Tree1,* $node\_O_1$*],* *Combine([Tree1,* $node\_O_1$*], [Tree2,* $node\_O_2$*], [new_parent_node_and])).*



(1) adjoining: Combine([Tree1, node_O1], [Tree2, node_O2], [new_parent_node_and])



(2) substitution: Replace([Tree1, node_O1], result_from_(1))

Figure 5.9: Using substitution and adjoining to realize the combination operation *extracting common object of two sentences*.

Similarly, all the 13 combination operations can be expressed using the *Replace* and *Combine* function. Note that in the discussion in this section, we use the term *node* to refer to a node in a tree, as well as the subtree which has that node as its root. Thus, *Replace* and *Combine* function can be performed on both words and phrases.[2]

The formalism we use here is based on Tree Adjoining Grammars (TAG) [Joshi, Levy, and Takahashi, 1975, Joshi, 1987, Joshi and Schabes, 1996]. As a tree-generating

[1] We could also substitute the object O_2 in sentence $\{S, V, O_2\}$ with the result after adjoining; the final results would be the same.

[2] Phrases are represented by subtrees in a parse tree.

system, TAG uses two composition operations: adjoining and substitution. We should state, however, that the definitions of adjoining and substitution in TAG are quite different from the *Replace* and *Combine* functions we have defined. For instance, the adjoining composition operation defined in TAG takes only two parameters as input: an auxiliary tree and a tree (it can be any tree, initial, auxiliary, or derived); adjoining builds a new tree from the two input trees. In contrast, the $Combine$ function we have defined takes three parameters: the two nodes (i.e., trees) to be combined and the new root node; it builds a new tree from the two input trees and adds the new node as the root. We may understand the relations between the substitution and adjoining operation defined in TAG and the *Replace* and *Combine* function defined in our system in this way: *Replace* and *Combine* are *functions* defined by taking into account the particular actions to be performed on parse trees for sentence combination purpose; they are not equivalent to *substitution* and *adjoining* tree composition operations in TAG, but can be implemented using the substitution and adjoining compositions.

5.3.2 Implementing Combination Rules

The combination rules we have shown earlier are very high level descriptions of the conditions under which particular combination operations are to be performed. In practice, to realize these rules, many tasks need to be accomplished.

For example, to realize the rule for adding names and descriptions for people or organizations, shown in Figure 5.4, the following tasks need to be accomplished. First, we need a tool that can recognize the names of people or organizations. For this, we use a named entity recognizer. Second, we need a tool that can recognize descriptions of people and organizations. For this, we implemented a tool of our own, which finds descriptions by analyzing appositions following a proper name. We consider an apposition immediately following a proper name as a description if it starts with "a", "an", "the", or a capitalized word and it does not end with verbs such as "say", "said" or "reports". This tool can find descriptions such as *"a member in the Congress Intelligence Committee"*, *"the non-profit organization that promotes international education"*, or *"President of the United States"*, but will eliminate phrases such as *"a London-based newspaper reported"*.[3] Third, we need to recognize which names and descriptions actually refer to the same person or organization. For instance, in the example shown in Figure 5.1, the system needs to detect that *Clayton* and *Paul Clayton* refer to the same person. We rely on a coreference system to find such connections. Last, the actual action of combining sentences and phrases needs to be implemented, which was presented in the previous subsection.

A couple of tools used in our sentence combination module are licensed from the MITRE Corporation. They are:

- **Named Entity Recognizer.** A named entity recognizer can identify entities (organizations, persons, and locations), times (dates, times), and quantities (monetary values, percentages). We use in our system the named entity recognizer provided by Alembic Workbench [Day et al., 1997].

- **Coreference Resolution System.** A coreference resolution system links together multiple expressions designating a given entity. The coreference relations are marked between elements of the following categories: nouns, noun phrases, and pronouns. We use the coreference system in the DeepRead system [Hirschman et al., 1999].

[3] [Radev, 1999] developed a tool for extracting person descriptions from multiple documents. It uses a finite-state grammar for noun phrases to represent different syntactic structures of pre-modifiers and appositions. Our person description tool is somewhat similar to his in that both systems rely on analyzing the syntactic structure of appositions to extract descriptions; the difference is that we do not consider pre-modifiers.

5.4 Investigating Machine Learning Methods

Having identified the combination operations and constructed a set of combination rules by manually analyzing examples, we are interested in exploring machine learning techniques that can possibly be used to automatically discover combination operations or rules that we have not found by hand.

We mainly experimented with symbolic machine learning techniques. Symbolic machine learning methods acquire non-numerical knowledge from supervised or unsupervised data. Standard supervised symbolic approaches include: decision tree induction, logical rule induction, and instance-based learning.

The machine learning program we experimented with is *ripper* [Cohen, 1995, Cohen, 1996]. Ripper learns a rule set from examples, which seems to suit our application quite well. It induces classification rules from a set of pre-classified examples. The user provides a set of examples, each of which has been labeled with the appropriate *class*. Ripper will then look at the examples and find a set of rules that will predict the class of later examples.

There are several reasons why we choose Ripper over other machine learning systems. First, ripper's hypothesis is expressed as a set of if-then rules. These rules are relatively easy for people to understand; if the ultimate goal is to gain insight into the data, then ripper is probably a better choice than a neural network learning method, or even a decision tree induction system. The combination rules we have manually constructed are also represented as a set of if-then rules; we would like to compare the manual rules with the rules learned by ripper to see exactly what we have gained by using automatic learning technique. Second, ripper is faster than other competitive rule learning algorithms. Third, ripper allows the user to specify constraints on the format of the learned if-then rules. If there is some prior knowledge about the concept to be learned, then these constraints can often lead to more accurate hypotheses.

In order to use ripper, we must first formulate the sentence combination problem as a classification problem and represent each combination example as a feature vector. We let each type of combination operation correspond to a class; therefore, the number of classes equals to the total number of combination operations. As for the features, we selected them by looking at the information that is used in the if-then rules that we have manually constructed before. The features include:

1. distance between source sentences
2. common element in source sentences (no common element, common subject, common verb, or common object)
3. named entities in source sentences (people or organizations)
4. coreference links

Sample rules learned by ripper are shown below:

- IF (common_element: "subject") THEN operation: "extract_common_subject"
- IF ((named_entity_type: people) and (named_entity_length: short)) THEN operation: "add_names_and_descriptions"

If we compare the second rule with the rule that we have manually constructed for *adding names and descriptions for people or organizations*, shown in Figure 5.4, we can see that the manual rule includes additional, useful constraints that the rule learned by ripper does not include: for instance, whether the named entity has appeared in the summary before.[4] An analysis of the rules learned by ripper indicates that: (1)

[4] If a person or organization has been mentioned before in the summary, supposedly the names and descriptions have been added then, so there is no need to add the names and descriptions again.

they generally have fewer constraints than the manual rules, and (2) one manual rule typically corresponds to more than one rules learned by ripper. The first finding can be explained by the fact that some of the constraint information in the manual rules are not represented as a feature in the input data to ripper. The learning program cannot possibly learn these constraints without such information. The second finding suggests that the rules that we manually constructed are more generalized and they cover more examples.

There are two main limitations with using ripper to learn combination operations and rules. First, the program cannot learn *new* combination operations. Ripper uses supervised data; the user need to label the examples into one of the *pre-defined* classes. Since each class corresponds to a combination operation, this requires the user to identify all the combination operations beforehand. Second, it is very difficult to represent combination examples as feature vectors. Ripper allows values of attributes to be either nominal, continuous, or set-valued. Therefore, we must represent the information about the two source documents sentences, such as named entity information, coreference information, and the relations between phrases and sentences, in the acceptable formats. Such representation is difficult.

Other well-known and widely used machine learning programs, such as CART, C4.5, and ID3, also learn classification rules using supervised data. Therefore, we will have above problems as well while using these programs to learn combination rules.

5.5 Evaluation

The evaluation of sentence combination module is not as straightforward as that of decomposition or reduction since combination happens later in the pipeline and its result depends on the output from prior modules. To evaluate only the combination component, we need to assume that the system makes the same reduction decision as humans and the coreference system has perfect performance. This involves manual tagging of some examples to prepare for the evaluation. The evaluation of sentence combination focuses on the assessment of the effectiveness of combination rules.

We manually tagged 30 examples, each of which includes a combined sentence and original document sentences. We then applied our combination rules to the original document sentences to see whether the system can reproduce human output. For the 30 examples, the output of 14 examples (47%) is exactly the same as that from humans, the output of 8 examples (27%) is similar to that from humans, and the output of the remaining 8 examples (27%) is dissimilar but considered clear and coherent enough to be acceptable.

5.6 Related Work

[Mani, Gates, and Bloedorn, 1999] addressed the problem of revising summaries to improve their quality. They suggested three types of operations: elimination, aggregation, and smoothing. The goal of the elimination operation is similar to that of the sentence reduction operation in our system. The goal of the aggregation operation and the smoothing operation is similar to that of the sentence combination operation in our system. The aggregation operation in their system deals only with sentences that have coreferential NPs. Sample actions include changing sentences to relative clauses, and copying and inserting non-restrictive relative clause modifiers, appositive modifiers of proper names, and proper name appositive modifiers of definite NPs. The smoothing operation performs tasks such as extracting common element of two sentences and substituting dangling noun phrases or pronouns with the names of entities they refer to in the original text. As we can see, there is some overlap between the set of combination operations we have identified and their set of aggregation and smoothing operations. This is not surprising, since there is only a finite set of operations that are frequently used to combine sentences and phrases. Such operations are also investigated for traditional natural language generation [Robin, 1994].

The difference between our approach and the above work is that instead of just relying on linguistic knowledge to derive these combination operations, we rely on a corpus, which not only provides us with the most frequently used operations by humans, but also lets us discover combination operations that would be neglected if relying on linguistic knowledge alone. Also, our system has explicit rules to constrict when to use which combination operation. These rules take into account the context in which the sentences appear and choose an operation that is considered appropriate for that context.

Sentence and phrase combination has also been used for multi-document summarization. For example, [Barzilay, McKeown, and Elhadad, 1999] proposed generating a summary for multiple documents by identifying and synthesizing related text across related text. [Radev, 1999] proposed generating summaries from multiple on-line sources using language reuse and regeneration technique. The fact that our work is on single document summarization and theirs is on multiple document summarization decides that we have very different focuses. In multiple document summarization, the focus is on how to integrate texts that come from different documents, which may overlap or contradict each other, while the focus in our single document summarization system is on how to integrate texts that come from different units within a document and generally do not have overlapping or contradiction problems.

5.7 Conclusion

We have presented a sentence combination program that merges sentences reduced by sentence reduction with other reduced sentences or phrases. A total of 13 combination operations are identified, and they are classified into 5 categories. A set of combination rules have been constructed to determine when to use which combination operation. The implementation of combination operations uses a formalism based on Tree Adjoining Grammars. We also explored using rule induction machine learning technique to automatically acquire combination rules.

Chapter 6

A Large-Scale, Reusable Lexicon for Natural Language Generation

Many types of knowledge are needed by the cut-and-paste summarization system for it to edit extracted sentences correctly. An important type of knowledge required by the system is the syntactic information. The sentence reduction module, presented in Chapter 4, relies on a large-scale lexicon we have constructed to provide the knowledge about the syntactic properties of verbs.

In this chapter, we present this large-scale lexicon that we constructed by combining multiple heterogeneous resources. In Section 6.1, we discuss the motivation for building such a lexicon. In Section 6.2, we give a brief introduction to the combined lexicon, presenting the resources that the lexicon was combined from, the types of knowledge included in the lexicon, and the size of the lexicon. In Section 6.3, we describe in detail how the lexicon was constructed by merging multiple, large-scale, heterogeneous resources. Our construction process is semi-automatic; the actual merging of different resources was all done automatically, but some databases used in the merging process were created manually. The combined lexicon has been used in many applications, including our cut-and-paste summarization system, traditional natural language generation, word sense disambiguation, and machine translation. We briefly describe the applications of the combined lexicon in Section 6.4. Finally, we conclude the chapter with a discussion of related work and a short summary of the chapter.

6.1 Motivation

The lexicon was initially constructed for traditional Natural Language Generation purposes. There are two main reasons for building this lexicon. First and foremost, we want to provide language generation with a lexicon that is indexed at the *semantic concept* level. Most existing large-scale resources encode their knowledge at the word level. While such resources are suitable for language interpretation, they are not particularly suitable for generation. An ideal generation lexicon should represent its knowledge at the semantic concept level rather than at the word level, because the input of a generation system generally consists of semantic concepts and most of the processing in the generation process is based on these semantic concepts.

The second reason is that we want to be able to use the knowledge from multiple resources simultaneously in a single generation system. Generally, generation requires many types of knowledge, such as lexical, syntactic, and semantic knowledge, but these different types of knowledge are often encoded in separate resources. Because the denotation and format used by one resource can be completely different from those used by another resource, it is difficult to directly use multiple resources in a single system. Combining these resources into a single, reusable lexicon makes sharing the knowledge from multiple resources possible.

Most traditional natural language generation systems that have been developed so far are supported by small lexicons with limited entries and hand-coded knowledge. Although such lexicons are reported to be sufficient for the specific domains in which generation systems work, there are some obvious deficiencies: (1) Hand-coding is time and labor intensive, and introduction of errors is likely; (2) Even though some knowledge, such as syntactic structures for a verb, is domain-independent, often it is re-encoded each time a new application is under development; (3) Hand-coding seriously restricts the scale and expressive power of a generation system.

The problem with hand-coding becomes more acute when we aim to develop a domain-independent system, like our cut-and-paste generation system for summarization. In such a situation, the entries needed to be included in a lexicon is are so many that the practice of hand-coding the lexicon by individual developers becomes impractical.

6.2 Introduction of the Lexicon

We selected four large-scale, heterogeneous resources as the information sources of the combined lexicon. In our selection of resources, we aim primarily for accuracy of the resource, large-coverage, and provision of a particular type of information especially useful for natural language generation. The four resources are:

1. The WordNet on-line lexical database [Miller et al., 1990]. WordNet is a well known on-line dictionary, consisting of 121,962 unique words, 99,642 synsets (each synset is a lexical concept represented by a set of synonymous words), and 173,941 senses of words.[1] It is especially useful for generation because it is based on *lexical concepts*, rather than words.

[1] As of Version 1.6, released in December 1997.

2. English Verb Classes and Alternations (EVCA) [Levin, 1993]. EVCA is an extensive linguistic study of diathesis alternations, which are variations in the realization of verb arguments. For example, the alternation “there-insertion” transforms *“A ship appeared on the horizon”* to *“There appeared a ship on the horizon”*. Knowledge of alternations facilitates the generation of paraphrases. [Levin, 1993] studied 80 alternations for 3,104 verbs.

3. The COMLEX (COMmon LEXicon) syntax dictionary [Grishman, Macleod, and Meyers, 1994]. The COMLEX Syntax dictionary consists of 38,000 head words (including approximately 21,000 nouns, 8,000 adjectives and 6,000 verbs), all of which are marked with a rich set of syntactic features and complements. Nouns have 9 possible features and 9 possible complements; adjectives have 7 features and 14 complements; and verbs have 5 features and 92 complements. The complements/subcategorization for verbs represent the arguments verbs can take. For example, verbs marked with subcategorization *“np”* take a noun phrase complement (e.g., *“John like Mary”*), and verbs marked with subcategorization *“np-to-inf”* take a noun phrase complement followed by a to-infinitive complement (e.g., *“I want John to go”*).

4. The Brown Corpus tagged with WordNet senses [Miller et al., 1993]. The Brown corpus [Kucera and Francis, 1967] consists of about a million words, all labeled with their parts of speech. Part of the Brown Corpus has been manually tagged with WordNet senses by the WordNet group. We use this corpus for frequency measurements.

In the construction of the lexicon, we focus on verbs, since they play the most important role in deciding phrase and sentence structure. The combined lexicon contains syntactic, semantic, and lexical knowledge for verbs, indexed by senses of verbs as required by generation, including:

- A complete list of syntactic subcategorizations for each sense of a verb.
- A large variety of transitivity alternations for each sense of a verb.
- Frequency of lexical items and verb subcategorizations and also selectional constraints derived from a corpus.
- Rich lexical relations between lexical concepts.

Figure 6.1 shows a sample entry of the lexicon. This entry shows that, as a verb, the word *appear* has 8 senses. For each sense, the lexicon lists the complements that *appear* can take when used in that particular sense, the syntactic alternations, and the semantic constraints on the subject and object of the sentence. The frequency information derived from corpus and the lexicon relations from WordNet are not shown in this figure, although they are included in the lexicon.

Figure 6.2 shows the construction process and the size of the combined lexicon. EVCA (consisting of 3,104 verbs), was first combined with COMLEX (5,583 verbs). The result after merging COMLEX with EVCA (5,920 verbs) was then combined with WordNet (14,738 verbs and phrases). The final lexicon consists of 5,676 verbs, each of which is represented at the sense level. On average, each verb in the lexicon has 2.5 senses. Each sense of a verb is marked with a rich set of subcategorizations and alternations, as shown in the sample entry for *appear* in Figure 6.1. Each sense has an average of 3.1 subcategorizations and 0.2 alternations.

```
appear:
sense 1 give an impression
((PP-TO-INF-RS :PVAL ("to") :SO ((sb, –)))
(TO-INF-RS :SO ((sb, –)))
(NP-PRED-RS :SO ((sb, –)))
(ADJP-PRED-RS :SO ((sb, –) (sth, –)))))
sense 2 become visible
((PP-TO-INF-RS :PVAL ("to")
    :SO ((sb, –) (sth, –)))
...
(INTRANS THERE-V-SUBJ
          :ALT there-insertion
          :SO ((sb, –) (sth, –))))
...
sense 8 have an outward expression
((NP-PRED-RS :SO ((sth, –)))
(ADJP-PRED-RS :SO ((sb, –) (sth, –))))
```

Figure 6.1: Entry for the verb *appear* in the combined lexicon.

6.3 Combining Multiple Large-Scale Heterogeneous Resources

This section describes in detail how the lexicon was constructed by combining the four linguistic resources we introduced in the previous section. We merged data from the four resources in a manner that aims to achieve high accuracy and completeness.

Our algorithm first merges COMLEX and EVCA, producing a list of syntactic subcategorizations and alternations for *each verb*. Distinctions in these syntactic restrictions according to each *sense* of a verb are achieved in the second stage, where WordNet is merged with the result of the first step. Finally, corpus information is added, complementing the static resources with actual usage counts for each syntactic pattern. This allows us to detect rarely used constructs that should be avoided during generation, and

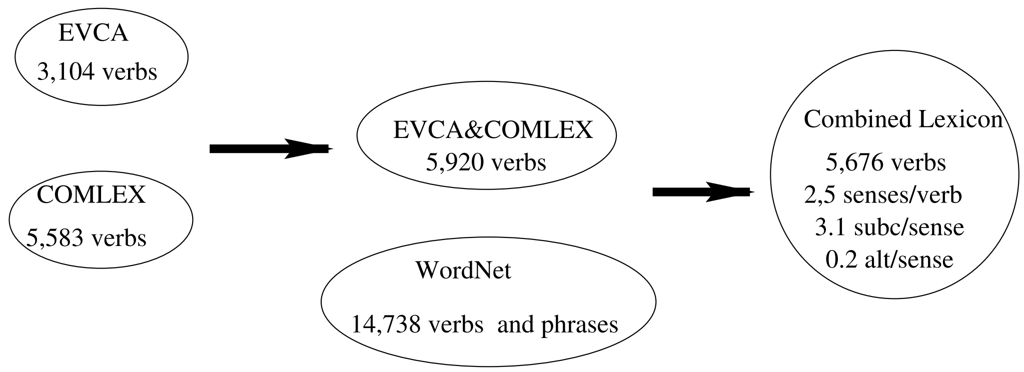


Figure 6.2: Construction of the combined lexicon and the size of resources.

possibly to identify alternatives that are not included in the lexical databases.

6.3.1 Step 1: Merging COMLEX with EVCA

COMLEX uses 92 frames to represent the structure of verb phrases. Figure 6.3 shows the entry for the verb *appear* in the COMLEX syntax dictionary. Most of these frames are self-explanatory. For example, *"INTRANS"* means that a verb is intransitive. Detailed explanation of these frames can be found in COMLEX Syntax Reference Manual [Macleod and Grishman, 1995].

EVCA contains subcategorization and alternation information for verbs and verb classes. Alternations involve syntactic transformations of verb arguments. They are thus a means to alleviate the usual lack of alternative ways to express the same concept in current generation systems.

To be able to merge the syntactic information from EVCA with that from COMLEX, we first need to represent the information in EVCA in a format that can be automatically analyzed. Therefore, we first convert the verb subcategorization and alternation information present in Levin's book [Levin, 1993] to a format that is readable by computers and that is compatible with the representation used in COMLEX. We ex-

```
appear:
((INTRANS)
 (SEEM-S)
 (SEEM-TO-NP-S)
 (TO-INF-RS)
 (NP-PRED-RS)
 (ADJP-PRED-RS)
 (ADVP-PRED-RS)
 (AS-NP)
 (EXTRAP-TO-NP-S)
 (PP-TO-INF-RS :PVAL ("to"))
 (PP-PRED-RS :PVAL ("to"
                    "of"
                    "under"
                    "against"
                    "in favor of"
                    "before"
                    "at")))
```

Figure 6.3: Entry for the verb *appear* in COMLEX.

tracted the relevant information for each verb using the verb classes to which the various verbs are assigned; verbs of the same class have the same syntactic properties. EVCA specifies a mapping between words and word classes, associating each class with alternations and with subcategorization frames. Using the mapping from word to word classes, and from word classes to alternations, alternations for each verb are extracted.

We manually formatted the alternate patterns in each alternation in COMLEX format. The reason to choose manual formatting rather than automating the process is to guarantee the reliability of the result. In terms of time, manual formatting is no more expensive than automation since the total number of alternations is small (80). When an alternate pattern cannot be represented by the labels in COMLEX, we need to add new labels during the formatting process; this also makes automating the process difficult.

The formatted EVCA consists of sets of applicable alternations and subcategorizations for 3,104 verbs. We show the sample entry for the verb *appear* in Figure 6.4. Each verb has 1.9 alternations and 2.4 subcategorizations on average. The maximum number of alternations (13) is realized for the verb “roll”.

The merging of COMLEX and EVCA is achieved by unification, which is possible due to the usage of similar representations. Two points are worth mentioning: (a) When a more general form is unified with a specific one, the latter is adopted in the final result. For example, the unification of PP[2] and PP-PRED-RS[3] is PP-PRED-RS. (b) Alternations are validated by the subcategorization information. An alternation is applicable only if both alternate patterns are applicable.

Applying this algorithm to our lexical resources, we obtain rich subcategorization and alternation information for each verb. COMLEX provides most subcategorizations, while EVCA provides certain rare usages of a verb which might be missing from COMLEX. Conversely, the alternations in EVCA are validated by the subcategorizations in COMLEX. The merging operation produces entries for 5,920 verbs out of 5,583 in COMLEX and 3,104 in EVCA.[4] Each of these verbs is associated with 5.2 subcategorizations and 1.0 alternations on average. Figure 6.5 is an updated version of Figure 6.4 after this merging operation.

6.3.2 Step 2: Merging COMLEX/EVCA with WordNet

WordNet is a valuable resource for generation because, most importantly, the synsets provide a mapping between concepts and words. Its inclusion of rich lexical relations also provide a basis for lexical choice. Despite these advantages, the syntactic information in WordNet is relatively poor. Conversely, the result we obtained after combining

[2] A verb can take a prepositional phrase complement.

[3] A verb can take a prepositional phrase complement, and the subject of the prepositional phrase is the surface subject of the sentence. E.g., “The work appears of great significance.”

[4] 2,947 words appear in both resources.

```
appear:
((INTRANS)
 (LOCPP)
 (PP)
 (ADJ-PER-PART)
 (INTRANS THERE-V-SUBJ :ALT There-Insertion)
 (LOCPP THERE-V-SUBJ-LOCPP :ALT There-Insertion)
 (LOCPP LOCPP-V-SUBJ :ALT Locative_Inversion))
```

Figure 6.4: Alternations and subcategorizations from EVCA for the verb *appear*.

```
appear:
((PP-TO-INF-RS :PVAL ("to"))
 (PP-PRED-RS :PVAL ("to" "of" "under" "against"
                    "in favor of" "before" "at"))
 (EXTRAP-TO-NP-S)
 (INTRANS)
 ...
 (INTRANS THERE-V-SUBJ :ALT There-Insertion)
 (LOCPP THERE-V-SUBJ-LOCPP :ALT There-Insertion)
 (LOCPP LOCPP-V-SUBJ :ALT Locative_Inversion)))
```

Figure 6.5: Entry for the verb *appear* after merging COMLEX with EVCA.

COMLEX and EVCA has rich syntactic information, but this information is provided at the word level and thus is unsuitable to use for generation directly. These complementary resources are therefore combined in the second stage, where the subcategorizations and alternations from COMLEX/EVCA for each word are assigned to each sense of the word.

Each synset in WordNet is linked to a list of verb frames, each of which represents a simple syntactic pattern and general semantic constraints on verb arguments. A sample verb frame used in WordNet is *"Somebody –s something"*, which means that a verb takes an inanimate object as complement and the subject of the sentence is animate (e.g., *"Women like chocolates"*). A total of 35 verb frames are used in WordNet. The fact that WordNet contains this syntactic information (albeit poor) makes it possible to link the result from COMLEX/EVCA with WordNet.

The merging operation is based on a compatibility matrix, which indicates the compatibility of each subcategorization in COMLEX/EVCA with each verb frame in WordNet. The subcategorizations and alternations listed in COMLEX/EVCA for each word is then assigned to different senses of the word based on their compatibility with the verbs frames listed under that sense of the word in WordNet. For example, if for a certain word, the subcategorizations PP-PRED-RS and NP are listed for the word in COMLEX/EVCA, and the verb frame *"Somebody –s PP"* is listed for the first sense of the word in WordNet, then PP-PRED-RS will be assigned to the first sense of the word while NP will not, because PP-PRED-RS and *"Somebody –s PP"* are compatible, while NP and *"Somebody –s PP"* are not. We also keep in the lexicon the general constraint on verb arguments from WordNet frames. Therefore, for this example, the entry for the first sense of the word will be *"PP-PRED-RS :subject(somebody)"*, which indicates that the verb can take a prepositional phrase as a complement, the underlying subject of the prepositional phrase is the surface subject, and the subject of the sentence should be in the semantic category "somebody". As you can see, the result incorporates information from three resources, but is more informative than any of them. An alternation is considered applicable to a word sense if both alternate patterns have matchable verb frames under that sense.

The compatibility matrix is the kernel of the merging operations. The 147×35 matrix (147 subcategorizations from COMLEX/EVCA, 35 verb frames from WordNet) was first manually constructed based on human understanding. In order to achieve high accuracy, very strict restrictions are used to decide whether a pair of labels are compatible when the matrix was first constructed. We then use regressive testing to adjust the matrix based on the analysis of merging results. During regressive testing, we first merge WordNet with COMLEX/EVCA using a current version of the compatibility matrix, and write all inconsistencies to a log file. In our case, an inconsistency occurs if a subcategorization or alternation in COMLEX/EVCA for a word cannot be assigned to any sense of the word, or a verb frame for a word sense does not match any subcategorization for that word. We then analyze the log file and adjust the compatibility matrix accordingly. This process is repeated 6 times until when we analyze a fair amount of inconsistencies in the log file, they are no longer caused by the restrictions in the compatibility matrix.

Inconsistencies between WordNet and COMLEX/EVCA result in subcategorizations that cannot be assigned to any sense of a verb and verb frames that do not match any of the subcategorizations for the verb. On average, 15% of subcategorizations and alternations for a word cannot be assigned to any sense of the word, mostly due to incomplete syntactic information in WordNet; 2% verb frames for each sense of a word do not match any subcategorizations for the word, either due to incomplete information in COMLEX/EVCA or erroneous entries in WordNet.

The lexicon at this stage contains a rich set of subcategorizations and alternations for each sense of a word, coupled with semantic constraints on verb arguments. For 5,920 words in the result after combining COMLEX and EVCA, 5,676 words also appear in WordNet and each word has 2.5 senses on average. After the merging operation, the average number of subcategorizations is refined from 5.2 per verb in COMLEX/EVCA to 3.1 per sense, and the average number of alternations is refined from 1.0 per verb to 0.2 per sense. Figure 6.1 shows the result for the verb *appear* after the merging operation.

6.3.3 Step 3: Adding Corpus Information

Finally, we enriched the lexicon with language usage information derived from corpus analysis. The corpus used here is the Brown Corpus tagged with WordNet senses. The language usage information in the lexicon includes: (1) frequency of each word sense; (2) frequency of subcategorizations for each word sense. The corpus analysis information complements the subcategorizations from the static resources by marking potential superfluous entries and supplying entries that are possibly missing in the lexical databases; (3) semantic constraints of verb arguments. The arguments of each verb are clustered based on hyponymy hierarchy in WordNet. The semantic categories we thus obtained are more specific compared to the general constraint (animate or inanimate) encoded in WordNet frame representation. The language usage information is especially useful for lexical choice in traditional natural language generation.

6.4 Applications of the Combined Lexicon

The combined lexicon has been used in many applications. We used it in our cut-and-paste summarization system to prevent removing obligatory verb arguments during the sentence reduction process; we used it in a practical, traditional natural language generation system [McKeown, Kukich, and Shaw, 1994, Jing and McKeown, 1998] to improve the reusability of the lexical choice and realization components and to improve the power of generating paraphrases; we integrated it with a unification-based natural language generator [Elhadad, 1992, Robin, 1994] to improve the system's ability to avoid generating non-grammatical output and to allow the lexicon to be usable by many generation systems; we used it in word sense disambiguation to prune verb senses. Other researchers have also used it for machine translation and language generation. Next, we briefly present the usage of the lexicon in these applications. Details can be found in relevant papers.

6.4.1 Cut-and-Paste Summarization

As shown in Section 4.1, the combined lexicon was used in the sentence reduction process to prevent removing obligatory verb arguments. The example shown earlier is the verb *convince*, which has the following entry in the combined lexicon:

convince

sense 1:	NP-PP :PVAL ("of")
	NP-TO-INF-OC
sense 2:	NP

This entry indicates that the verb "convince" can be followed by a noun phrase and a prepositional phrase starting with the preposition "of" (e.g., he convinced me of his innocence). It can also be followed by a noun phrase and a to-infinitive phrase (e.g., he convinced me to go to the party). This information is used in step 2 of the reduction algorithm to mark that the "of" prepositional phrase or the to-infinitive are obligatory parts of the verb phrase, and therefore, they will not be deleted even if context information or other types of information may indicate that they are not essential.

Two points are critical for the application of the lexicon in the summarization system to be successful: first, the lexicon must have broad word coverage, because we aim to develop a domain-independent system and the summarizer must be able to handle all the words that are commonly encountered; second, the syntactic information in the lexicon should be as complete as possible, since missing subcategorizations for verbs may result in the wrongful removal of obligatory verb arguments and thus incoherent output. Our lexicon has broad coverage and includes extensive syntactic information, so it meets the requirements of summarization.

6.4.2 Traditional Natural Language Generation

The fact that knowledge in the combined lexicon is encoded at the semantic concept level makes it particularly suitable for traditional natural language generation applications. PlanDOC [McKeown, Kukich, and Shaw, 1994, Jing and McKeown, 1998] is a practical generation system that we developed for the former Bellcore company (now called Telcordia Technologies). It is an enhancement to Bellcore's LEIS-PLAN^{TM} network planning product. PlanDOC transforms lengthy execution traces of engineer's interaction with LEIX-PLAN into human-readable summaries.

We proposed a multi-level feedback architecture for lexical choice and realization. When used together with our combined lexicon, this architecture improves the reusability of lexical choice and realization modules. Figure 6.6 shows the multi-level feedback architecture.

To understand this architecture, we first need to distinguish three different types of paraphrases: semantic paraphrases, lexical paraphrases, and syntactic paraphrases. If one is asked whether one will be at home tomorrow, then answers such as *"I'll be at work tomorrow"*, *"No, I won't be at home."*, and *"I'm leaving for vacation tonight"* can be considered paraphrases at the semantic level since they all express the same facts, that is, the person won't be at home tomorrow. Paraphrases like *"He bought an umbrella"* and *"He purchased an umbrella"* are at the lexical level since they are acquired by substituting certain words with synonymous words. Paraphrases like *"A ship appeared on the horizon"* and *"On the horizon appeared a ship"* are at the syntactic level since

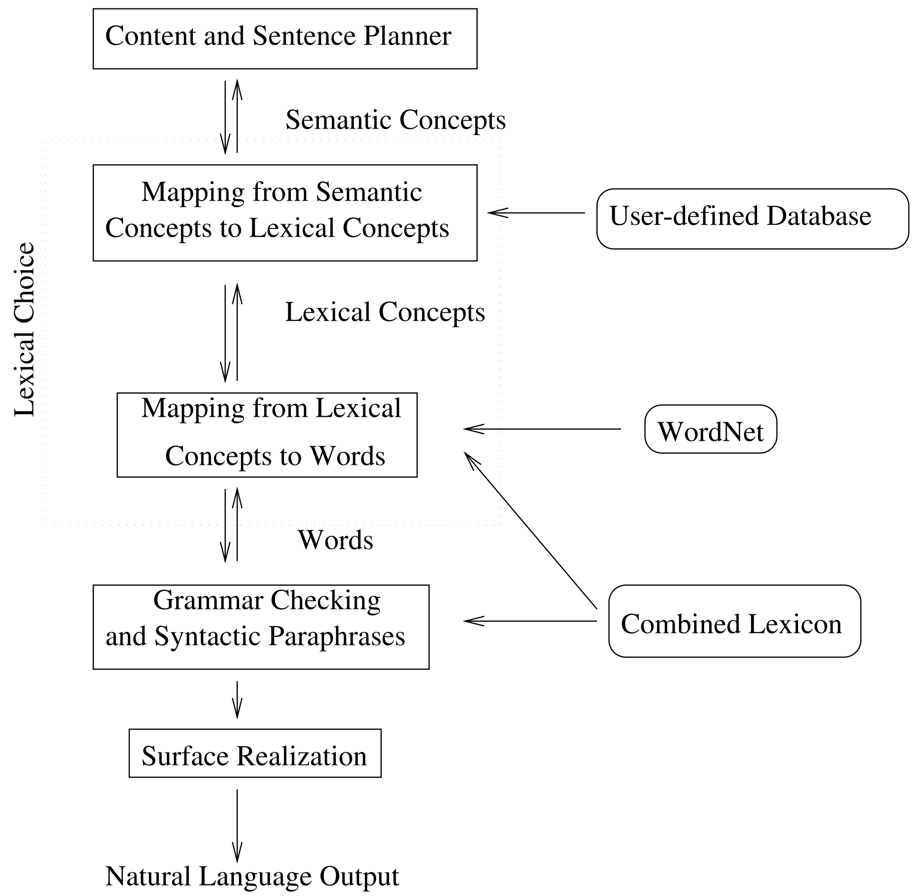


Figure 6.6: The multi-level feedback architecture for lexical choice and realization.

they only involve syntactic transformations.

There are three stages in the multi-level architecture we proposed. The first stage is semantic paraphrasing, which maps semantic concepts to lexical concepts using a user-defined database. The second stage is lexical paraphrasing, which maps lexical concepts to words, using the synsets in WordNet and the syntactic and semantic constraints in our combined lexicon. The third stage is syntactic paraphrasing, which does grammatical checking and generates syntactic paraphrases using the subcategorization/alternation information in our combined lexicon. The first stage is domain-dependent, while the modules that implement the second and the third stage can be reused. The combined lexicon, which is used in the second and third stage, can also be reused by many generation systems. By refining the processing steps in lexical choice and realization into multiple levels, we aim to separate the processes that are domain-independent from the processes that are domain-independent, and therefore, reuse the domain-independent modules when a new application is under development.

In PlanDOC, at least three paraphrases were defined for each message at the semantic level. For example, *"The base plan called for one fiber activation at CSA 2100"* and *"There was one fiber activation at CSA 2100"* are semantic paraphrases in PlanDOC. At the lexical level, we use synonymous words from WordNet to generate lexical paraphrases. A sample lexical paraphrase for *"The base plan called for one fiber activation at CSA 2100"* is *"The base plan proposed one fiber activation at CSA 2100"* Subcategorizations and alternations from the lexicon are then applied at the syntactic level. After three levels of paraphrasing, each message in PlanDOC has over ten paraphrases on average.

For a specific domain such as PlanDOC, an enormous proportion of a general lexicon like the one we constructed is unrelated and thus not used at all. On the other hand, domain-specific knowledge may need to be added to the lexicon. The problem of how to adapt a general lexicon to a particular application domain and merge domain ontologies with a general lexicon was discussed in [Jing, 1998].

6.4.3 Integration of the Lexicon with a Natural Language Generator

We integrated the lexicon with FUF/SURGE [Elhadad, 1992, Robin, 1994], a unification-based syntactic realizer. This integration makes it possible to reuse major parts of a lexical chooser, which is the component in a generation system that is responsible for mapping semantic input to surface generator input. We show that although the *whole* lexical chooser cannot be made domain-independent, it is possible to reuse a large amount of lexical, syntactic, and semantic knowledge across applications.

In addition, the lexicon bring other benefits to a generation system, including the ability to automatically generate many lexical and syntactic paraphrases and to avoid non-grammatical output. Once integrated into the FUF/SURGE package, the lexicon can be easily used by any generation system which uses FUF/SURGE as its surface generator.

The integration of the lexicon with FUF/SURGE is implemented through incremental unification. The main problems involved include how to represent the lexicon in FUF format, how to unify input with the lexicon incrementally to generate more sophisticated and informative representations, and how to design an appropriate semantic input format so that the integration of the lexicon and FUF/SURGE can be done easily. For details, refer to [Jing et al., 2000].

6.4.4 Word Sense Pruning

We explored using the syntactic and semantic constraints encoded in the combined lexicon for word sense disambiguation. A given word may have n distinct senses and appear within m different syntactic contexts, but typically, not all $n \times m$ combinations are valid. The syntactic context can partly disambiguate the semantic content. For example, when the verb *appear* has a to-infinitive complement, it can only have the sense of "give an impression" (sense 1 in Figure 6.1) out of 8 possible senses. The lexicon entry for the verb *appear*, as shown in Figure 6.1, can be converted to a syntax-semantics restriction matrix, shown in Table 6.1. When *appear* is encountered in a particular syntactic structure, an automatic program can consult the restriction matrix to eliminate senses that can be excluded. In the case of *appear*, only 47 cells of the 8×23 matrix represent possible combinations of syntactic patterns with senses, corresponding to a 74.5% reduction in ambiguity.

For the 5,676 verbs present in the combined lexicon, the average reduction in ambiguity was 36.82% for words with two to four senses, 59.36% for words with five to ten senses, and 73.86% for words with more than ten senses; the overall average for all polysemous words was 47.91%.

This method of word sense pruning has two significant advantages: first, it can be automatically applied, assuming a robust method for parsing the relevant verb phrase context; second, it can be applied to any text since it is domain-independent.

To further restrict the size of the set of valid senses produced, we also explored using domain-dependent, automatically constructed semantic classifications to identify predominant senses of words in a specific-domain. Applying this method and the syntactic constraints method in tandem and intersecting the sense sets produced by them, we can reduce the size of the final tag.

[Jing et al., 1997] described our experiments in verb sense pruning in detail.

6.4.5 Other Applications

The lexicon can also be used for other purposes, such as machine translation. [Baldwin, Bond, and Hutchinson, 1999] proposed a valency dictionary architecture for machine translation. The architecture uses a sense-based dictionary structure to describe monolingual lexicons and a set of transfer links to indicate correspondences between lexicons in different languages. The proposed dictionary structure comprises of, in descending order, the word, sense and frame levels. Our combined lexicon can be easily represented in the proposed structure and used in machine translation.

Subcategorization/Alternation		Sense							
		S1	S2	S3	S4	S5	S6	S7	S8
PP-TO-INF-RS	(sb, −)	+	+				+		
	(sth, −)		+	+				+	
PP-PRED-RS	(sb, −)		+				+		
	(sth, −)		+	+				+	
EXTRAP-TO-NP-S					+				
INTRANS	(sb, −)		+			+			
	(sth, −)		+	+		+			
SEEM-S					+				
SEEM-TO-NP-S					+				
TO-INF-RS	(sb, −)	+							
	(sth, −)								
NP-PRED-RS	(sb, −)	+							
	(sth, −)	+							+
ADJP-PRED-RS	(sb, −)	+							+
	(sth, −)	+							+
ADVP-PRED-RS	(sb, −)	+							+
	(sth, −)	+							+
AS-NP	(sb, −)		+				+		
	(sth, −)		+	+				+	
LOCPP	(sb, −)		+			+			
	(sth, −)		+	+		+			
THERE-INSERTION			+	+		+			
LOCATIVE-INVERSION			+	+		+			

Table 6.1: Valid combinations of syntactic subcategorization /alternations and senses (marked with $+$) for the verb *appear*.

6.5 Discussions

Merging resources is not a new idea; previous work has investigated integration of resources for machine translation and interpretation [Klavans and Tzoukermann, 1990, Knight and Luk, 1994]. Our work differs from previous work in that for the first time, a generation lexicon is built by this technique. Unlike other work that aims to combine resources with similar type of information, we select and combine multiple resources containing different types of information. While others combine lexicons like LDOCE

(Longman Dictionary of Contemporary English) in which the definitions are written in natural language, we chose resources in which the information is well encoded for computer use or manually formated the resource so as to get reliable and usable results. A semi-automatic rather than a fully automatic approach is adopted to ensure accuracy. Corpus analysis information is also linked with information from static resources. Using these measures, we are able to acquire an accurate, reusable, rich, and large-scale lexicon for natural language generation.

The lexicon we have constructed focus on verbs, since we are interested in syntactic properties and such properties are mostly associated with verbs. For other part-of-speeches, such as nouns and adjectives, the properties that we need to focus on while combining multiple resources may be different. Nouns have much less possible complements than verbs; COMLEX have 9 possible complements for nouns compared to 92 complements for verbs, and WordNet does not include complement information for nouns. But there are other features that are important for nouns: for example, whether a noun is aggregate (that is, it can occur as the subject of both definitely singular and definitely plural verbs, as in *"the GROUP have changed their minds"* and *"the GROUP has changed its mind"*, or whether a noun is countable, or whether it can precede a person's name. Such features are encoded in COMLEX. Therefore, if we combine the noun entries in COMLEX with the noun entries in WordNet, we need to include these features in addition to the complement information. Similarly to nouns, adjectives also have less complements than verbs — 14 complements for adjectives in COMLEX compared to 92 complements for verbs — but they have other important features, such as whether an adjective has a comparative or superlative form, or whether it can occur before a quantifier. Such features, in addition to the complement information, need to be considered when adjective entries from multiple resources are combined.

6.6 Conclusion

We have presented research on building a rich, large-scale, and reusable lexicon for generation by combining multiple heterogeneous linguistic resources. Novel semi-automatic transformation and integration were used in combining resources to ensure reliability of the resulting lexicon. The lexicon has been used in many applications, including summarization, traditional natural language generation, word sense disambiguation, and machine translation.

Chapter 7

Putting it All Together

In this chapter, we first describe the sentence extraction module in our summarization system. Our extraction module primarily relies on lexical coherence information to identify key sentences, and also uses other types of information, including *tf* $\times$ *idf* scores, cue phrases, and sentence positions. In Section 7.2, we briefly describe the implementation of the system and give an example to show how a document goes through the system pipeline. Section 7.3 is devoted to evaluation issues. We discuss the techniques that are used for summarization evaluations and present the result of our evaluation experiment. Section 7.4 is concerned with portability issues. We describe our experiments in four different domains to test the portability of our cut-and-paste summarization system.

7.1 The Extraction Module

The role of the extraction module is to identify key sentences in a document. We developed an extraction module that primarily relies on lexical coherence information to identify key sentences and also incorporates other types of information, including *tf* $\times$ *idf* scores, cue phrases, and sentence positions.

7.1.1 The Lexical Links Approach

The extraction module uses the lexical coherence information in a way that is very similar to the lexical links approach used in step 3 of the sentence reduction algorithm (see Section 4.1.2). The only difference is that besides considering the number and types of lexical links, here we also consider the *directions* of those links. We distinguish two directions of lexical links: *forward* and *backward*. A forward link connects a word in the current sentence with a word in a subsequent sentence, while a backward link connects a word from the current sentence with a word in a preceding sentence.

The algorithm for computing lexical links can be summarized as follows. First, content words in each sentence are linked with related words in other sentences of the document. Two words are considered related if they are repetitions, morphologically related, or linked through one of the lexical relations represented in WordNet. The system then computes an importance score for each word in the sentence, based on the number of lexical links it has with other words in the document, the types of the links, and the directions of the links. The formulae for computing the lexical links score for a word w are shown below:

$$ForwardWeight(w) = \sum_{i=1}^{9}(L_i \times ForwardNum_i(w))$$

$$BackwardWeight(w) = \sum_{i=1}^{9}(L_i \times BackwardNum_i(w))$$

$$Score(w) = \max(ForwardWeight(w), BackwardWeight(w))$$

Here, i represents the different types of lexical relations the system considers. The nine types of lexical relations that is considered by the system include repetition, inflection, derivation, synonymy, meronymy (part-of), hypernymy, antonymy, entailment, and causation. We assign a weight to each type of lexical relation, represented by L_i in the formula. Relations such as repetition or inflection are considered more important and are assigned higher weights, while relations such as hypernymy are considered less important and assigned lower weights.

$ForwardWeight(w)$ computes the weight of all the forward links for a given word *w*. $ForwardNum_i(w)$ represents the number of a particular type of lexical links that the word w has with other words in the document. Similarly, $BackwardWeight(w)$ computes the weight of all the backward links for the word *w*.

The final score for word *w*, $Score(w)$, indicates the importance of the word *w*. It is computed as the larger value of the forward weight and the backward weight. The idea is that if a word has many links with other words in the document and the links are mostly in one direction, then the word is probably important since it is likely to start a new topic (if most of the lexical links are forward links) or conclude a topic (if most of the lexical links are backward links). If a word has many links but the links distribute in both directions about equally, then the word is not considered as important as the previous case since it is more likely to be in a middle of a discussion. If a word has few links, then it is considered not important because it is less likely to be related to the main topic.

After computing an importance score for each content word in a sentence, we can compute an importance score for the sentence by adding up the scores for the words in the sentence. This score indicates how important the sentence is based on the lexical coherence information.

7.1.2 Incorporating Other Types of Information

Besides lexical coherence information, the extraction module incorporates three other types of information while determining sentence importance: *tf* $\times$ *idf* scores, cue phrases, and sentence positions. The different types of information are integrated as follows when we rank the importance of sentences:

- If a sentence contains cue phrases, such as *"in summary"* and *"in conclusion"*, and it is at the beginning or at the end of the document, and it has a high score based on lexical links or *tf* $\times$ *idf*, then the sentence is ranked most important.
- If a sentence does not include a cue phrase but it has a high lexical links score or *tf* $\times$ *idf* score, and it appears at the beginning or at the end of the document, it is considered very important.
- If a sentence does not include a cue phrase, and it does not appear at the beginning or at the end of the document, but it has a high score based on both lexical links and *tf* $\times$ *idf*, then it is considered important.
- If a sentence does not include a cue phrase, and it does not appear at the beginning or at the end of the document, but it has a high lexical links score, then it is considered somewhat important.
- If a sentence does not include a cue phrase, and it does not appear at the beginning or at the end of the document, and it has a low lexical links score and a low *tf* $\times$ *idf* score, then it is considered least important.

Sentences are extracted in the order of their importance until the summary reaches the required length.

7.2 Implementation

We have developed a full-fledged summarization system using the cut-and-paste techniques presented in this thesis. The programs were mostly written in the *PERL* language. The main components of the system include:

- Decomposition Module
- Sentence Reduction Module
- Sentence Combination Module
- Extraction Module
- The Large-scale Lexicon

Each component of the system is quite independent of others. The sentence reduction module and sentence combination module can be integrated into other extraction-based summarizers to serve as its generation component. The decomposition module can be used by any summarization system that needs to automatically build training and testing corpora. The large-scale lexicon can be used in various types of applications, as we have shown in Chapter 6.

The tools that we licensed from other research organizations include:

- Syntactic Parser (ESG from IBM)
- Named Entity Recognizer (Alembic from MITRE)
- Coreference Resolution System (DeepRead from MITRE)

7.3 An Example

We show an example from the beginning to the end to illustrate how a document goes through the pipeline of the system and what result is produced at each step.

Figure 7.1[1] shows a newspaper article that tells a story of five men being arrested after they tried to free drug prisoners from a county jail. The input document is first sent to the extraction module, which extracts key sentences. In this example, sentence 1, 2, 5, 10, 11 are selected, as shown in Figure 7.2. For most of automatic summarizers, this is the end of the summarization process. The result shown in Figure 7.2 will be considered as the output summary and presented to readers.

In contrast, our system sends the above result to the sentence reduction module, which removes extraneous information from extracted sentences. The result after reduction is shown in Figure 7.3. In this case, the reduction module removed a temporal phrase from sentence 1, a clause from sentence 2, and the entire sentence 5.[2]

In the next step, the reduction result is sent to the combination module, which merges reduced sentences with other sentences and phrases. The result after sentence combination is shown in Figure 7.4. As we can see, the reduced sentence 2 was merged with sentence 3 by extracting common subject. Note that sentence 3 was not extracted by the extraction module, but since it is closely related to sentence 2, the combination module merged the two sentences. Also, reduced sentence 10 and reduced sentence 11 are merged by adding the connective *and*.

Finally, the sentences after sentence combination are concatenated to produce the summary. A comparison of the extraction-based result, shown in Figure 7.2, and the cut-and-paste result, shown in Figure 7.4, demonstrates that, for this example, the cut-and-

[1] The figures for the example are at the end of this chapter.

[2] Because the reduction program finds that the sentence has a low context importance score, suggesting that it is not very related to the topic and may not be a good candidate for extraction

paste method has improved the conciseness and coherence of the generated summary.

7.4 Evaluation of the Overall System

In previous chapters, we presented the results of numerous experiments that evaluate individual modules of our cut-and-paste summarization system. In this section, we focus on the evaluation of the overall system.

7.4.1 Evaluation Methods

Evaluation of summarization is a difficult issue. We do not yet know what is the best method to evaluate a summarization system. Many techniques have been proposed, but there is no consensus as to which one is superior than all others. The truth of the matter, we believe, is that there is no single best method for summarization evaluation. Each method has its advantages and limitations, and we need to make sensible choices as to which method to use based on the characteristics of an individual summarization system. Next, we briefly describe the techniques that have been used for summarization evaluation.

Evaluation of summarization systems can be intrinsic or extrinsic [Sparck Jones and Galliers, 1996]. Intrinsic methods measure a system's quality; extrinsic methods measure a system's performance in a particular task. Much of the early work in summarization evaluations uses the intrinsic method: the qualities of the summaries are judged by direct human assessment of, for example, informativeness, coverage, or fluency, or by comparing them with an “ideal” summary. In contrast, most of the recent work in evaluation uses the extrinsic approach, also called the task-based approach. In the extrinsic approach, the performance of a summarization system is evaluated based on its helpfulness in performing a particular task, such as information retrieval, text categorization, or news analysis.

The most frequently used intrinsic evaluation technique is the *ideal summary* method [Edmunson, 1969, Paice, 1990, Kupiec, Pedersen, and Chen, 1995, Marcu, 1997, Salton et al., 1997, Ono, Sumita, and Miike, 1994]. Typically, an "ideal" summary is created, either by professional abstractors or by merging summaries provided by multiple human subjects using methods such as majority opinion, union, or intersection. Automatic summaries are then compared with the "ideal" summary. Precision and recall are used to measure the quality of the summary. The main problem with this method is obvious and is mentioned by other researchers [Edmunson, 1969, Paice, 1990, Hand, 1997]: there is no single correct summary. Johnson [Johnson et al., 1993] proposed matching a template of manually generated key concepts with the concepts included in an abstract, but again, there is no single correct template of key concepts and matching of concepts is a fuzzy problem too.

Extrinsic methods evaluate the performance of a summarization system in a given task, such as GMAT test [Morris, Kasper, and Adams, 1992], news analysis [Miike et al., 1994] and information retrieval [Mani and Bloedorn, 1997]. The most significant step in task-based evaluation is the TIPSTER Text Summarization Evaluation (SUMMAC) conducted by the U.S. Government in May 1998 [Mani et al., 1998]. The *text categorization* task is used to evaluate generic summaries; systems were scored on how well the summaries, in lieu of full text, helped users in categorizing documents into different topics. The *ad hoc information retrieval task* was used to evaluate query-based summaries. Time and accuracy were used to measure system performance.

7.4.2 Evaluation Results

The focus of our evaluation is on comparing the quality of the summaries that have been edited by our cut-and-paste generation system with that of the extraction-based summaries, in order to measure whether the cut-and-paste technique improves the quality of generated summaries and to what extent. We conducted an evaluation experiment that is based on human judgment.

In the experiment, three human subjects were asked to compare the quality of extraction-based summaries and their revised versions produced by our sentence reduction and combination modules. We selected 20 documents from the SUMMAC evaluation data collection. Three different automatic summarizers generated a generic summary for each document respectively, producing 60 summaries in total. These summaries were all extraction-based. We then ran our sentence reduction and sentence combination system to revise these extraction-based summaries, producing a revised version for each extraction-based summary. We presented each human subject with the full documents (20 in total), the extraction-based summaries (60 in total), and their revised versions (60 in total) at the same time, and asked them to compare the quality of extraction-based summaries and that of their revised versions. The human subjects were asked to score the conciseness of the summaries based on a scale from 0 to 10 — the higher the score, the more concise a summary is. They were also asked to score the coherence of the summaries based on a scale from 0 to 10. The three human subjects were graduate students in humanities at Columbia University.

The results are shown in Table 7.1 and Table 7.2. System A, B, and C represent the three extraction-based summarizers. For each summarizer, the table shows the average score given by human subjects for the extraction-based summaries and the average score for the revised summaries.

Score	*System A*	*System B*	*System C*	*Average*
Extract	4.6	3.8	4.1	4.2
Revised	8.2	7.8	7.7	7.9
Improvement	3.6	4.0	3.6	3.7

Table 7.1: Result of conciseness comparison.

Score	*System A*	*System B*	*System C*	*Average*
Extract	3.5	4.2	4.0	3.9
Revised	5.9	6.6	6.1	6.2
Improvement	2.4	2.4	2.1	2.3

Table 7.2: Result of coherence comparison.

On average, the extraction-based summaries achieved a score of 4.2 for conciseness, while the revised summaries achieved a score of 7.9 (an improvement of 3.7 on a scale of 10). The improvement for the three systems are 3.6, 4.0, and 3.6 respectively. The revised summaries are on average 41% shorter than the original extraction-based summaries. For summary coherence, the average score for the extraction-based summaries is 3.9, while the average score for the revised summaries is 6.2 (an improvement of 2.3 on a scale of 10). The improvement for the three systems are 2.4, 2.4, and 2.1 respectively. This demonstrates that the cut-and-paste approach is effective in improving the conciseness and coherence of automatic summaries.

We decided not to carry out the SUMMAC style task-based evaluation. From the result of SUMMAC evaluation, we observed that the result from task-based evaluation can have very small correlation with the quality of the generated summaries. For example, in the SUMMAC evaluation, the 14 teams that participated in the text categorization task achieved very close scores. This is despite the fact that the acceptabilities of the summaries, which are scores given by human to indicate whether a summary is "acceptable", ranged widely from 11% to 71%. The lack of correlation between the acceptability of the summaries and the result of text categorization based evaluation is especially problematic for our evaluation purpose. If we want to use data from SUMMAC so that we can compare with other systems, we need to choose the text categorization task because it is the only task used to evaluate generic summaries and our system works on generic summaries. But given the lack of correlation between the acceptability of the summaries and the scores of text categorization based evaluation, it is very possible that our cut-and-paste approach will achieve scores very similar to extraction-based systems, even if it has significantly improved the quality of generated summaries. Given that the result from text categorization based evaluations is not conclusive of the quality of generated summaries and that performing such a large-scale evaluation requires a significant number of human subjects to be involved and a fair amount of funding, we did not perform the task-based evaluation.

7.5 Portability

The research presented in this thesis aims to develop generation techniques for domain-independent summarization. Our system does not assume a specific domain for input documents and it does not rely on domain knowledge to aid the understanding. We tested our system using four different collections to study the portability of the sentence reduction and combination program. The portability of the decomposition program was discussed in Chapter 3.

The main collection we use in the system is the Benton Collection, which contain news reports on telecommunication related issues. We have mentioned this corpus a few times in previous chapters. It was used to train and evaluate the decomposition module, the reduction module, and the combination module.

After training the sentence reduction module and sentence combination module using the Benton collection, we tested the programs on documents from *three other domains: general medical news, legal documents, and travel guides.*

The *medical news articles* were collected from the *HIV/STD/TB Prevention News Update* provided by the Center for Disease Control (CDC) (http://www.cdcnpin.org/news/prevnews.htm). The CDC provides synopses of key scientific articles and lay media reports on HIV/AIDS, other sexually transmitted diseases, and tuberculosis as a public service. The synopses are written by staff writers daily. We collected 20 synopses from the CDC website and downloaded the full text of the articles from the web. We first used the decomposition program to align the synopses with the original documents and build the corpus for testing the sentence reduction and combination program. Our sentence reduction program achieved a success rate of 72% for the medical news articles, compared to 81.3% for the Benton Collection on which it is trained. This shows that when we apply the reduction program to a domain that it is not trained on, there is a decrease in the performance, but the decrease is not significant. One reason for this may be that both collections contain news articles, although they focus on different topics (telecommunication vs. HIV/AIDS). We also checked the coverage of the probability values that were computed from training corpus and that indicate how likely a certain type of phrase is removed. The probabilities were computed using 400 example sentences in the Benton Collection. When we tested the reduction program on 100 sentences from the same Benton collection, we found that the computed probabilities covered 58% of instances in the test corpus. The probability coverage declined to 30% for the medical news.

For sentence combination, we found that the combination operations and combination rules we constructed using the Benton Collection can still apply to the medical news. This suggests that the operations and rules in our combination program are general enough to be used on documents from other domains.

The above results show that once we have trained the sentence reduction and combination module on one collection, it can be used on collections with similar type of text and the performance does not decrease significantly. Of course, we can also use decomposition program to build a large training corpus for the new collection and train the system using the new corpus, which is likely to lead to better performance.

The *legal document* collection contain documents that describe courts' decisions on law suits. The documents in this collection have more specific structure and writing style than newspaper articles. Also, they may contain extremely long sentences. The following is an example sentence in a sample document:

"In an action to recover damages for personal injuries, etc., the defendant Chesebro-Whitman Co. appeals, as limited by its brief, from so much of an order of the Supreme Court, Nassau County (Winslow, J.), dated August 28, 1998, as, upon reargument of an order of the same court dated March 31, 1998, denied those branches of its motion which were for summary judgment dismissing the causes of action to recover damages for negligence and strict products liability insofar as asserted against it."

Experiments on legal documents show that although we can still apply the reduction module and combination module that have been trained on Benton Collection to the legal documents, the performance is much worse. We cannot present quantitative results here since we do not have human-written summaries for comparison. Given that the sentences in this collection tend to have a very fixed structure, we expect that if we can train the reduction and combination system on the examples from the same collection, the performance should improve significantly.

The *travel guides* collection contain documents on Caribbean travel that we downloaded from the Web. The results show that sentence reduction and combination can still apply, but the performance of the system is very sensitive to the errors in the input document. One misspelling of a word may cause the parser not to analyze a sentence correctly, which in turn may lead to incoherent output by the reduction and combination system.

The four collections we use contain documents of two genres: news documents and legal documents. The Benton collection, the medical documents, and the travel guides all contain news articles; the legal document collection contain legal documents with a fixed text structure. The sentence reduction and combination modules work better on the news articles than on the legal documents, partly because the programs were trained on news articles. To improve the performance on documents of a particular genre, we need to retrain the system using documents of that genre. we may also need to include techniques that are specifically developed to handle documents of that specific genre.

TITLE: Five Accused of Trying to Spring Drug Prisoners

Five men reputed to be members of a terrorist squad linked to Colombian drug lords were arrested Friday in what authorities say was a plot to free prisoners from a county jail.

The five were apprehended along Interstate 95, heading south in vehicles containing an array of gear including paramilitary uniforms, a stun gun, knives, two-way radios and smoke grenades, authorities said.

They were charged with interstate transportation in aid of racketeering and could face state charges as well, said U.S. Attorney Bart Daniel.

Agent Fred Verinder of the FBI said law agencies around the nation have received threats that "Colombian drug cartels would use Colombian terrorist squads known as `narco-terrorism' to free Colombians held in American jails."

Juan Carlos Perez, a Cuban national, was in the Charleston County Jail on charges stemming from last year's seizure of 1,069 pounds of cocaine on Hilton Head Island. He goes on trial Monday.

A defendant convicted earlier in that case, Jorge Samuel Cruz of Puerto Rico, was also in the jail awaiting sentencing. Testimony in the trial linked the drugs seized on the island to the Medellin drug cartel in Colombia.

In addition, a Colombian national, Alphonso Parada, was in the jail awaiting trial on state drug charges, authorities said.

Charleston County Sheriff Al Cannon said authorities received a tip last week concerning a possible escape attempt by a prisoner.

The five men arrested were seen outside the jail early Friday morning in two vehicles with Florida license tags.

Among other items seized, authorities who searched the cars found paramilitary uniforms, two-way radios, binoculars, stun guns, mace containers, black baseball caps and ski masks, flares, hunting knives, flashlights and about $10,000 in cash.

The five were arrested without incident, officials said.

Lydia Glover of the U.S. Marshals Service said the agency has increased security and notified jails where other federal prisoners are incarcerated.

Verinder said the FBI has created a special unit to handle such cases "because we're concerned about intelligence reports that this is going to happen more and more as the extraditables are returned to the United States."

The five arrested were Fernando Botero, 32, of Miami; Jesus Walter Jaramillo, 45, of Homestead, Fla.; Pedro Aragon, 41, of New York; Roberto Rego, 47, of Miami; and another individual who authorities said refused to identify himself.

Figure 7.1: Sample input document.

TITLE: Five Accused of Trying to Spring Drug Prisoners

Sentence 1: Five men reputed to be members of a terrorist squad linked to Colombian drug lords were arrested Friday in what authorities say was a plot to free prisoners from a county jail.

Sentence 2: The five were apprehended along Interstate 95, heading south in vehicles containing an array of gear including paramilitary uniforms, a stun gun, knives, two-way radios and smoke grenades, authorities said.

Sentence 5: Juan Carlos Perez, a Cuban national, was in the Charleston County Jail on charges stemming from last year's seizure of 1,069 pounds of cocaine on Hilton Head Island. He goes on trial Monday.

Sentence 10: Charleston County Sheriff Al Cannon said authorities received a tip last week concerning a possible escape attempt by a prisoner.

Sentence 11: The five men arrested were seen outside the jail early Friday morning in two vehicles with Florida license tags.

Figure 7.2: Result after sentence extraction.

TITLE: Five Accused of Trying to Spring Drug Prisoners

Sentence 1: Five men reputed to be members of a terrorist squad linked to Colombian drug lords were arrested *Friday* in what authorities say was a plot to free prisoners from a county jail.

Sentence 2: The five were apprehended along Interstate 95, *heading south in vehicles containing an array of gear including paramilitary uniforms, a stun gun, knives, two-way radios and smoke grenades,* authorities said.

Sentence 5: *Juan Carlos Perez, a Cuban national, was in the Charleston County Jail on charges stemming from last year's seizure of 1,069 pounds of cocaine on Hilton Head Island.*

Sentence 10: Charleston County Sheriff Al Cannon said authorities received a tip last week concerning a possible escape attempt by a prisoner.

Sentence 11: The five men arrested were seen outside the jail *early Friday morning* in two vehicles *with Florida license tags.*

Figure 7.3: Result after sentence reduction (the phrases in *italic* are removed).

TITLE: Five Accused of Trying to Spring Drug Prisoners

Sentence 1: Five men reputed to be members of a terrorist squad linked to Colombian drug lords were arrested in what authorities say was a plot to free prisoners from a county jail.

Sentence 2 + 3: The five were apprehended along Interstate 95 and were charged with interstate transportation in aid of racketeering, authorities said.

Sentence 10 + 11: Charleston County Sheriff Al Cannon said authorities received a tip last week concerning a possible escape attempt by a prisoner and the five men arrested were seen outside the jail in two vehicles.

Figure 7.4: Result after sentence combination.

Chapter 8

Conclusion

In this thesis we have described a new approach to the generation problem in summarization, called *cut-and-paste summarization*. In this approach, the text of summaries is constructed by reformulating the text in the original documents. Our approach is better than a simple extraction-based method since we edit and smooth extracted sentences; it also has advantages over a deep natural language generation based approach since it does not assume a sophisticated semantic representation as input.

8.1 Summary of Contributions

The main contributions of this thesis include:

- **Decomposition of human-written summary sentences.** We developed a Hidden Markov Model based decomposition program for analyzing human-written summaries. Decomposition offers valuable insight into the summary construction process that expert summarizers use and provides much needed corpora for the training and evaluation of automatic summarizers. The Hidden Markov Model solution we proposed for this problem is unique in that it creatively uses estimated probabilities and achieves good performance without requiring a large-scale, annotated training corpus.

- **Sentence reduction techniques.** We developed a sentence reduction algorithm to identify and remove inessential information from extracted sentences. Reduction significantly improves the conciseness of generated summaries and is frequently used by expert summarizers. The algorithm we proposed uses multiple types of knowledge, including linguistic knowledge, probabilities from corpora, and contextual information to determine whether a phrase can be deleted.

- **Sentence combination techniques.** We developed a sentence combination algorithm to merge sentences and phrases. Combination improves the coherence of generated summaries. Like sentence reduction, it is also frequently used by expert summarizers. We identified a number of combination operations, constructed a set of combination rules, implemented the operations using the TAG formalism, and also investigated using machine learning techniques to automatically acquire combination rules.

- **A large-scale, reusable lexicon.** We built a large-scale lexicon by strategically combining multiple, large-scale, heterogeneous resources. The combined lexicon includes rich syntactic and lexical information. It is particularly suitable for language generation applications since the information in the lexicon is indexed at the semantic concept level. It can also be used in many other applications, such as word sense disambiguation, machine translation, and summarization.

- **New sentence extraction techniques.** We developed a sentence extraction algorithm to identify key sentences in a document.[1] We integrated several different approaches that have been used for sentence selection. Our extraction technique uses primarily lexical coherence information to identify key sentences, but it also incorporates other types of information, such as *tf* $\times$ *idf* scores, cue phrases, and sentence positions.

Using the techniques presented in this thesis, I have developed a full-fledged, cut-and-paste text summarization system.

8.2 Future Work

There are a number of directions for future work, including:

1. **Interaction between sentence reduction and sentence combination.** It would be interesting to look at how the reduction decisions should be revised if we consider whether the reduction result will make a good candidate for performing sentence combination at a later stage. For example, reduction might be required to delete a clause it would otherwise not delete to allow the reduced sentence to be merged with other sentences. Reduction and combination are not independent processes; the decisions made by one should in some way influence the decisions made by the other. A feedback architecture between the reduction module and the combination module might be desirable. Thus, reduction can revise its decisions based on the feedback from the combination module.

[1] We should state that developing extraction techniques is not the main focus of the research presented in this thesis.

2. **Improving robustness on ill-formatted text.** Emails, faxes, speech transcripts, OCR results, and many Web pages often contain ill-formatted text. Such text may have numerous spelling errors, it may not have punctuation marks, or it may include ill-structured sentences. How to improve the robustness of our system for this kind of text is an interesting topic. We can either develop a preprocessor that can transform the ill-formatted text into well-formatted text, or make our system accept this kind of text by improving the robustness of the syntactic parser and other tools in the system. Probably we will need to do both to achieve good performance.

3. **Query-based Summarization.** When we summarize a document based on a user's interests, we should aim to include information that is highly relevant to the particular user. How to modify our cut-and-paste system so that it can be used for query-based summarization is an interesting topic. Both sentence reduction and combination module should take a user's interest into account while removing or combining phrases. The challenges associated with this task include representing a user's interests based on his or her query, matching the user's interest to the text in the document, deciding the relevance between a phrase and the query, and determining whether a phrase should be removed or combined with others based on its relevance to the user's interests.

4. **Reformulating the text for other summarization purposes.** In this thesis, we have looked at how to reformulate the text from a document into a text summary that can be read by readers. If a summary is used for a different purpose, the style of the generated summary may need to change accordingly. For example, a summary can be sent to a text to speech synthesizer and provided to a listener.

In speech, we typically prefer sentences to be short and have simple syntactic structures. How to reformulate the text in a document into a a speech summary is an interesting problem. Sentence reduction and combination may still be useful, but we may need to introduce other operations to produce desirable output.

Bibliography

[ANSI, 1997] ANSI. 1997. Guidelines for abstracts. Technical Report Z39.14-1997, NISO Press, Bethesda, Maryland.

[Aretoulaki, 1997] Aretoulaki, M. 1997. Cosy-mats: an intelligent and scalable summarization shell. In *Proceedings of ACL/EVAL'97 Workshop on Intelligent Text Summarization*, pages 74–81, Madrid, Spain.

[Baldwin and Morton, 1998] Baldwin, B. and T. Morton. 1998. Coreference-based summarization. In *Proceedings of the TIPSTER Text Phase III Workshop*, Washington.

[Barzilay and Elhadad, 1997] Barzilay, R. and M. Elhadad. 1997. Using lexical chains for text summarization. In *Proceedings of the ACL/EACL'97 Summarization Workshop*, Madrid, Spain.

[Barzilay, McKeown, and Elhadad, 1999] Barzilay, R., K. R. McKeown, and M. Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 550–557, University of Maryland, Maryland, June.

[Baum, 1972] Baum, L. E. 1972. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process. *Inequalities*, 3:1–8.

[Baum and Petrie, 1966] Baum, L. E. and T. Petrie. 1966. Statistical inference for probabilistic functions of finite state markov chains. *Annals of Mathematical Statistics*, 37(6):1554–1563.

[Benbrahim and Ahmad, 1995] Benbrahim, M. and K. Ahmad. 1995. Text summarisation: The role of lexical cohesion analysis. *The New Review of Document and Text Management*, 1:321–335.

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

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

[Brown, Lai, and Mercer, 1991] Brown, P. F., J. C. Lai, and R. L. Mercer. 1991. Aligning sentences in parallel corpora. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 169–176, Berkeley, California, June.

[Carroll et al., 1998] Carroll, J., G. Minnen, Y. Canning, S. Devlin, and J. Tait. 1998. Practical simplification of English newspaper text to assist aphasic readers. In *Proceedings of the AAAI 1998 Workshop on Integrating Artificial Intelligence and Assistive Technology*, Madison, Wisconsin, July.

[Chandrasekar, Doran, and Srinivas, 1996] Chandrasekar, R., C. Doran, and B. Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, Denmark, August.

[Cohen, 1995] Cohen, W. W. 1995. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*.

[Cohen, 1996] Cohen, W. W. 1996. Learning trees and rules with set-valued features. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*.

[Corston-Oliver and Dolan, 1999] Corston-Oliver, S. H. and W. B. Dolan. 1999. Less is more: eliminating index terms from subordinate clauses. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 349–356, University of Maryland, Maryland, June.

[Cremmins, 1982] Cremmins, E. T. 1982. *The Art of Abstracting*. ISI Press, Philadelphia.

[Dalianis and Hovy, 1993] Dalianis, H. and E. Hovy. 1993. Aggregation in natural language generation. In *Proceedings of the 4th European Workshop on Natural Language Generation*, Pisa, Italy.

[Day et al., 1997] Day, D., J. Aberdeen, L. Hirschman, R. Kozierok, P. Robinson, and M. Vilain. 1997. Mixed-initiative development of language processing systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington D.C.

[Edmundson, 1968] Edmundson, H. P. 1968. New methods in automatic extracting. *Journal of the Association for Computing Machinery*, 16(2):264–285.

[Edmunson, 1969] Edmunson, H. P. 1969. New methods in automatic abstracting. *Journal of the ACM*, 16(2):264–285.

[Elhadad, 1992] Elhadad, M. 1992. *Using Argumentation to Control Lexical Choice: A Functional Unification-Based Approach*. Ph.D. thesis, Department of Computer Science, Columbia University, New York.

[Endres-Niggemeyer et al., 1998] Endres-Niggemeyer, B., K. Haseloh, J. Müller, S. Peist, I. Santini de Sigel, A. Sigel, E. Wansorra, J. Wheeler, and B. Wollny. 1998. *Summarizing Information*. Springer, Berlin.

[Endres-Niggemeyer, Hobbs, and Sparck Jones, 1993] Endres-Niggemeyer, B., J. Hobbs, and K. Sparck Jones. 1993. Summarizing text for intelligent communication. Technical report, Universität des Saarlandes, Dagstuhl, Germany, December. Also available at http://www.ik.fh-hannover.de/ik/projekte/Dagstuhl/Abstract/.

[Endres-Niggemeyer and Neugebauer, 1995] Endres-Niggemeyer, B. and E. Neugebauer. 1995. Professional summarising: no cognitive simulation without observation. In *Proceedings of the International Conference in Cognitive Science*, San Sebastian, May.

[Fidel, 1986] Fidel, R. 1986. Writing abstracts for free-text searching. *Journal of Documentation*, 42(1):11–21, March.

[Grefenstette, 1998] Grefenstette, G. 1998. Producing intelligent telegraphic text reduction to provide an audio scanning service for the blind. In *Working Notes of the AAAI 1998 Spring Symposium on Intelligent Text Summarization*, Stanford University, Stanford, California, March.

[Hand, 1997] Hand, T. F. 1997. A proposal for task-based evaluation of text summarization systems. In *Proceedings of the ACL/EACL'97 Workshop on Intelligent Text Summarization*, pages 31–36, Madrid, Spain.

[Hirst and St-Onge, 1998] Hirst, G. and D. St-Onge. 1998. Lexical chains as representations of context for the detection and correction of malapropisms. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, MA, pages 305–332.

[Hoey, 1991] Hoey, M. 1991. *Patterns of Lexis in Text*. Oxford University Press.

[Jing, 1998] Jing, H. 1998. Usage of wordnet in natural language generation. In *Proceedings of the WorkShop on the Usage of WordNet in Natural Language Processing Systems*, pages 128–134, Université de Montréal, Quebec, Canada, August.

[Jing et al., 1997] Jing, H., V. Hatzivassiloglou, R. Passonneau, and K. R. McKeown. 1997. Investigating complementary methods for verb sense pruning. In *Proceedings of ANLP'97 Lexical Semantics Workshop*, pages 58–65, Washington, D.C., April.

[Jing and McKeown, 1998] Jing, H. and K. R. McKeown. 1998. Combining multiple, large-scale resources in a reusable lexicon for natural language generation. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, volume 1, pages 607–613, Université de Montréal, Quebec, Canada, August.

[Jing et al., 2000] Jing, H., Y. D. Netzer, M. Elhadad, and K. R. McKeown. 2000. Integrating a large-scale, reusable lexicon with a natural language generator. In *Proceedings of the First International Conference on Natural Language Generation*, Mitzpe Ramon, Israel, June.

[Jing and Tzoukermann, 1999] Jing, H. and E. Tzoukermann. 1999. Information retrieval based on context distance and morphology. In *Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 90–96, University of Berkeley, CA, August.

[Johnson et al., 1993] Johnson, F. C., C. D. Paice, W. J. Black, and A. P. Neal. 1993. The application of linguistic processing to automatic abstract generation. *Journal of Document and Text Management*, 1(3):215–241.

[Joshi, 1987] Joshi, A. K. 1987. Introduction to tree-adjoining grammars. In A. Manaster-Ramis, editor, *Mathematics of Language*. John Benjamins, Amsterdam.

[Joshi, Levy, and Takahashi, 1975] Joshi, A. K., L. S. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1).

[Joshi and Schabes, 1996] Joshi, A. K. and Y. Schabes. 1996. Tree-adjoining grammars. In G. Rosenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3. Springer-Verlag, New York, NY, pages 69–123.

[Klavans and Tzoukermann, 1990] Klavans, J. L. and E. Tzoukermann. 1990. The BICORD system: combining lexical information from bilingual corpora and machine readable dictionaries. In *Proceedings of the Thirteenth International Conference on Computational Linguistics*, Helsinki, Finland.

[Knight and Luk, 1994] Knight, K. and S. K. Luk. 1994. Building a large-scale knowledge base for machine translation. In *Proceedings of AAAI'94*.

[Knight and Marcu, 2000] Knight, K. and D. Marcu. 2000. Statistics-based summarization - step one: sentence compression. In *Proceedings of the 17th National Conference of the American Association for Artificial Intelligence*, Austin, Texas.

[Kupiec, Pedersen, and Chen, 1995] Kupiec, J., J. Pedersen, and F. Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th International Conference on Research and Development in Information Retrieval*, pages 68–73, Seattle, Washington.

[Kucera and Francis, 1967] Kucera, H and W. N. Francis. 1967. *Computational Analysis of Present-day American English*. Brown University Press, Providence, RI.

[Levin, 1993] Levin, B. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago, Illinois.

[Macleod and Grishman, 1995] Macleod, C. and R. Grishman, 1995. *COMLEX Syntax Reference Manual*. Computer Science Department, New York University, February.

[Mani and Bloedorn, 1997] Mani, I. and E. Bloedorn. 1997. Multi-document summarization by graph search and matching. In *Proceedings of AAAI'97*, pages 622–628, Providence, Rhode Island.

[Mani, Bloedorn, and Gates, 1998] Mani, I., E. Bloedorn, and B. Gates. 1998. Using cohesion and coherence models for text summarization. In *Working Notes of the AAAI'98 Spring Symposium on Intelligent Text Summarization*, pages 69–76, Stanford, CA.

[Mani, Gates, and Bloedorn, 1999] Mani, I., B. Gates, and E. Bloedorn. 1999. Improving summaries by revising them. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 558–565, University of Maryland, Maryland, June.

[Mani et al., 1998] Mani, I., D. House, G. Klein, L. Hirschman, L. Obrst, T. Firmin, M. Chrzanowski, and B. Sundheim. 1998. The TIPSTER SUMMAC text summarization evaluation final report. Technical Report MTR 98W0000138, The MITRE Corporation.

[Marcu, 1997] Marcu, D. 1997. From discourse structures to text summaries. In *Proceedings of ACL/EACL'97 Workshop on Intelligent Text Summarization*, pages 82–88, Madrid, Spain.

[Marcu, 1999] Marcu, D. 1999. 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, August.

[McCord, 1990] McCord, M., 1990. *English Slot Grammar*. IBM.

[McKeown, Kukich, and Shaw, 1994] McKeown, K. R., K. Kukich, and J. Shaw. 1994. Practical issues in automatic documentation generation. In *Proceedings of the Fourth Conference on Applied Natural Language Processing*, pages 7–14, Stuttgart, Germany, October.

[McKeown and Radev, 1995] McKeown, K. R. and D. R. Radev. 1995. Generating summaries of multiple news articles. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 74–82, Seattle, Washington, July.

[Miller et al., 1990] Miller, G. A., R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. 1990. Introduction to WordNet: an on-line lexical database. *International Journal of Lexicography (special issue)*, 3(4):235–312.

[Miller et al., 1993] Miller, G. A., C. Leacock, R. Tengi, and R. T. Bunker. 1993. A semantic concordance. Cognitive Science Laboratory, Princeton University.

[Morris, Kasper, and Adams, 1992] Morris, A. H., G. M. Kasper, and D. A. Adams. 1992. The effects and limitations of automated text condensing on reading comprehension. *Information Systems Research*, 3(1):17–35.

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

[Ono, Sumita, and Miike, 1994] Ono, K., K. Sumita, and S. Miike. 1994. Abstract generation based on rhetorical structure extraction. In *Proceedings of the 15th International Conference on Computational Linguistics*, volume 1, pages 344–384, Kyoto, Japan.

[Paice, 1990] Paice, C. D. 1990. Constructing literature abstracts by computer: techniques and prospects. *Information Processing and Management*, 26(1):171–186.

[Paice and Johns, 1993] Paice, C. D. and A. P. Johns. 1993. The identification of important concepts in highly structured technical papers. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[Rabiner, 1989] Rabiner, L. R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[Radev, 1999] Radev, D. 1999. *Language Reuse and Regeneration: Generating Natural Language Summaries from Multiple On-Line Sources*. Ph.D. thesis, Department of Computer Science, Columbia University, New York.

[Robin, 1994] Robin, J. 1994. *Revision-Based Generation of Natural Language Summaries Providing Historical Background: Corpus-Based Analysis, Design, Im-*

plementation, and Evaluation. Ph.D. thesis, Department of Computer Science, Columbia University, New York.

[Salton et al., 1994] Salton, G., J. Allan, C. Buckley, and A. Singhal. 1994. Automatic analysis, theme generation, and summarization of machine readable texts. *Science*, 264(5164):1421–1426, June.

[Salton et al., 1997] Salton, G., A. Singhal, M. Mitra, and C. Buckley. 1997. Automatic text structuring and summarization. *Information Processing and Management*, 33(2):193–208.

[Shaw, 1995] Shaw, J. 1995. Conciseness through aggregation in text generation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (Student Session)*, pages 329–331.

[Teufel and Moens, 1997] Teufel, S. and M. Moens. 1997. Sentence extraction as a classification task. In *Proceedings of the ACL/EACL'97 Workshop on Intelligent Scalable Text Summarization*, pages 58–65, Madrid, Spain.

[Thurber, 1924] Thurber, S., editor. 1924. *Précis Writing for American Schools*. The Atlantic Monthly Press, INC., Boston.

[Viterbi, 1967] Viterbi, A. J. 1967. Error bounds for convolution codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269.