NAIST-IS-DD9761210

Doctoral Dissertation

Machine Learning Approaches to Rhetorical Parsing and Open-Domain Text Summarization

Tadashi Nomoto

December, 2004

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of Engineering

Tadashi Nomoto

Thesis Committee:

Professor Yuji Matsumoto	(Supervisor)
Professor Shunsuke Uemura	(Member)
Professor Shin Ishii	(Member)

Machine Learning Approaches to Rhetorical Parsing and Open-Domain Text Summarization

Tadashi Nomoto

Abstract

The present thesis primarily concerns the use of machine learning for rhetorical parsing and open-domain text summarization. Chapter 1 sets a general backdrop on text summarization and its subfield, rhetorical parsing, and defines the area of investigation. Chapters 2 through 9 form the core of the thesis, developing each theme in great depth, for which we will give a brief overview below. (Throughout the thesis, we talk about *extractive* summarization, meaning that we create a summary by putting together bits and pieces, usually, sentences extracted from text.)

In chapters 2 through 5, we motivate and develop a novel approach to rhetorical parsing based on the decision tree (DT) learning, which one could adapt for any genre and language given a training corpus. (Unless stated otherwise, DT here and below means Quinlan's C4.5 with default settings.) An important goal of rhetorical parsing is to recover rhetorical structure of text for potential use with text summarization. Performance of our approach is evaluated using an hand-annotated corpus of Japanese newspaper articles. Also some problems with annotating with rhetorical information such as the variability of human judgments on labeling are noted and discussed.

In addition some refinements are made on the DT learning itself by appeal to the minimum description length principle (MDL) and active learning. Evaluation is done using the same data as above. We also look into how a DT harnessed with MDL (DT/MDL), compares in performance with AdaBoosted DT.

Due to poor results with the linguistically motivated paradigm that previous chapters represent, we turn an eye on non-linguistic approaches to summarization. Chapter 6 explores an unsupervised paradigm for text summarization. We develop there what we call the diversity based summarization or DBS, which consists in the K-means clustering (again extended with MDL) and a simple sentence ranking scheme. A new evaluative scheme for summarization (which we call the information-centric approach to evaluation of summaries, or ICE) is also proposed with an eye to providing an objective assessment

of the utility of machine generated summaries. Evaluation is conducted using a publicly available corpus known as BMIR-J2.

Then we proceed to the issue of modeling human created summaries in the DBS paradigm. We compare performance of DBS and DT- (and DT/MDL-) based summarizers trained on a human-annotated corpus. Curiously enough, it is found that DBS closely rivals and sometimes outperforms DT- and DT/MDL- based summarizers – which we collectively call 'DT(/MDL)' here – when tested on those annotations which judges tend to disagree on, but falls behind DT(/MDL) on annotations for which there is a strong agreement among judges. The result suggests that there are some useful, i.e., DT-learnable, patterns in annotations for which people have a more or less same idea about what they should be like. While DT(/MDL) is apparently able to exploit patterns to its advantage, DBS, being unsupervised, is not able to perform as well as when it is run on annotations with varying judgments. Which however points to an integration of DT(/MDL) with DBS as a possible alternative to DBS as the combine should then be able to take into account the regularity as well as variability of human summaries, an issue that engages us in subsequent chapters, where we consider other variations of DT. We argue that taking into account both properties indeed leads to a better performing summarizer.

Finally, we look at curious regularities in the way people vote for summary sentences when asked to pick up those they consider important or summary-worthy. Texts from a news wire domain typically show that initially occurring sentences are popularly voted or preferred for summary sentences while those occurring later in text decidedly get less popular. Texts from a column domain, on the other hand, exhibit a somewhat different pattern, showing that sentences occurring towards the end are as much favored by people as those occurring text-initially. We argue that the distribution of votes for summary sentences, which we call 'DOV,' has some shape specific to a domain, and propose a particular approach that directly exploits DOVs by way of Bayesian modeling. We show that the Bayesian model provides a significant leverage over approaches based on pattern classifiers such as C4.5, Adtree, Kstar, Naive Bayes, etc.

Keywords: Machine Learning, Text Summarization, MDL, Bayes Model, Parsing, Rhetorical Structure

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

/newpage

Chapter 1 Background and Overview

In this chapter, we go over some background on automatic text summarization, and give a brief overview of what will come in each chapter. Text summarization is generally characterized as a scientific inquiry into identifying or extracting a small portion of a text, usually sentences, that serves as a surrogate of that text, providing the gist of that text. Throughout the thesis, we hold to the view of summarization as a process of extracting a subtext for use as a surrogate of its full-length version.

Consider Table 1-1. It lists some of major research paradigms in automatic text summarization and their features. Luhn (1958),Edmundson (1969) and Pollock and Zamora (1999) represent the classical paradigm which typically relies on the use of superficial information such as sentence location, term frequency and cue words for identifying sentences that may be able to serve as a summary. Edmundson (1969), for instance, uses a simple weighting scheme, which involves a weighted combination of information of the sorts mentioned above, to determine relative significance of sentences, and select those ranking highest. In the learning paradigm, on the other hand, one makes an explicit use of human supplied information such as whether or not a given sentence is to be included in a summary.

Kupiec et al. (1995) are first to approach automatic summarization from the learning paradigm. What distinguishes their work is their view of summarization as a statistical classification problem. They devised a simple Bayesian classifier to estimate the probability that a given sentence is included in a summary. A summary can then be generated by selecting sentences with top probabilities.

Marcu (1999b), Boguraev and Kennedy (1999), and Barzilay and Elhadad (1999) represent another research direction in summarization. They typically advocate linguistically motivated models of discourse for text summarization. Marcu (1999b), for instance, attempts to recover a rhetorical structure tree à la Mann and Thompson (1987a) out of a text and exploit that tree for determining which sentence to extract for a summary. In chapters 2 to 5, we pursue the idea of using a linguistic model of discourse for summarization and point out some inherent problems with this approach. In particular, we examine the assumption that importance or *summary-worthiness* of a sentence can be determined

Table 1-1 Major Paradigms in Summarization Research

Paradigm	Features	Papers
Classical	uses location,frequency, cue. non-learning	(Luhn, 1958; Edmundson, 1969; Pollock & Zamora, 1999)
Learning (Statistical)	learns human supplied data. uses features (e.g., location, frequency, cues, etc.)	(Kupiec, Pedersen, & Chen, 1995; Aone, Gorlinsky, Larsen, & Okurowski, 1999; Nomoto & Matsumoto, 1997; Zechner, 1996; Gong & Liu, 2001; Nomoto & Matsumoto, 2001b)
Linguistic	exploits discourse structure	(Miike, Itoh, Ono, & Sumita, 1994; Marcu, 1999b; Boguraev & Kennedy, 1999; Barzilay & Elhadad, 1999)

on the basis of rhetorical structure associated with a text. We empirically examine the assumption by modeling it in the supervised learning framework and identify difficulties with the approach and with the theoretical assumption that the approach adopts. Faced with the problems, we turn in chapter 6 to a radically different assumption that summary-worthiness of a sentence can be determined without reference to rhetorical structure and even without supervision by humans. We empirically compare the two assumptions using a set of data created from a natural language corpus.

In chapter 2 we talk about discourse parsing. The goal there is to discover rhetorical dependencies among sentences that make up the discourse, for a possible application in text summarization. A collection of news articles from a Japanese economics daily are manually marked for dependency relations and used as a training and testing corpus. We construct a C4.5 based parser to model rhetorical dependencies in text. We further study effects of features such as clue words, distance and similarity on performance of the discourse parser.

In chapter 3, we discuss some issues related to creating a corpus for the rhetorical analysis. We introduce Ichikawa's theory of discourse (ITDR) largely because it allows for the reduction of annotation labor through simplifying assumptions it makes about Japanese discourse. We report poor agreement among humans judges on labeling with ITDR, and discuss improving agreement through some modification of ITDR.

Chapter 4 addresses the question of how one might reduce the amount of training data for discourse parsing. The issue of reducing training data is largely motivated by practical concerns such as reducing work load for labeling, and a rapid system deployment. It is arguably the case that creating a training corpus with discourse labels involves a great deal of human efforts.

To that end, we work on two kinds of approaches here: one is what we call the *sampling centric* approach (SCA) and the other *classifier centric* approach (CCA). In SCA, one seeks to reduce data by selecting only those one finds useful based on some criterion, while in CCA, one attempts to reduce amounts of data by using a strong classifier with the ability to generalize itself on a small amount of data. As a representative SCA, we consider a selective sampling method called the *committee-based sampling* (CBS) (Dagan & Engelson, 1995; Engelson & Dagan, 1996), while we implement CCA through a DT extended to support MDL.

We compare performance of SCA and CCA using a data set created from a Japanese newspaper corpus. Chapter 5 examines two well-known enhancements to a learning algorithm, namely MDL and boosting, and compares their respective effectiveness on the analysis of discourse, in particular, the task of identifying rhetorical relations in text. We create a data set containing 1,736 news paper articles. Each article is hand-annotated with rhetorical relations. We apply the two enhancements to C4.5 and find out how DT/MDL and boosted DT compare in performance.

Chapter 6 shifts the focus from supervised to unsupervised paradigm. This shift is motivated by a concern that the supervised approach to rhetorical parsing tends to perform poorly presumably due to inconsistencies or variability in annotations produced by humans. Since the whole business of rhetorical parsing is to provide the rhetorical analysis to be exploited for summarization, poor performance of parsing leads to a degraded performance of whatever summarizer we may want to use. Which gave us enough reason to explore an alternative avenue for text summarization, desirably one which does not rely on human annotations. One such is what we call a diversity based approach to summarization (DBS), which we spend most of time discussing in the chapter. It is clustering based and operates by extracting sentences representative of various topics discussed in the text.

The problem is, however, as was the case with annotation with rhetorical relations, it is notoriously difficult to establish a unique summary or extract for a given text, as

people tend to disagree on what they want to see in a summary. We respond to this lack-of-uniqueness issue by proposing what we call the *information-centric* evaluation (ICE), where the quality of a summary is judged not in terms of how well it matches its human-created version but in terms of how well it serves as a surrogate of the source document in IR tasks such document retrieval and text categorization. We evaluate performance of the diversity based approach using a data set known as BMIR-J2 within the ICE framework.

Chapter 7 continues with the theme of the diversity based summarization from a different perspective. In particular, we deal with the question of how closely the diversity approach models human judgments on summary extraction. We create a test data by asking human subjects to extract part of a text as a summary. We also make a comparison between a decision tree based summarizer and the diversity based summarizer.

In chapter 8, we will be talking about combining DT and DBS. The idea is motivated by results from the previous chapter, which indicate that DT works better than DBS when there is a strong tendency for people to prefer a particular set of sentences for a summary, the converse is true when they don't. Since people may or may not agree on their preferences for summary sentences, we may expect to do well in both situations by combing DT and and DBS. Whether or not it is indeed the case is a question that we try to answer in this chapter.

Chapter 9 is an interesting departure from previous chapters. In contrast to approaches discussed earlier, which are mostly discriminative, i.e., classificatory, we will work here with a Bayesian approach to summarization. It takes seriously the distribution of preferences people have for summary sentences, which we believe to have a shape specific to a given domain. What we aim to do in the chapter is to build a summarizer based on a probabilistic model of preferences.

Chapter 2 Rhetorical Parsing and Decision Tree Learning

2.1 Introduction

Attempts to the automatic identification of a structure in discourse have so far met with the limited success in the computational linguistics literature. Part of the reason is that, compared to sizable data resources available for parsing research such as the Penn Treebank (Marcus, Santorini, & Marcinkiewicz, 1993), large corpora annotated for discourse information are hard to come by. Researchers in discourse usually work with a corpus of a few hundred sentences (Kurohashi & Nagao, 1994; Litman & Passonneau, 1995; Hearst, 1994). The lack of a large-scale corpus prevented us from making general and reliable statements about results of discourse studies.

The chapter describes an effort to collect and exploit a large corpus of natural language data for computational research on discourse, in particular, rhetorical parsing. We begin by collecting a relative large corpus from a financial newspaper and manually annotating it with rhetorical information. It is comprised of 645 articles, totalling 12,770 sentences and 5,352 paragraphs, which is relatively large compared to previous similar efforts in the discourse research. We annotate each article with what we call rhetorical dependency relations. We then build a generic discourse parser based on the C4.5 decision tree algorithm (Quinlan, 1993), which is trained and evaluated on the corpus created. The idea of the rhetorical parser here is inspired by Haruno (1997)'s work on statistical sentence parsing.

The chapter is organized as follows. Section 2.2 presents a brief review of Rhetorical Structure Theory (Mann & Thompson, 1987a), which sets a linguistic background for the discussion on rhetorical parsing. Section 2.3 presents general ideas about statistical parsing as applied to discourse, in particular, text. After a brief introduction to the decision tree learning, we discuss how to embed it within a statistical parsing framework. Section 2.4 describes in detail the procedure and evaluation of the current approach.

Table 2-1 An excerpt from a news article.

Music giant sets sights on Internet

Britain's EMI Group and Time Warner Inc. said Monday they were merging their music businesses to create the world's top record company, worth $20 billion, with a powerful presence on the Internet.

The new music giant, Warner EMI Music, will bring together a star-studded roster of artists and catalogues in a 50-50 joint venture.

The U.S. group will have management control of the new combine, which will have a global market share of around 20 percent and annual sales of $8 billion.

Asahi Evening News, Jan. 25, 2000.

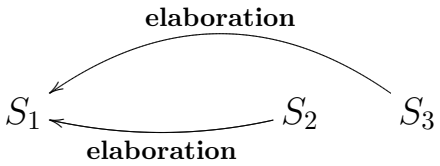

Figure 2-1 Schematic representation of rhetorical relations

2.2 Rhetorical Structure Theory: A Review

Since the present approach to rhetorical parsing draws heavily on previous work in text linguistics (Mann & Thompson, 1987a; Ichikawa, 1990), it would be in order to review some established work in the field.

In general, text linguistics is concerned with finding regularities in relations among sentences or text fragments spanning multiple sentences. In some instances, relations between sentences are explicitly signalled by connectives such as *because*, *however*, *therefore*, etc., in English. However, more often than not, they are not marked by any linguistic devices. Consider, for instance, a news article excerpt in Table 2-1. It has three paragraphs. While there is no explicit cues indicating any discourse relation, it is easy to see that the second and third paragraph stand in a particular relation to the leading paragraph; that is, both of them are an elaboration on the first paragraph, giving complementary information on the

Table 2-2 RST relations (Mann & Thompson, 1987a)

Circumstance	Antithesis and Concession
Solutionhood	Anthithesis
Elaboration	Concession
Enablement and Motivation	Condition and Otherwise
Enablement	Condition
Motivation	Otherwise
Evidence and Justify	Interpretation and Evaluation
Evidence	Interpretation
Justify	Evaluation
Relations of Cause	Restatement and Summary
Volitional Cause	Restatement
Non-Volitional Cause	Summary
Volitional Result	Other Relations
Non-Volitional Result	Sequence
Purpose	Contrast

event described in the first paragraph. Schematically, we have the situation like Figure 2-1, where S_n represents the n-th paragraph of the article.

Notice, moreover, that the order in which they appear is important for interpreting the article; exchanging the first and the second paragraphs makes the article a non-sequitur, while replacing the second with third does not change the general claim of the text as a whole.

Rhetorical Structure Theory or simply RST (Mann & Thompson, 1987a) represents a major attempt to construct a linguistic theory of the rhetorical organization of naturally occurring English texts. It subscribes to the view that a text is a set of sentences arranged not haphazardly, but in such a way as to achieve some rhetorical goals such as informing, persuading, motivating, etc. A rhetorical relation according to RST is typically an relation that holds between two text spans, unbroken non-overlapping linear fragments of text. One of the text spans is called the *nucleus* and the other the *satellite*. RST recognizes the kinds of relations listed in Table 2-2. The circumstance relation holds if the satellite provides a temporal or spatial framework in which to interpret the nucleus; the solutionhood relation holds if the satellite provides some solution to the problem presented in the nucleus; the

Table 2-3 Circumstance relation (Mann & Thompson, 1987a)

1. Probably the most extreme case of Visitors Fever I have ever witnessed was a few summers ago
2. when I visited relatives in the Midwest.

Table 2-4 Solutionhood relation from an insurance advertisement on the Web.

1. Why Choose Between Service and Price?
2. Safe drivers can save up to 15%
3. and get expert claims advice with The American Express Property Casualty companies.
4. Save even more with our multi-policy discount.
5. Get a free online quote.

---

elaboration relation holds if the satellite provides some additional detail for the nucleus. Now it is important to note that rhetorical relations are generally asymmetrical; if A is elaboration for B, then B is not elaboration for A. (Recall our discussion on the article in Table 2-1, where we mentioned that replacing the leading paragraph with the second makes the text nonsensical, which is a case in point.)

Table 2-3 gives an example of circumstance relation from (Mann & Thompson, 1987a). The clause "Probably ..." is a nucleus and the when clause a satellite, which sets a temporal framework in which to interpret the initial clause. Schematically, we have the situation depicted in Figure 2-2, where S_2 represents the second clause and S_1 the nucleus.

Table 2-4 illustrates an solutionhood relation, which somewhat differs from the previous example in that a set of sentences following the initial nucleus clause "Why ... " collectively

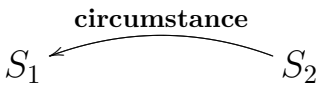

Figure 2-2 Schematic representation of circumstance relation

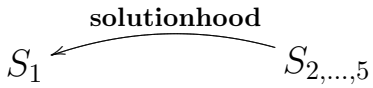

Figure 2-3 Schematic representation of solutionhood relation

form a satellite, as schematically represented in Figure 2-3. S_1 refers to the initial clause, which is a nucleus, and $S_{2,...,5}$ sentences 2 through 5.

As for elaboration relation, the reader is referred to Table 2-1, which provides a good illustration of the relation, and also to the discussion thereof.

RST makes several strong claims about what the rhetorical organization of text should look like as listed below:

completeness	The rhetorical structure of a text consists of a set of rhetorical relations, and there is one "root" relation that spans the entire text.
connectedness	Each text span is part of some rhetorical relation.
uniqueness	Each rhetorical relation accounts for a different set of text spans.
adjacency	One text span does not overlap another text span.

Thus RST demands in effect that a rhetorical analysis be a tree covering the entire text.

The realization that a rhetorical relation is asymmetrical and one member of the relation is more central to the function of the text than the other is one of the most important contributions of RST to computational as well as text linguistics. Indeed, in the natural language processing community, there has been a strong expectation that asymmetries of rhetorical relations can be exploited to provide useful information for text summarization and information extraction.

The problem, however, is that at present there is no robust method available for identifying such abstract relations. The purpose of the present chapter is to explore the use of machine learning techniques to furnish such a method.[1]

[1]Throughout the rest of the chapter, we use "discourse relation" and "rhetorical relation", interchangeably.

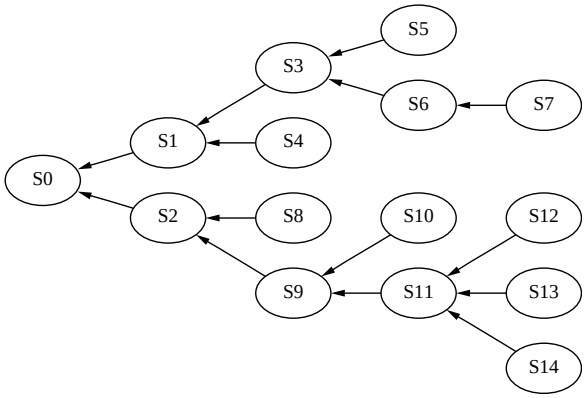

Figure 2-4 A discourse tree. 'S' denotes a sentence.

2.3 Statistical Approach to Rhetorical Parsing

The foremost job of parsing is to find whatever dependencies there are among minimal units that constitute a particular linguistic object. In the context of rhetorical parsing, we view minimal units as corresponding to sentences, and an linguistic object to a discourse or a text. So a dependency we are looking for is a rhetorical relation between a pair of sentences, where the interpretation of a satellite sentence literally depends on that of the nucleus, as discussed above.

Then the task of a discourse parser is to take as inputs a discourse, or a set of sentences that make up a discourse and to produce as output a parse, or a set of dependency relations (which may give rise to a tree-like structure as in Figure 2-4). In statistical parsing, this could be formulated as a problem of finding a best parse with a model $P(T \mid D)$, where T is a set of dependencies and D a discourse.

$$T_{best} = \mathbf{arg\ max}_T P(T \mid D)$$

T_{best} is a set of dependencies that maximizes the probability $P(T \mid D)$. Further, we assume that a discourse D is a set of sentences marked for some pre-defined set of features $F = \{f_1, \ldots, f_n\}$. Let $\mathbf{C}_F(S_1)$ be a characterization of sentence S_1 in terms of a feature set F. Then for $D = \{S_1, \ldots, S_m\}$, $\mathbf{C}_F(D) = \{\mathbf{C}_F(S_1), \mathbf{C}_F(S_2), \ldots, \mathbf{C}_F(S_m)\}$. Let us assume that:

$$P(T \mid D) = \prod_{A \leftarrow B \in T} P(A \leftarrow B \mid \mathbf{C}_F(D)).$$

Table 2-5 An illustration: hotel preferences. 'Bath/shower' means a room has a bath, a shower or none. 'Time' means the travel time in min. from an airport. 'Class' indicates whether a particular hotel is a customer's choice.

	bath/shower	time	room rate	class
1	bath	15	expensive	no
2	shower	20	inexpensive	no
3	shower	10	inexpensive	yes
4	bath	15	moderate	yes
5	bath	25	moderate	yes
6	none	20	inexpensive	no
7	shower	50	inexpensive	no

'$A \leftarrow B$' reads like " sentence B is dependent on sentence A", where $A, B \in \{S_1, \ldots, S_m\}$. The probability that T is an actual parse of discourse D is estimated as the product of probabilities of its member dependencies when a discourse has a representation $\mathbf{C}_F(D)$. We make a usual assumption that member dependencies are probabilistically independent.

2.3.1 Decision Tree Model

A general framework for discourse parsing described above is thus not much different from that for statistical sentence parsing. Differences, however, lie in a makeup of the feature set F. Rather than to use information on word forms, word counts, and part-of-speech tags as in much research on statistical sentence parsing, we exploit as much information as can be gleaned from a discourse, such as lexical cohesion, distance, location, and clue words, to characterize a sentence. Therefore it is important that you do not end up with a mountain of irrelevant features.

A decision tree method represents one of approaches to classification problems, where features are ranked according to how much they contribute to a classification, and models are then built with features most relevant to that classification. Suppose, for example, that you work for a travel agency and want to find out what features of a hotel are more important for tourists, based on data from your customers like Table 2-5. With decision tree techniques, you would be able to tell what features are more closely associated with customers' preferences.

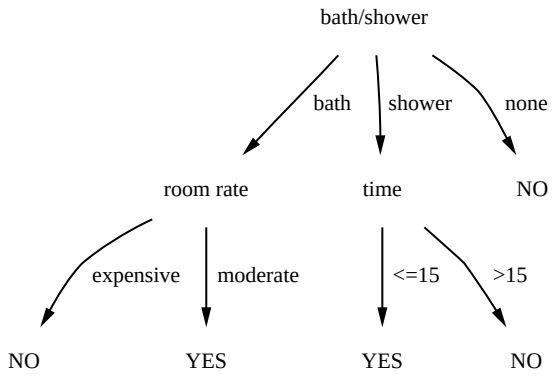

Figure 2-5 A decision tree for the hotel example.

The aim of the decision tree approach is to induce rules from data that best characterize classes. A particular approach called C4.5 (Quinlan, 1993), which we adopt here, builds rules by recursively dividing the training data into subsets until all divisions contain only single class cases. In which subset a particular case is placed is determined by the outcome of a 'test' on that case. Let us explain how this works by way of the hotel example above. Suppose that the first test is "bath/shower", which has three outcomes, `bath`, `shower`, and `none`. Then the data set breaks up into three groups, {1,4,5} (`bath`), {2,3,7} (`shower`), and {6}(`none`). Since the last group {6} consists of only a single case, there is no further division of the group. The `bath` group, being a multi-class set, is further divided by a test "room rate", which produces two subdivisions, one with {1} (`expensive`), and the other with {4,5} (`moderate`). Either set now consists of only single class cases. For the `shower` group, applying the `time` test(`<=15`) would produce two subsets, one with {3}, and the other with {2,7}.[2] Either one now contains cases from a single class. A decision tree for divisions we made is shown in Figure 2-5.

Now compare a hand-created decision tree in Figure 2-5 with one in Figure 2-6, which is generated by C4.5 for the same data. Surprisingly, the latter tree consists of only one test node. This happens because C4.5 ranks possible tests, which we did not, and apply one that gives a most effective partitioning of data based on information-theoretic criteria known as the *gain criterion* and the *gain ratio criterion* (See Appendix A.1). The intuitive idea behind the criteria is to prefer a test with a least entropy, i.e., a test that partitions

[2]Here we choose a midpoint between 10 and 20 as in C4.5.

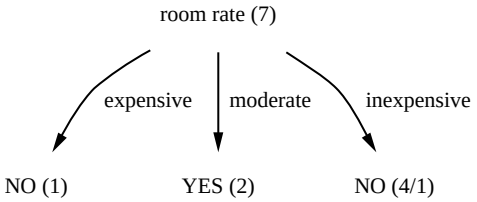

Figure 2-6 A tree for the hotel example by C4.5. Figures in parentheses indicate the number of cases that reach relevant nodes. A figure after a slash, e.g. (4/1), indicates the number of misclassified cases.

data in such a way that a particular class may become dominant for each subset it creates. Thus a feature that best accounts for a class distribution in data is always chosen in preference to others. For the data in Table 2-5, C4.5 determined that the test `room rate` is a best class identifier and everything else is irrelevant to identifying the classes. All that one needs to account for the class distribution in Table 2-5 turn out to be just one feature. So we might just as well conclude that the customers are just interested in the room charge when they pick up a hotel.

A benefit of using the decision tree method is that it enables us to identify relevant features for classification and disregard those that are not relevant, which is particularly useful for a task such as ours, where a large number of features are potentially involved and their relevance to classification is not always known.

2.3.2 Parsing with Decision Tree

As we mentioned in section 2.3, we define discourse parsing as a task of finding a best tree T, or a set of dependencies among sentences that maximizes $P(T \mid D)$.

$$T_{best} = \mathbf{arg\ max}_T P(T \mid D)$$

$$P(T \mid D) = \prod_{A \leftarrow B \in T} P(A \leftarrow B \mid \mathbf{C}_F(D)).$$

What we do now is to equip the model with a feature selection functionality. This can be done by assuming:

$$P(A \leftarrow B \mid \mathbf{C}_F(D)) = \frac{P(A \leftarrow B \mid \mathbf{C}_F(D), \mathbf{DT}_F)}{\sum_{X<B} P(X \leftarrow B \mid \mathbf{C}_F(D), \mathbf{DT}_F)} \tag{2-1}$$

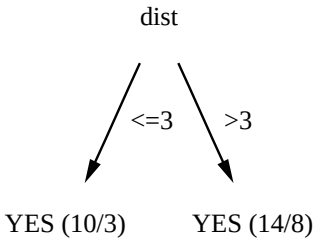

Figure 2-7 A hypothetical decision tree.

$\mathbf{DT}_F$ is a decision tree constructed with a feature set F by C4.5. '$X < B$' means that X is a sentence that precedes B.[3] $P(X \leftarrow Y \mid \mathbf{C}_F(D), \mathbf{DT}_F)$ is the probability that sentence Y depends on sentence X under the condition that both $\mathbf{C}_F(D)$ and $\mathbf{DT}_F$ are used. We estimate P, using *class distributions* from the decision tree $\mathbf{DT}_F$. For example, we have numbers in parentheses after leaves in the decision tree in Figure 2-6. They indicate the number of cases that reach a particular leaf and also the number of misclassified cases. Thus a leaf with the label `inexpensive` has the total of 4 cases, one of which is misclassified. This means that we have 3 cases correctly classified as "NO" and one case wrongly classified. Thus a class distribution for "NO" is 3/4 and that for "YES" is 1/4. In practice, however, we slightly correct class frequencies, using Laplace's rule of succession, i.e., $x/n \rightarrow x + 1/n + 2$.

Now suppose that we have a discourse $D = \{\ldots, S_i, \ldots, S_j, \ldots, S_k, \ldots\}$ and want to know what S_i depends on, assuming that S_i depends on either S_j or S_k. To find that out involves constructing $\mathbf{C}_F(D)$ and $\mathbf{DT}_F$. Let us represent sentences S_j and S_k in terms of how far they are separated from S_i, measured in sentences. Suppose that $dist(S_j) = 2$ and $dist(S_j) = 4$; that is, sentence S_j appears 2 sentences behind S_i and S_k 4 sentences behind. Assume further that we have a decision tree constructed from data elsewhere that looks like Figure 2-7.

With $\mathbf{C}_F(D)$ and $\mathbf{DT}_F$ at hand, we are now in a position to find $P(A \leftarrow B \mid \mathbf{C}_F(D))$,

[3]Note that here we are in effect making a claim about the structure of a discourse, namely that a sentence modifies one that precedes it. Changing it to something like '$X \in D, X \neq B$' allows one to have forward as well as backward dependencies.

for each possible dependency $S_j \leftarrow S_i$, and $S_k \leftarrow S_i$.

$$\begin{aligned} P(S_j \leftarrow S_i \mid \mathbf{C}_{dist}(D), \mathbf{DT}_{dist}) \\ &= (10 - 3 + 1)/(10 + 2) \\ &= .67 \end{aligned}$$

$$\begin{aligned} P(S_k \leftarrow S_i \mid \mathbf{C}_{dist}(D), \mathbf{DT}_{dist}) \\ &= (14 - 8 + 1)/(14 + 2) \\ &= .44 \end{aligned}$$

Since S_i links with either S_j or S_k, by Equation 2-1, we normalize the probability estimates so that they sum to 1.

$$P(S_j \leftarrow S_i \mid \mathbf{C}_{dist}(D)) = .67/(.67 + .44) = .60$$

$$P(S_k \leftarrow S_i \mid \mathbf{C}_{dist}(D)) = .44/(.67 + .44) = .40$$

Recall that class frequencies are corrected by Laplace's rule. Let $T_j = \{S_j \leftarrow S_i\}$ and $T_k = \{S_k \leftarrow S_i\}$ Then $P(T_j \mid D) > P(T_k \mid D)$. Thus $T_{best} = T_j$. We conclude that S_i is more likely to depend on S_j than S_k.

2.3.3 Features

The following list a set of features we used to encode a discourse. As a convention, we refer to a sentence for which we like to find a dependency as 'B', and a sentence preceding 'B' as 'A'.

`<DistSen>` records information on how far ahead A appears from B, measured in sentences.

$$\frac{\#S(B) - \#S(A)}{Max\_Sen\_Distance}$$

'$\#S(X)$' denotes an ordinal number indicating the position of a sentence X in a text, i.e., $\#S(k\text{th\_sentence}) = k$. '*Max_Sen_Distance*' denotes a distance, measured in sentences, from B to A, when B occurs farthest from A, i.e., $\#S(\text{last\_sentence\_in\_text}) - 1$. `DistSen` thus has continuous values between 0 and 1. We discard texts which contain no more than one sentence.

`<DistPar>` is defined similarly to `DistSen`, except that the distance is measured in paragraphs.

$$\frac{\#Par(B) - \#Par(A)}{Max\_Par\_Distance}$$

'$Par(X)$' is a paragraph that contains a sentence X, and '$\#Par(X)$' denotes an ordinal number of $Par(X)$. '*Max_Par_Distance*' is a maximal distance one could have between two paragraphs in a text, that is, $\#Par$(last_sentence_in_text) $- 1$.

`<LocSen>` defines the location of a sentence by:

$$\frac{\#S(X)}{\#S(Last\_Sentence)}$$

Here 'Last_Sentence' is the last sentence of a text. `LocSen` takes values between 0 and 1. A discourse-initial sentence takes 0, and a discourse-final sentence 1.

`<LocPar>` is defined similarly to `DistPar`. It gives information on the location of a paragraph in which a sentence X occurs.

$$\frac{\#Par(X)}{\#Last\_Paragraph}$$

'#Last_Paragraph' is the position of the last paragraph in a text, represented by its ordinal number.

`<LocWithinPar>` gives information on the location of a sentence X within a paragraph in which it appears.

$$\frac{\#S(X) - \#S(Par\_Init\_Sen)}{Length(Par(X))}$$

'Par_Init_Sen' refers to the initial sentence of a paragraph in which X occurs, 'Length(Par(X))' denotes the number of sentences that occur in that paragraph. `LocWithinPar` takes continuous values ranging from 0 to 1. A paragraph initial sentence would have 0 and a paragraph final sentence 1.

`<LenText>` the length of a text, measured in Japanese characters.

`<LenSenA>` the length of A in Japanese characters.

`<LenSenB>` the length of B in Japanese characters.

`<Sim>` gives information on the lexical similarity between A and B, based on an information-retrieval measure known as $tf \cdot idf$.[4] One important point here is that we did not use words *per se* in measuring the similarity. What we did was to break up nominals from sentences into simple characters (grapheme) and use only them to measure the similarity. We did this to deal with abbreviations and rewordings, which we found quite frequent in the corpus we used.

`<Sim2>` same as `Sim` feature, except that the similarity is measured between A and $Par(B)$, a paragraph in which B occurs. We define `Sim2` as '$SIM(A, Concat(Par(B)))$' (see footnote 4 for the definition of SIM), where '$Concat(Par(B))$' is a concatenation of sentences in $Par(B)$.

`<IsATitle>` indicates whether A is a title. We regarded a title as a special sentence that initiates a discourse.

`<Clues>` differs from features above in that it does not refer to any single feature but is a collective term for a set of clue-related features, each of which is used to indicate the presence or absence of a relevant clue in A and B. We examined N most frequent words found in a corpus and associated each with a different clue feature. We experimented with cases where N is 0, 100, 500 and 1000. A sentence can be marked for a multiple number of clue expressions at the same time. For a clue c, an associated `Clues` feature c' takes one of the four values, depending on the way c appears in A and B. $c' = 0$ if c appears in neither A or B; $c' = 1$ if c appears in both A and B; $c' = 2$ if c appears in A and not in B;

[4] For a word $j \in S_i$, its *weight* w_{ij} is defined by:

$$w_{ij} = tf_{ij} \cdot \log \frac{N}{df_j}$$

df_j is the number of sentences in the text which have an occurrence of a word j. N is the total number of sentences in the text. The $tf \cdot idf$ metric has the property of favoring high frequency words with local distribution. For a pair of sentences $X = \{x_1, \dots\}$ and $Y = \{y_1, \dots\}$, where x and y are words, we define the lexical similarity between X and Y by:

$$SIM(X,Y) = \frac{\sum_{i=1}^{t} w(x_i)w(y_i)}{\sqrt{\sum_{i=1}^{t} w(x_i)^2 \cdot \sum_{i=1}^{t} w(y_i)^2}}$$

Table 2-6 Top 20 lexical clues. *Suushi* below is a grammar term of a class of numerals. Since there are infinitely many of them, we decided not to treat them individually, but to represent them collectively with a single feature suushi.

lemma	explanation
、	comma
。	period
suushi*	numerals
wa	topic marker
suru	'do'
」	right angular parenthesis
「	left angular parenthesis
mo	topic marker
(	left parenthesis
)	right parenthesis
nado	'so forth'
―	dash
nai	negative auxiliary
aru	'exist','be'
kara	'from'
koto	nominalizer
dewa	topic marker
nen	'year'
hi	'day'
no	possessive particle

and $c' = 3$ if c appears not in A but in B. We consider clue expressions from the following grammatical classes: *nominals*, *adjectives*, *demonstratives*, *adverbs*, *sentence connectives*, *verbs*, *sentence-final particles*, *topic-marking particles*, and *punctuation marks*.[5] While we did not consider a *complex* clue expression, which can be made up of multiple elements from various grammatical classes[6], it is possible to think of a complex clue in terms of its *component clues* for which a sentence is marked.

[5]They are extracted from a corpus by a Japanese tokenizer program (Sakurai & Hisamitsu, 1997).

[6]English examples would be *for example*, *as a result*, etc., which are thought of as an indicator of a discourse relationship.

Classes For a sentence pair A and B, the class is either yes or no, corresponding to the presence or absence of a dependency link from B to A.

The features above are more or less plucked from the air. Some are motivated, and some are less so. Our strategy here, however, is to rely on the decision tree mechanism to select 'good' features and filter out features that are not relevant to the class identification.

Let us make further notes on the issue of encoding a discourse with the set of features we have described. We characterize a sentence in relation to its potential nucleus sentence, a sentence which it is likely to depend on. Thus encoding is based on a pair of sentences, rather than on a single sentence. For example, a discourse $D = \{S_1, S_2, S_3\}$ would give a set of possible dependency pairs $\mathcal{P}(D) = \{< S_2, S_1 >, < S_3, S_1 >, < S_1, S_2 >, < S_3, S_2 >, < S_1, S_3 >, < S_2, S_3 >\}$. We assume that $\mathbf{C}_F(D) = \mathbf{C}_F(\mathcal{P}(D))$. Furthermore, we could constrain $\mathcal{P}$ by focusing only on pairs of a particular type. If we are interested only in backward dependencies, then we will have $\mathcal{P}(D) = \{< S_1, S_2 >, < S_1, S_3 >, < S_2, S_3 >\}$.

In experiments to be described below, for more or less practical purposes such as lessening the burden of annotators and speeding up the annotation process, we adopted a moderate version of RST with the following constraints.

1. A text consists solely of backward dependencies.
2. Each sentence has exactly one preceding nucleus.
3. A text may have crossing dependencies.

Note that the adjacency principle is no longer upheld by the moderate version.

2.4 Evaluation

2.4.1 Test Data

To evaluate our method, we have done a set of experiments, using data from a Japanese economics daily (Nihon-Keizai-Shimbun-Sha, 1995). They consist of 645 articles of diverse text types (prose, narrative, news report, expository text, editorial, etc.), which are randomly drawn from the entire set of articles published during the year. Sentences and paragraphs contained in the data set totalled 12,770 and 5,352, respectively. We had, on the average, 984.5 characters, 19.2 sentences, and 8.2 paragraphs, for one article in the

data. Each sentence in an article was annotated with a link to its associated nucleus sentence. Annotations were given manually by the first author. Each sentence was associated with exactly one sentence.

In assigning a link tag to a sentence, we did not follow any specific discourse theories such as Rhetorical Structure Theory (Mann & Thompson, 1987b). This was because they often do not provide information on discourse relations detailed enough to serve as tagging guidelines. In the face of this, we fell back on our intuition to determine which sentence links with which. Nonetheless, we followed an informal rule, motivated by a linguistic theory of cohesion by Halliday and Hasan (1990): which suggests that we relate a sentence to one that is contextually most relevant to it, or one that has a cohesive link with it. This included not only rhetorical relationships such as 'reason', 'cause-result', 'elaboration', 'justification' or 'background' (Mann & Thompson, 1987b), but also communicative relationships such as 'question-answer' and those of the 'initiative-response' sort (Fox, 1987; Levinson, 1994; Carletta et al., 1997).

Since the amount of data available at the time of the experiments was rather moderate (645 articles), we decided to resort to a test procedure known as *cross-validation*. The following is a quote from Quinlan (1993).

> "In this procedure, the available data is divided into N blocks so as to make each block's number of cases and class distribution as uniform as possible. N different classification models are then built, in each of which one block is omitted from the training data, and the resulting model is tested on the cases in that omitted block."

The average performance over the N tests is supposed to be a good predictor of the performance of a model built from all the data. It is common to set $N = 10$.

However, we are concerned here with the accuracy of dependency parses and not with that of class decisions by decision tree models. This requires some modification to the way the validation procedure is applied to the data. What we did was to apply the procedure not on the set of cases as in C4.5, but on the set of articles. We divided the set of articles into 10 blocks in such a way that each block contains as uniform a number of sentences as possible. The procedure would make each block contain a uniform number of correct dependencies. (Recall that every sentence in an article is manually annotated with exactly

Table 2-7 Effects of lexical clues on the performance of models. N is the number of clues used. Figures in parentheses represent the ratio of improvements against a model with $N = 0$.

$N = 0$	$N = 100$	$N = 500$	$N = 1000$
0.642	0.635 (-1.100%)	0.632 (-1.580%)	0.628 (-2.220%)

one link. So the number of correct links equals that of sentences.) The number of sentences in each block ranged from 1,256 to 1,306.

The performance is rated for each article in the test set by using a metric:

$$precision = \frac{number\ of\ correct\ dependencies\ retrieved}{total\ number\ of\ dependencies\ retrieved}$$

At each validation step, we took an average performance score for articles in the test set as a precision of that step's model. Results from 10 parsing models were then averaged to give a summary figure.

2.4.2 Results and Analyses

We list major results of the experiments in Table 2-7, The results show that clues are not of much help to improve performance. Indeed we get the best result of 0.642 when $N = 0$, i.e., the model does not use clues at all. We even find that an overall performance tends to decline as models use more of the words in the corpus as clues. It is somewhat tempting to take the results as indicating that clues have bad effects on the performance (more discussion on this later). This, however, appears to run counter to what we expect from results reported in prior work on discourse(Kurohashi & Nagao, 1994; Litman & Passonneau, 1995; Grosz & Sidner, 1986; Marcu, 1997), where the notion of clues or cue phrases forms an important part of identifying a structure of discourse.[7]

Table 2-8 shows how the confidence value (CF) affects the performance of discourse models. The CF represents the extent to which a decision tree is pruned; A small CF leads to a heavy pruning of a tree. The tree pruning is a technique by which to prevent a

[7]One problem with earlier work is that evaluations are done on very small data; 9 sections from a scientific writing (approx. 300 sentences) (Kurohashi & Nagao, 1994); 15 narratives (1113 clauses) (Litman & Passonneau, 1995); 3 texts (Marcu, 1997). It is not clear how reliable estimates of performance obtained there would be.

Table 2-8 Effects of pruning on performance. CF refers to a confidence value. Small CF values cause more prunings than large values.

Clues	$CF = 5\%$	$CF = 10\%$	$CF = 25\%$	$CF = 50\%$	$CF = 75\%$	$CF = 95\%$
0	0.626	0.636	0.642	0.633	0.625	0.624
100	0.629	0.627	0.635	0.626	0.614	0.609
500	0.626	0.630	0.632	0.616	0.604	0.601
1000	0.628	0.627	0.628	0.616	0.601	0.597

decision tree from fitting training data too closely. The problem of a model fitting data too closely or overfitting usually causes an increase of errors on unseen data. Thus a heavier pruning of a tree would result in a more general tree.

While Haruno (1997) reports that a less pruning produces a better performance for Japanese sentence parsing with a decision tree, results we got in Table 2-8 show that this is not true with discourse parsing. In Haruno (1997), the performance improves by 1.8% from 82.01% ($CF = 25\%$) to 83.35% ($CF = 95\%$). 25% is the default value for CF in C4.5, which is generally known to be the best CF level in machine learning. Table 2-8 shows that this is indeed the case: we get a best performance at around $CF = 25\%$ for all the values of N.

Let us turn to effects that each feature might have on the model's performance. For each feature, we removed it from the model and trained and tested the model on the same set of data as before the removal. Results are summarized in Table 2-9. It was found that, of the features considered, `DistSen`, which encodes a distance between two sentences, contributes most to the performance; at $N = 0$, its removal caused as much as an 8.62% decline in performance. On the other hand, lexical features `Sim` and `Sim2` made little contribution to the overall performance; their removal even led to a small improvement in some cases, which seems consistent with the earlier observation that lexical features are a poor class predictor.

To further study effects of lexical clues, we have run another experiment where clues are limited to sentence connectives (as identified by a tokenizer program). Clues included any connective that has an occurrence in the corpus, which is listed in Table 2-10. Since a sentence connective is relevant to establishing inter-sentential relationships, it was ex-

pected that restricting clues to connectives would improve performance. As with earlier experiments, we have run a 10-fold cross validation experiment on the corpus, with 52 attributes for lexical clues. We found that the accuracy was 0.642. So it turned out that using connectives is no better than when we do not use clues at all.

Figure 2-8 gives a graphical summary of the significance of features in terms of the ratio of improvement after their removal (given as parenthetical figures in Table 2-9). Curiously, while the absence of the `DistSen` feature caused a largest decline, the significance of a feature tends to diminish with the growth of N. The reason, we suspect, may have to do with the susceptibility of a decision tree model to irrelevant features, particularly when their number is large. But some more work needs to be done before we can say anything about how irrelevancy affects a parser's performance.

One caveat before leaving the section; the experiments so far established neither positive or negative correlation between the use of lexical information and the performance on discourse parsing. To say anything definite would probably require experiments on a corpus much larger than is currently available. However, it would be safe to say that distance and length features are more prominent than lexical features when a corpus is relatively small.

Table 2-9 Measuring the significance of features. Figures below indicate how much the performance is affected by the removal of a feature. 'REF' refers to a model where no feature is removed. '`Clues`' indicates the number of clues used for a model. A minus sign at a feature indicates the removal of that feature from a model.

FEATURES/#CLUES	0	100	500	1000
REF.	0.642	0.635	0.632	0.628
`DistSen`$^-$	0.591 (-8.620%)	0.598 (-6.180%)	0.604 (-4.630%)	0.603 (-4.140%)
`LenText`$^-$	0.626 (-2.550%)	0.626 (-1.430%)	0.620 (-1.930%)	0.623 (-0.800%)
`LocWithinPar`$^-$	0.631 (-1.740%)	0.627 (-1.270%)	0.624 (-1.280%)	0.628 ($\pm 0.000\%$)
`Sim`$^-$	0.643 ($+0.160\%$)	0.640 ($+0.790\%$)	0.632 ($\pm 0.000\%$)	0.630 ($+0.320\%$)
`Sim2`$^-$	0.644 ($+0.320\%$)	0.647 ($+1.860\%$)	0.638 ($+0.950\%$)	0.629 ($+0.160\%$)
`LenSenA`$^-$	0.641 (-0.150%)	0.638 ($+0.480\%$)	0.632 ($\pm 0.000\%$)	0.632 ($+0.640\%$)
`LenSenB`$^-$	0.642 ($\pm 0.000\%$)	0.639 ($+0.630\%$)	0.629 (-0.470%)	0.631 ($+0.480\%$)
`LocPar`$^-$	0.640 (-0.310%)	0.637 ($+0.320\%$)	0.631 (-0.150%)	0.627 (-0.150%)
`LocSen`$^-$	0.639 (-0.460%)	0.631 (-0.630%)	0.634 ($+0.320\%$)	0.630 ($+0.320\%$)
`DistPar`$^-$	0.636 (-0.940%)	0.631 (-0.630%)	0.628 (-0.630%)	0.628 ($\pm 0.000\%$)
`IsAtitle`$^-$	0.638 (-0.620%)	0.635 ($\pm 0.000\%$)	0.631 (-0.150%)	0.628 ($\pm 0.000\%$)

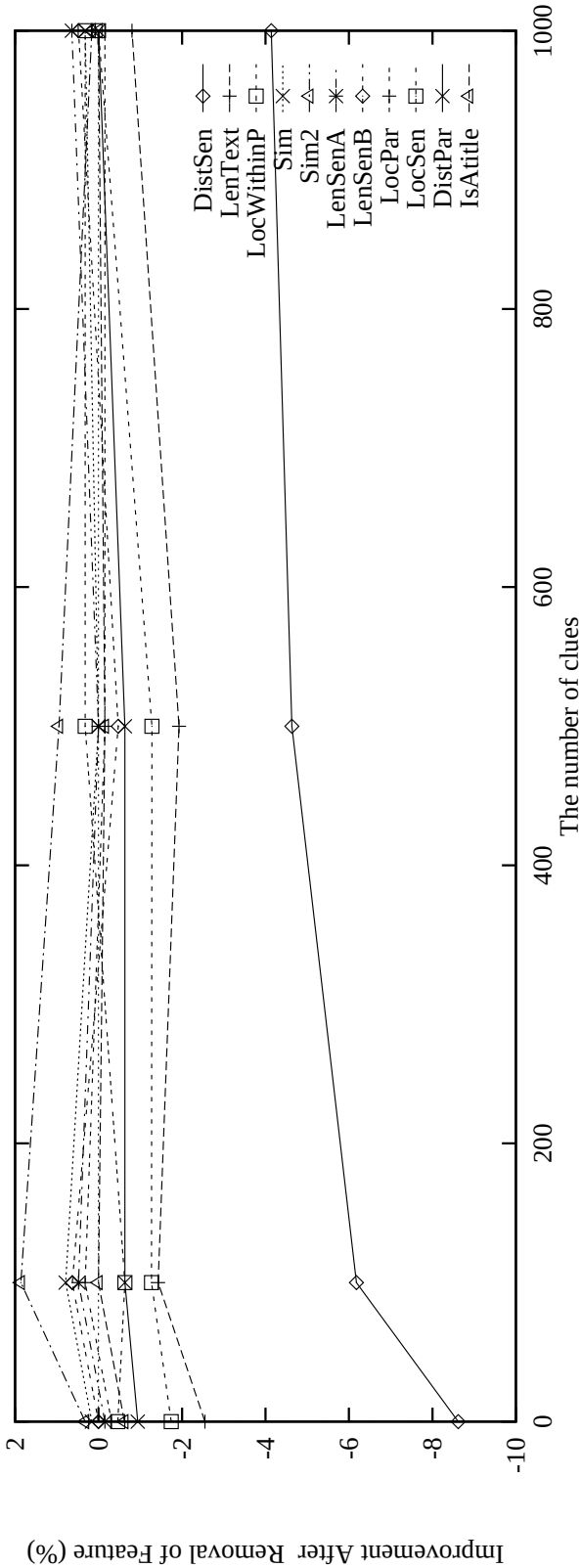

Figure 2-8 The ratio of improvement after removal of feature.

Figure 2-9 Rhetorical Parser in action

Table 2-10 Connectives found in the corpus. Underlined items (also marked with an asterisk) are those that the tokenizer program erroneously identified as a connective.

shikashi *but*, ippou *whereas*, daga *but*, soreo *(⋆)*, shikamo *moreover*, tokoroga *but*, soshite *and*, soreni *moreover*, sokode *incidentally*, soredemo *still*, sore *(⋆)*, tadashi *provided that*, soredakeni *all the more because*, tokini *by the way*, dakara *so*, demo *but*, sonoue *moreover*, sitagatte *therefore*, dewa *now*, nimokakawarazu *despite*, soredewa *well*, sorede *and then*, sorekara *after that*, towaie *nevertheless*, shitagatte *therefore*, tsuide *while*, katoitte *but*, dakarakoso *consequently*, matawa *or*, soretomo *or else*, soreto *for another thing*, nanishiro *anyhow*, omakeni *in addition*, sunawachi *in other words*, toiunowa *because*, naraba *if*, sonokawari *instead*, samunakuba *or else*, sunawachi *namely*, naishiwa *or*, sate *by the way*, toshite *(⋆)*, toiunomo *because*, sorenimokakawarazu *nonetheless*, sorenishitemo *yet*, oyobi *moreover*, tokorode *incidentally*, nazenara *because*, tosureba *if*, nanishiro *anyhow*, otto *(⋆)*, nanoni *but*

Finally, Figure 2-9 shows an experimental reading aid which makes use of the rhetorical parser. It generates a fully connected rhetorical tree and allows the user to fold and unfold a given subtree while reading the text: at the initial state, the user is presented only with with the top node (left panel). Unfolding it gives the user more details about topics discussed in the node. So if the user does not wish to bother about details, all he or she has to do is just to keep the node folded, which spares the user trouble reading parts he or she is not interested in. Thus the aid provides the user with some control over what to read in the text.

2.5 Summary

The chapter demonstrated how it is possible to build a discourse parser which performs reasonably well on diverse data. It relies crucially on (a) feature selection by the decision tree model and (b) the way a discourse is encoded. While we have found that distance and length features are more prominent than lexical features, we were not able to establish the usefulness of the latter features, which is expected from earlier work on discourse as well as on sentence parsing (Magerman, 1995; Collins, 1996).

One of the important questions that remain is whether the consistency or integrity of annotations by humans in any way affects performance of the parser trained on them. We will return to the issue in a slightly different guise in chapter 7.

Chapter 3 Towards Learning Rhetorical Relations

In this chapter and the next two, we are going to be concerned with the computational analysis of rhetorical relations and how to bring machine learning to bear on it. We learned from the previous chapter that manual annotation of rhetorical dependencies is prone to inconsistency and incoherence due to the lack and, perhaps, impossibility of objective rules to follow. So, from the results obtained, it is not possible to extrapolate general performance of the rhetorical parser on data elicited from humans. In face of this problem, we will attempt some reformulation of the problem, which involves making simplifying assumptions about rhetorical structure and introducing explicit and intuitively clear guidelines.

3.1 Rhetorical Theories and Corpus Development

Since, at the time of the research, there was no discourse corpus available for rhetorical relations, we decided to create one by collecting news articles from a Japanese financial paper, and hand-labeling each sentences with a possible discourse or rhetorical relation, drawing largely on Ichikawa's theory of discourse relations (ITDR) (Ichikawa, 1990) (Table 4-2). We chose ITDR over other more popular theories mainly because it makes a number of assumptions about discourse which make itself an easy guideline to follow when annotating the corpus. For instance, ITDR directly associates discourse relations with explicit surface cues (e.g., sentential connectives), making it possible for the coder to determine a discourse relation by figuring out a most natural cue that goes with the sentence. In contrast, rhetorical structure theory (RST)(Mann & Thompson, 1987a) , which is one of the more popular theories of discourse, assumes discourse relations to be highly abstract ones, and their relation to actual sentences is far more indirect (see Section 2.2 for a review).

Table 3-1 Taxonomy of discourse relations (Ichikawa, 1990).

The leftmost column lists major classes and the center subclasses. The rightmost lists some examples associated with each subclass. Note that EXPANDING has no examples in it because Japanese makes available no explicit cue to mark the relation.

LOGICAL	CONSEQUENTIAL	dakara *therefore*, shitagatte *thus*
	ANTITHESIS	shikashi *but*, daga *but*
SEQUENCE	ADDITIVE	soshite *and*, tsuigi-ni *next*
	CONTRAST	ippô *in contrast*, soretomo *or*
	INITIATION	tokorode *to change the subject*, sonouchi *in the meantime*
ELABORATION	APPOSITIVE	tatoeba *for example*, yôsuruni *in other words*
	COMPLEMENTARY	nazenara *because*, chinamini *incidentally*
	EXPANDING	

Another advantage of ITDR is that it attempts to characterize discourse in terms of local relations holding between adjacent sentences, which, we expected, would further lessen the difficulty of annotation and increase the reliability of the corpus[1].

In ITDR, discourse relations are organized into three major classes: the first class includes logical (or strongly semantic) relations where one sentence is a logical consequence or contradiction of another; the second class consists of sequential relations where two semantically independent sentences are juxtaposed; the third class includes elaboration-type relations where one of the sentences is semantically subordinate to the other.

When labeling sentences, coders were instructed not to identify abstract discourse relations such as LOGICAL, SEQUENCE and ELABORATION, but to choose from a list of pre-determined connective expressions. We expected that the coder would be able to identify a discourse relation with more confidence when working with explicit cues than with abstract concepts of discourse relations. Moreover, since 93% of sentences considered for labeling in the corpus did not contain any of pre-determined relation cues, the annotation task was in effect one of guessing a possible connective cue that may go with a sentence.

[1]This does not mean to say that *all* of the discourse relations are local. There could be exceptions that involve sentences separated far apart. However we did not consider non-local relations, as our preliminary study found that they are rarely agreed upon by coders.

Figure 3-1 Emacs based annotation aid

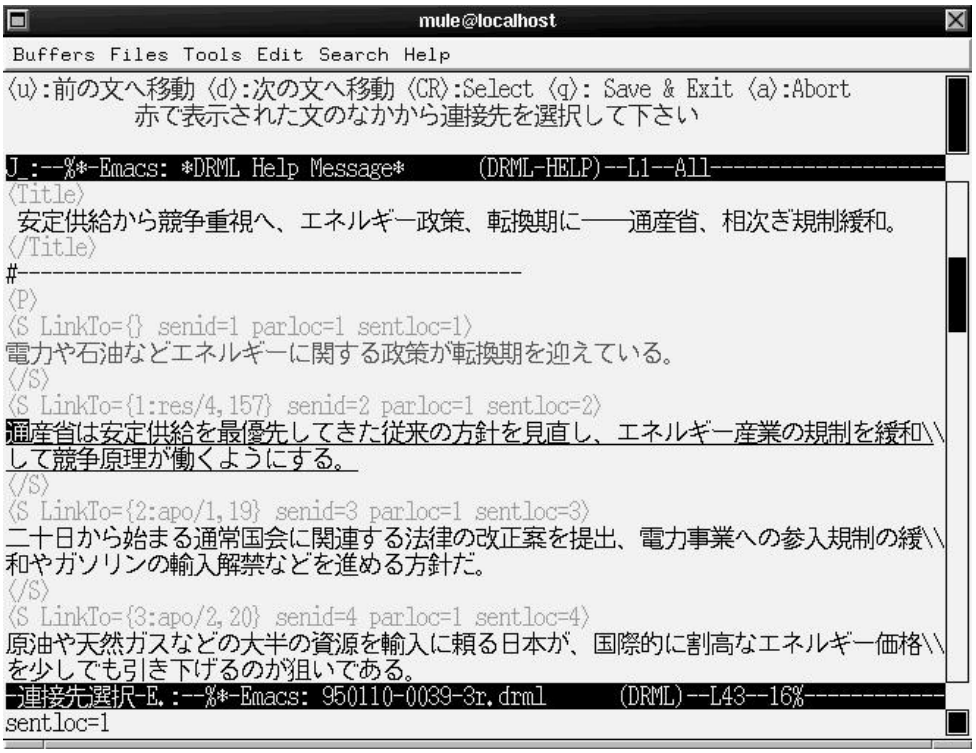

Table 3-2 Average kappa agreement among three humans on three annotation tasks under the ITDR paradigm. "3 class" denotes a three-way classification task where one is asked to first identify pairs of rhetorically associated sentences and then determine for each pair whether the association is "logical", "sequence", or "elaboration." The second column gives an agreement score as found when the three-way taxonomy of relations is reorganized into two groups, elaboration and non-elaboration. The third column shows what happens in terms of agreement when we take out relation types and consider dependencies alone.

3 class	2 class	dep. only
0.333	0.708	0.956

The advantage of using explicit cues to identify discourse relations is that even if one has little or no background in linguistics, he or she may be able to assign a discourse relation to a sentence by just asking him/herself whether the associated cue fits well with the sentence. In addition, in order to make the usage of cues clear and unambiguous, the annotation instruction carried a set of examples for each of the cues. Further, we developed an emacs-based software aid which guides the coder to work through a corpus and also is capable of prohibiting the coder from making moves inconsistent with the coding instruction (Figure 3-1).

A preliminary study, however, found that coders only modestly agree even on the three major relations (LOGICAL, SEQUENCE and ELABORATION) ($\kappa = 0.333$ for three coders, see Table 3-2). But agreement climbed to 0.708 when we used a two-way categorization scheme involving ELABORATION and NON-ELABORATION, where the the latter subsumes LOGICAL and SEQUENCE, which suggests that humans are more reliable in recognizing elaboration than any other relation, and therefore gives some ground for preferring the elaboration/non-elaboration dichotomy over the three-way taxonomy as formulated in Table 4-2. Let us note parenthetically that high agreement on dependencies in Table 3-2 is attributable to the fact that humans worked on a paragraph-by-paragraph basis.

Moreover, the elaboration/non-elaboration dichotomy can be motivated by its potential use for applications such as automatic text summarization, where the distinction plays an important role; as suggested in Mann and Thompson (1987a), one simple approach to summarization is to eliminate elaborations from the text. Moreover, it is arguable that the non-elaboration/elaboration distinction correspond to the nuclear/satellite distinction in

rhetorical structure theory (Mann & Thompson, 1987a). Intuitively, nuclear elements of the text are something that could be understood by itself and without which the text becomes incomprehensible. Satellites are elements whose interpretation cannot be determined independent of other elements such as sentences or clauses. In Table 4-1, for instance, the leading sentence is a nuclear, for it can be understood by itself, whereas the second and third sentences are satellites because they will become a non sequitur if we remove the leading paragraph. RST also claims that satellites could be dispensed with without affecting the coherence of the text. It is this claim that makes the nuclear/satellite dichotomy particularly important for text summarization, since it means that we could make a coherent summary simply by identifying satellites and then removing them. This stands in contrast to other extraction-based approaches to summarization, which usually destroy the integrity or coherence of the text. So the non-elaboration/elaboration distinction has practical as well as linguistic significance. Chapter 6 will discuss some computational consequences of this idea.

Prompted by above considerations, we decided to work with elaboration/non-elaboration scheme instead of a taxonomy involving three or more relations as originally given in Ichikawa (1990).

Chapter 4 A Minimally Supervised Learning of Rhetorical Relations

4.1 Introduction

The success of corpus-based approaches to discourse ultimately depends on whether one is able to acquire a large volume of data annotated with discourse-level information. However, to acquire merely a few hundred texts annotated for discourse information is often impossible due to the enormity of the human labor required.

This chapter explores several approaches to reducing labeled data for training a decision tree classifier in the discourse domain. While there has been some work exploring the use of machine leaning techniques for discourse and dialog (Marcu, 1997; Samuel, Carberry, & Vijay-Shanker, 1998; Choi, Cho, & Seo, 1999; Marcu & Echihabi, 2002), to our knowledge, no computational research on discourse or dialog so far has addressed the problem of minimizing data for training a learning algorithm.[1]

One obvious way to reduce training data is by avoiding data which do not contribute to improving performance. A representative approach of this sort is known as *committee-based sampling* (CBS), a voting-based sampling method initially proposed for Bayesian classifiers by Dagan and Engelson (1995), where an example is selected from the corpus, according to its utility in improving statistics. To use CBS with decision trees, however, requires some modifications to the way committee members are generated, since there is no straightforward way to randomize model parameters as in Bayesian classifiers. In the present chapter, we experimented with two methods for model generation: *bootstrapping* (Cohen, 1995) and *randomization* (Dietterich, 2000).

[1]One caveat. Later in the chapter, we will introduce a model selection scheme known as the minimum description length principle or MDL, which, though it bears the word *minimum*, has nothing to do with the issue of minimizing training data. It refers to a well known criterion for selecting among statistical models (Rissanen, 1997), and is used here to help boost performance of a decision tree algorithm. Similar principles include Bayesian Information Criterion and Akaike Information Criterion (Duda, Hart, & Stork, 2001).

Table 4-1 An excerpt from a news article.

Music giant sets sights on Internet

Britain's EMI Group and Time Warner Inc. said Monday they were merging their music businesses to create the world's top record company, worth $20 billion, with a powerful presence on the Internet.

The new music giant, Warner EMI Music, will bring together a star-studded roster of artists and catalogs in a 50-50 joint venture.

The U.S. group will have management control of the new combine, which will have a global market share of around 20 percent and annual sales of $8 billion.

Asahi Evening News, Jan. 25, 2000.

Cotraining represents another entirely different approach to reducing supervision (Collins & Singer, 1999; Blum & Mitchell, 1998; Abney, 2002).[2] Cotraining is an attempt to make most use of unlabeled data for training a classifier. It starts by training two distinct classifiers on a small set of labeled data, and bootstraps the classifiers by training them on each other's outputs from unlabeled data. Collins and Singer (1999) report some promising results for cotraining on named entity tasks.

The present approach to discourse analysis is much informed by previous work on text linguistics (Mann & Thompson, 1987a; Ichikawa, 1990). In general, text linguistics is concerned with finding regularities in relations among sentences or text-parts spanning multiple sentences. In some instances, relations between sentences are explicitly signaled by connectives such as *because*, *however*, or *therefore*. However, it is often the case that they are not marked by any linguistic means.

Consider for instance a news article excerpt with three paragraphs in Table 4-1.[3] While there is no explicit cues indicating any discourse relation, it is easy to see that the second

[2]Note that bootstrapping here refers to a particular statistical method for generating a sampling distribution without having to know anything about the population distribution (Efron & Tibshirani, 1993; Cohen, 1995), which is to be distinguished from its use in the NLP literature, where it usually means a method which incrementally learns by feeding upon itself.

[3]The sentences in Table 4-1 are presented as they appear in a news article. We call them paragraphs because each of them starts with an indentation.

Table 4-2 Taxonomy of discourse relations (Ichikawa, 1990). The leftmost column lists major discourse relations and the center subrelations. The rightmost lists some connectives associated with each subrelation. Note that EXPANSION has no explicit connectives to mark the relation.

LOGICAL	CONSEQUENTIAL	dakara *therefore*, shitagatte *thus*
	ANTITHESIS	shikashi *but*, daga *but*
SEQUENCE	ADDITIVE	soshite *and*, tsuigi-ni *next*
	CONTRAST	ippô *in contrast*, soretomo *or*
	INITIATION	tokorode *to change the subject*, sonouchi *in the meantime*
ELABORATION	APPOSITIVE	tatoeba *for example*, yôsuruni *in other words*
	COMPLEMENTARY	nazenara *because*, chinamini *incidentally*
	EXPANSION	no relevant cues

and third paragraph stand in a particular relation to the leading paragraph; that is, both of them are an elaboration on the first paragraph, giving complementary information on the event described in the first paragraph.

Notice moreover that the order in which they appear is important for interpreting the article; exchanging the first and the second paragraphs makes the article somewhat less readable, though exchanging the second and third does not affect the readability. Thus being in a particular relation has some syntactic consequences as well.[4]

In the natural language processing community, there is a popular belief that text-level dependencies such as discourse relations could provide useful information for text summarization and information extraction. Recent years have witnessed a growing interest in this area. Nomoto and Matsumoto (1998) apply a probabilistic decision tree to identifying discourse-level dependency relations in Japanese texts. Marcu (1999a) makes use of the C4.5 decision tree which refers to explicit linguistic cues, among others, to identify rhetorical relations in the text and reports that its performance is moderately comparable to humans. Choi et al. (1999) develop a statistical approach based on the maximum entropy model and puts it to the task of marking off a subdialog boundary. Marcu and Echihabi

(2002) address an approach to detecting rhetorical relations using binary classifiers based on Naive Bayes.

4.2 Building Corpus with Ichikawa's Theory of Discourse Relations

Since, at the time of the research, there was no discourse corpus available for evaluating our approach,[5] we started by collecting news articles from a Japanese financial paper, and hand-labeling each sentences with a possible discourse relation, drawing on Ichikawa's theory of discourse relations (ITDR) (Ichikawa, 1990) (Table 4-2).[6] ITDR, first and foremost, purports to be a theory of discourse structure in Japanese, and shows how linguistic connectives are usefully exploited to identify discourse level relations among sentences. Our goal was to read some RST-like features off ITDR and use them to characterize sentential relations in Japanese.

We chose ITDR over other more popular theories such as RST (Mann & Thompson, 1987a) mainly because it makes a number of assumptions about discourse which lend it to an easy-to-follow guideline for annotating the corpus. For instance, ITDR directly associates discourse relations with explicit surface cues (e.g., sentential connectives), making it possible for the coder to determine a discourse relation by figuring out a most natural cue that goes with the sentence.[7] By contrast, RST assumes discourse relations to be highly abstract ones, and their relation to actual sentences is far more indirect.[8]

Another advantage of ITDR is that it attempts to characterize discourse in terms of local relations holding between adjacent sentences, which, we expected, would make the job easier and therefore the annotation more reliable.[9]

[4]It rarely happens in English that an elaborative sentence precedes a sentence it is an elaboration of. If it does, then it would be interpreted as something else such as background, cause, sequence,etc.

[5]We started working on a corpus around 1997, which predates LDC's RST corpus (www.ldc.upenn.edu).

[6]We had two to three Japanese graduate students working on the annotation.

[7]Moser and Moore (1993) also talk about identifying discourse relations through lexical cues.

[8]Another difference is that RST allows rhetorical relations to hold among linguistic units larger than sentences, which ITDR does not, as it is more concerned with rhetorical functions of sentential connectives.

[9]Note that by defining discourse relations as those that relate two adjacent sentences, ITDR is by no means attempting to equate discourse segments with sentences. It is simply restricting its attention to those relations that involve adjacent sentences. So with ITDR, one is not able to work with segments larger than sentence, but it is a sort of compromise we make in order to have an easy-to-follow annotation scheme.

In ITDR, discourse relations are organized into three major classes: the first class includes logical (or strongly semantic) relations where one sentence is a logical consequence or contradiction of another; the second class consists of sequential relations where two semantically independent sentences are juxtaposed; the third class includes elaboration-type relations where one of the sentences is semantically subordinate to the other.[10]

Just to give a better idea of what ITDR is about, let us go through some examples. There, by 'SBJ,' 'OBJ,' and 'DAT,' we mean case marking devices signaling subject, object, and dative, respectively. 'COP' indicates copula.

(4-1) *Namikimichi* *wo* *aruite* *itta.* Suruto *hitorino* *otoka* *ga*
street lined with trees OBJ walk continued to then a man SBJ
mukowkara *chikazuite* *kita.*
from far side approach come.
'I was walking down the street lined with trees when I found a man approaching from the far side.'

Here the word suruto acts as a connective, signaling a 'consequential' relation between the two sentences. Consider the following.

(4-2) *Madono* *soto* *wa* *harusame* *da.* *Watashi* *tabako* *ni* *hi* *wo*
window outside SBJ spring rain COP I tabacco DAT fire OBJ
tsukeru.
add
'Outside the window is spring rain falling. I light my tabacco.'

In this somewhat poetic example, we have no connective linking the two sentences. Whatever relation they are in needs to be inferred (or imagined) to make sense of them.

(4-3) *8 gatsu* *15 nichi.* *Watashi* *wa* *kono-hi* *ga* *wasure-rarenai.*
August 15th I SBJ this day OBJ cannot forget.
'August 15th. The day is forever on my memory.'

[10]It is worth noting that sequence and elaboration in ITDR have close analogues in Dynamic Discourse Model (DDM) (Polanyi & Scha, 1984): what DDM calls the 'list' structure is in fact something ITDR is getting at with; the additive relation, and also elaboration in ITDR is though of as broadly corresponding to what DDM calls the *expansion* relation.

Table 4-3 Discourse Functions in ITDR.

Relation	Definition	Nuclearity
CONSEQUENTIAL	B describes a consequence of A	−
ANTITHESIS	B presents an idea that runs against A	−
ADDITIVE	B describes an event that runs in parallel or in succession to A	+
CONTRAST	B describes a idea which stands in contrast to A	−
INITIATION	B initiates a subject distinct from A	+
APPOSITIVE	B reframes A	−
COMPLEMENTARY	B provides a supplementary statement for A	−
EXPANSION	B expands/elaborates on an idea presented by A	−

Again there is no explicit connective here. One of the possible rhetorical relations could be ELABORATION, the second sentence giving a qualification to the first. Table 4-3 lists definitions of relevant discourse relations from ITDR, along with their nuclearity (Also see footnote 14 below). 'A' and 'B' are sentences in discourse, with 'B' standing in a particular rhetorical relation to 'A.' By the nuclearity, we mean the nuclearity of sentence 'B.'

When labeling sentences, coders were instructed not to label them with abstract discourse relations such as LOGICAL, SEQUENCE and ELABORATION, but to choose from a list of pre-determined connective expressions.[11] We expected that the coder would be able to identify a discourse relation with more confidence when working with explicit cues than with abstract concepts of discourse relations. Moreover, since 93% of sentences considered for labeling in the corpus did not contain any of pre-determined relation cues, the annotation task was in effect one of guessing a possible connective cue that may go with a sentence. The advantage of using explicit cues to identify discourse relations is that even if one has little or no background in linguistics, he or she may be able to assign a discourse relation to a sentence by just asking him/herself whether the associated cue fits well with the sentence. In addition, in order to make the usage of cues clear and unambiguous, the annotation instruction carried a set of examples for each of the cues. Further, we provided an emacs-based software aid which helps the coder with tagging and also is capable of prohibiting the coder from making moves inconsistent with the coding instruction.[12]

[11] See Table 4-2 for examples (given in the rightmost column). The connectives here are all among those given in Ichikawa (1990). Coders are asked to label each sentence with one of the eight relation types

A preliminary study, however, found that the agreement in kappa among three coders on the leaf-level relations, namely, CONSEQUENTIAL, ANTITHESIS, ADDITIVE, etc, was at 0.36 on 114 sentences. (We had the agreement at 0.56 on the top-level categories in the taxonomy, i.e, LOGICAL, SEQUENCE, and ELABORATION.) The agreement, however, rose to 0.63 when we switched to a two-way categorization scheme involving ELABORATION and NON-ELABORATION, where the the latter subsumes both LOGICAL and SEQUENCE.[13]

The elaboration/non-elaboration distinction, though not part of the ITDR taxonomy, is motivated by its potential for yielding applications such as automatic text summarization, where the distinction plays an important role; as suggested in Mann and Thompson (1987a), one simple approach to summarization would be to eliminate elaborations from the text (Rino & Scott, 1996; Marcu, 1998).[14]

We could put forward another argument in favor of the two-way taxonomy, from the perspective of machine learning (or engineering), which is that since the three- or eight-way taxonomy produces data with poor agreement on their relation labels, it is likely to contain more inconsistencies and irregularities than the two-way approach, and therefore could only deliver a train/test data of low quality. We do not expect an automatic learner of any sort does well if the data it is working on contain a large amount of noise. Indeed

listed at the second tier of the taxonomy in Table 4-2.

[12]Note that coders are asked to first identify a type of discourse relation such as CONSEQUENTIAL, ANTITHESIS, etc, before proceeding to choose among possible cues that belong to each relation. So we would reasonably expect them to be able to resolve ambiguity if any across relations, of cues.

[13]The experiment involved seven texts from a Japanese newswire domain (Nihon-Keizai-Shimbun-Sha, 1995).

[14] Here is why it makes sense to do so. Intuitively, nuclear elements of the text are something that could be understood by itself and without which the text becomes incomprehensible. Satellites are elements whose interpretation cannot be determined independent of other elements such as sentences or clauses. In Table 4-1, for instance, the leading sentence is a nuclear, for it can be understood by itself, whereas the second and third sentences are satellites because they will become a non sequitur if we remove the leading paragraph. RST also claims that for the most of time, satellites could be dispensed with, without affecting the coherence of the text. It is this idea that makes the nuclear/satellite dichotomy particularly important for text summarization, as it implies that we could make a coherent summary simply by removing satellites. This stands in contrast to other extraction-based approaches to summarization, which usually destroy the integrity or coherence of the text.

For instance, in a corpus we will discuss later on in the chapter (Section 4.5), we found about 39% of sentences there are elaborative and the remaining 61% are non-elaborative: elaboration forms a single largest class in terms of membership in comparison to other relation types, namely, logical and sequence, which have the membership ratio of 27% and 34%, respectively. So simply by eliminating elaborative sentences, we get, on average, as much as 39% reduction in the text length. To get further reduction, one would probably need some way of choosing among non-elaborative sentences, perhaps in the manner of Zechner (1996). Also, a cursory review of the corpus suggested that for the most of time, eliminating elaborative sentences does not seriously hurt the integrity of a text, as long as one is working in a news domain, from which our corpus is derived, though it may not hold for other genres.

we could argue that data labeled with the three- or eight-way taxonomy is unfit to serve as a train/test data for automatic learning.

All this had led us to primarily work with the two-way taxonomy instead of the three- or eight-way scheme as originally given in Ichikawa (1990): our decision here is more of an engineering decision than a linguistic one.

In what follows, we will look at how we might devise decision tree based learners for detecting elaboration and non-elaboration relations which run on the minimal amount of training data. We like to see whether it is possible to train a machine learner on reduced training data and get it working just as well as if it were trained on a full data set.

Here is how we proceed. We begin by invoking a particular sampling method known as committee based sampling (CBS) to choose among the available data those that are most useful, in a sense defined below, and then train a base learning algorithm such decision tree on only those that are chosen. (To slightly complicate the picture, we try two alternative approaches to CBS, one that uses bootstrapping and another that uses randomization.) We are done if the base learner trained on a small portion selected from the training data performs just as good as if it were trained on the whole, unabridged data. For a base learner, we adopt the C4.5 decision tree (Quinlan, 1993).

4.3 Learning with Minimum Supervision

In the following, we will look at two alternative approaches to minimizing supervised data: committee-based sampling and randomization. Either operates by calling upon an ensemble of classifiers (C4.5s) to take a vote on which sample to use for training a machine learner, but they differ in specific ways in which they come up with voter classifiers. The committee based sampling addresses a data-driven approach based on a statistical method known as bootstrapping, while randomization focuses more on manipulating the internal structure of C4.5.

In addition, we will supply C4.5 with some extension based on a model selection strategy known as the minimum description length principle (MDL) in the statistics literature. Note that even though it bears the word *minimum*, MDL has absolutely nothing to do with the issue of minimizing supervised data.

Although it happens that a voter classifier, one that is involved in selecting data, and a base classifier, that is, one that learns on the data chosen by voter classifiers, are all derived from C4.5, obviously it need not be the case.

4.3.1 Committee-based Sampling

In the committee-based sampling method (CBS) (Dagan & Engelson, 1995; Engelson & Dagan, 1996), a training example is selected from a corpus according to its usefulness; a preferred example is one whose addition to the training corpus improves the current estimate of a model parameter which is relevant to classification and also affects a large proportion of examples. CBS tries to identify such an example by randomly generating multiple models (*committee members*) based on posterior distributions of model parameters and measuring how much the member models disagree in classifying the example. The rationale for this is: disagreement among models over the class of an example would suggest that the example affects some parameters sensitive to classification, and furthermore estimates of affected parameters are far from their true values. Since models are generated randomly from posterior distributions of model parameters, their disagreement on an example's class implies a large variance in estimates of parameters, which in turn indicates that the statistics of parameters involved are insufficient and hence its inclusion in the training corpus (so as to improve the statistics of relevant parameters).

For each example it encounters, CBS goes through the following steps to decide whether to select the example for labeling.

1. Draw k models (*committee members*) randomly from the probability distribution $P(M \mid S)$ of models M given the statistics S of a training corpus.

2. Classify an input example by each of the committee members and measure how much they disagree on classification.

3. Make a biased random decision as to whether or not to select the example for labeling. This would make a highly disagreed-upon example more likely to be selected.

As an illustration of how this might work, consider a problem of tagging words with parts of speech, using a Hidden Markov Model (HMM). A (bigram) HMM tagger is typically given as:

$$T(w_1 \dots w_n) = \underset{t_1 \dots t_n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i \mid t_i) P(t_{i+1} \mid t_i)$$

where $w_1 \dots w_n$ is a sequence of input words, and $t_1 \dots t_n$ is a sequence of tags. For a sequence of input words $w_1 \dots w_n$, a sequence of corresponding tags $T(w_1 \dots w_n)$ is one that maximizes the probability of reaching t_n from t_1 via t_i $(1 < i < n)$ and generating $w_1 \dots w_n$ along with it. Probabilities $P(w_i \mid t_i)$ and $P(t_{i+1} \mid t_i)$ are called *model parameters* of an HMM tagger. In Dagan and Engelson (1995), $P(M \mid S)$ is given as the posterior multinomial distribution $P(\alpha_1 = a_1, \dots, \alpha_n = a_n \mid S)$, where α_i is a model parameter and a_i represents one of the possible values. $P(\alpha_1 = a_1, \dots, \alpha_n = a_n \mid S)$ represents the proportion of the times that each parameter α_i takes a_i, given the statistics S derived from a corpus. (Note that $\sum_i^n P(\alpha_i = a_i \mid S) = 1$.) For instance, consider a task of randomly drawing a word with replacement from a corpus consisting of 100 different words $(w_1, \dots, w_{100})$. After 10 trials, you might have outcomes like $w_1 = 3, w_2 = 1, \dots, w_{55} = 2, \dots, w_{71} = 3, \dots, w_{76} = 1, \dots, w_{100} = 0$: *i.e.*, w_1 was drawn three times, w_2 was drawn once, w_{55} was drawn twice, etc. If you try another 10 times, you might get different results. A multinomial distribution tells you how likely you get a particular sequence of word occurrences. Dagan and Engelson (1995)'s idea is to assume the distribution $P(\alpha_1 = a_1, \dots, \alpha_n = a_n \mid S)$ as a set of binomial distributions, each corresponding to one of its parameters. An arbitrary HMM model is then constructed by randomly drawing a value a_i from a binomial distribution for a parameter α_i, which is approximated by a normal distribution. Given k such models (*committee members*) from the multinomial distribution, we ask each of them to classify an input example. We decide whether to select the example for labeling based on how much the committee members disagree in classifying that example. Dagan and Engelson (1995) introduces the notion of *vote entropy* to quantify disagreements among members. Though one could use the *kappa* statistic (Siegel & Castellan, 1988) or other disagreement measures such as the α statistic (Krippendorff, 1980) instead of the vote entropy, in our implementation of CBS, we decided to use the vote entropy, for the lack of reason to choose one statistic over another. A precise formulation of the vote entropy is as follows:

$$V(e) = -\sum_{c} \frac{V(c,e)}{k} \log \frac{V(c,e)}{k}$$

Here e is an input example and c denotes a class. $V(c,e)$ is the number of votes for c. k is the number of committee members. A selection function is given in probabilistic terms,

based on $V(e)$.

$$P_{select}(e) = \frac{g}{\log k} V(e)$$

g here is called the *entropy gain* and is used to determine the number of times an example is selected; a greater g would increase the number of examples selected for tagging. Engelson and Dagan (1996) investigated several plausible approaches to the selection function but were unable to find significant differences among them.

At the beginning of the section, we mentioned some properties of 'useful' examples. A useful example is one which contributes to reducing variance in parameter values and also affects classification. By randomly generating multiple models and measuring a disagreement among them, one would be able to tell whether an example is useful in the sense above; if there were a large disagreement, then one would know that the example is relevant to classification and also is associated with parameters with a large variance and thus with insufficient statistics.

In the following section, we investigate how we might apply CBS to a decision tree classifier. As a decision tree algorithm, we used C4.5 (Release 5)(Quinlan, 1993).

4.3.2 CBS with Decision Tree Classifiers

Since it is difficult to express the model distribution of decision tree classifiers in terms of multinomial distribution, we turn to the bootstrap sampling method to obtain $P(M \mid S)$. Bootstrapping provides a way for artificially establishing a sampling distribution for a statistic, when the distribution is not known (Cohen, 1995). For us, a relevant statistic would be the posterior probability that a given decision tree may occur, given the training corpus.

Bootstrap Sampling Procedure
Repeat $i = 1 \ldots K$ times:

1. Draw a bootstrap pseudosample S_i^* of size N from S by sampling with replacement as follows:
   Repeat N times: select a member of S at random and add it to S_i^*.

2. Build a decision tree model M from S_i^*.
   Add M to $S_{\mathcal{B}}$.

S is a small set of samples drawn from the tagged corpus. Repeating the procedure 100 times would give 100 decision tree models, each corresponding to some S_i^* derived from the sample set S. Note that the bootstrap procedure allows a datum in the original sample to be selected more than once. Given a sampling distribution of decision tree models, a committee can be formed by randomly selecting k models from $S_{\mathcal{B}}$.

4.3.3 Randomization

An alternative to bootstrapping for constructing multiple classifiers (Dietterich, 2000) is *randomization*, whose idea is to randomize the decision on splits to introduce at each node. As a comparison with the bootstrapping method, we implemented the randomization in the C4.5 decision tree algorithm and performed experiments. Our implementation is based on Dietterich (2000), where a split is chosen at random from the 20 best splits (with non-negative information gain) computed at each node. For continuous attributes, each possible threshold is considered a distinct split, so that the 20 best splits may include multiple splits on the same attribute. As with the bootstrapped CBS, an example is selected by taking votes on an ensemble of randomized classifiers.

4.3.4 Features

In the following, we will discuss a set of features we adopted for characterizing a pair of sentences, a minimum text over which a discourse relation could be defined. We consider here two contiguous sentences A and B, where B is running right after A. We are interested in particular in whether B is an elaboration of A. We note that both voter and base classifiers make use of a same set of features which are described below.

`<LocSen>` defines the location of a sentence by:

$$\frac{Ord(X)}{Ord(Last\_Sentence)}$$

'$Ord(X)$' denotes an ordinal number indicating the position of a sentence X in a text, starting with 0, *i.e.*, $Ord(k\text{th\_sentence}) = k$ ($k \geq 0$). 'Last_Sentence' refers to the last sentence in a text. `LocSen` takes a continuous value between 0 and 1. A text-initial sentence takes 0, and a text-final sentence 1.

Table 4-4 Sentence-final forms `EndCue` encodes.

VALUE	EXPLANATION
1	base form of verb or verbal adjective
2	ta-form (past/perfective) of verb or verbal adjective
3	copula
4	nominal
5	parentheses
6	sentence-final particle
0	none of above

`<LocPar>` encodes the location of a paragraph in which a sentence X occurs.

$$\frac{Ord(Par(X))}{Ord(Last\_Paragraph)}$$

'$Ord(Par(X))$' denotes the ordinal number indicating the position of a paragraph containing X. '$Ord(Last\_Paragraph)$' is the position of the last paragraph in a text, represented by the ordinal number.

`<LocWithinPar>` records information on the location of a sentence X within a paragraph in which it appears.

$$\frac{Ord(X) - Ord(Par\_Init\_Sen)}{Length(Par(X))}$$

'Par_Init_Sen' refers to the initial sentence of a paragraph in which X occurs, 'Length(Par(X))' denotes the number of sentences that paragraph. `LocWithinPar` takes continuous values ranging from 0 to $(N-1)/N$, where N is the length of a paragraph: a paragraph initial sentence would get 0 and a paragraph final sentence $(N-1)/N$.

`<LenText>` is the length of a text, measured in Japanese character.

`<LenSenA>` is the length of A in Japanese character.

`<LenSenB>` is the length of B in Japanese character.

We use the following two attributes to encode information about sentence-ending cues.

`<EndCueA>` records information about a sentence-ending form of the 'A' sentence. It takes a discrete value from 0 to 6, with 0 indicating the absence in the sentence of relevant cues. (Table 4.3.4)

<EndCueB> Same as above except that this feature is concerned with a sentence-ending form of the 'B' sentence. Finally, we have two classes ELABORATION and NON-ELABORATION.

Now let us qualify some of the choices we made in Table 4.3.4. In the Japanese linguistics literature, there is an argument that some sentence endings could indicate semantic relations among sentences. Some such are inflectional categories of verbs such as PAST/NON-PAST, INTERROGATIVE, and also morphological categories like nouns and particles (e.g. question-markers). Drawing in part on Sakuma (1987), we defined six types of sentence-ending cues and marked a sentence according to whether it contains a particular type of cue. Included in the set are inflectional forms of the verb and the verbal adjective, PAST/NON-PAST, morphological categories such as COPULA, and NOUN, parentheses (quotation markers), and sentence-final particles such as -ka. All the relevant linguistic cues as well as sentences are identified using a Japanese tokenizer ChaSen (Matsumoto, Kitauchi, Yamashita, & Hirano, 1999). (Also detecting sentence boundary is not much of a problem, since ChaSen is already equipped for doing that. Besides, written Japanese sentences rarely fail to accompany a sentence boundary marker.)

Consider, for instance, the following, a short text consisting of two sentences pulled out of a Japanese newspaper article. 'GEN' stands for genitive, 'CLS' for numeral classifier (in a linguistic sense).

(4-4) *Chugaku-sotugyo no kyûshokusha-sû wa yaku 13,000-nin to*
middle school graduate GEN job-seekers SBJ some 13,000-CLS at
12-nen renzokusi-te genshôshi-ta. Kyûjin-sû wa
12 years continue decline-PERF number of available jobs SBJ
zennen yori 32.4% heri yaku 30,400-nin.
previous year from 32.4% decrease some 30,400-CLS.
'The number of middle school graduates looking for jobs is on the decline for 12 years in a row, and is estimated at around 13,000. The number of jobs available for them is around 30,400, 32.4% decline from the previous year.'

Note the second sentence ends with a noun phrase or a numeral, which is somewhat unusual for an SOV language like Japanese. What is remarkable, however, is that they abound in newspaper texts and often serve as a complementary statement to the sentence immediately preceding them. Also a cursory look at the corpus indicates that a newspaper article often opens a paragraph (a leading paragraph in particular) with a sentence with the perfective or ta ending and elaborate on that sentence with those ending with non-past verb forms like suru (*do*) as well as auxiliary da (copula).

As a final note on features, let us mention that not every feature here is linguistically motivated. This is rightly so. Since it is simply unknowable in advance which feature is effective for recognizing elaboration, it is best to try out whatever clue is available in a text, be it linguistic or otherwise. We will be least surprised if a feature of little or no linguistic import turns out to be very relevant for identifying discourse relations. The features we have here are meant to capture some notable characteristics, whether syntactic, morphological, orthographic or whatever, of two adjacent sentences in a text.

4.4 Extending C4.5 with MDL

In what follows, we will discuss extending C4.5 with what is called the minimum description length principle (MDL), a well-known criterion for choosing among statistical models. We will first provide a somewhat technical introduction to MDL and go on to talk about how it can be integrated with C4.5.

4.4.1 Minimum Description Length Principle (MDL)

Given a data sequence $(x_1, y_1), \ldots, (x_m, y_m)$ of objects and corresponding categories, the minimum description length principle (MDL) claims that a model that allows a shortest description of the sequence $y_1, \ldots, y_m$ is most likely to have given rise to the observed data. A model here refers to the probability distribution of a variable Y given X, where $X = x_i$ and $Y = y_i$. In the MDL, the length of a description of data is given as the sum of bits required to encode a model and bits required to encode the data given the model. The best hypothesis h for $y^m = y_1, \ldots, y_m$ is then expressed as follows:

$$h_{best}(y^m) = \arg\min_{M \in \mathcal{M}} L(y^m : M)$$

where $L(y^m : M) = L(y^m \mid M) + L(M)$, $\mathcal{M}$ is the set of possible models, and $L(x)$ is a description-length of x.

In the context of the decision tree learning, it is natural to think of MDL as a way to find a best pruned tree from the set of all prunings possible for a certain decision tree $\mathcal{T}$. $\mathcal{M}$ would be a set of all subtrees of $\mathcal{T}$. Moreover, MDL requires the probability distributions of categories. But given a decision (sub)tree $M \in \mathcal{M}$, probability distributions of categories can be obtained by associating each leaf in M with $P(Y \mid X)$, the probability of category Y given a set X of attribute values. Thus instead of outputting a classification for an object,

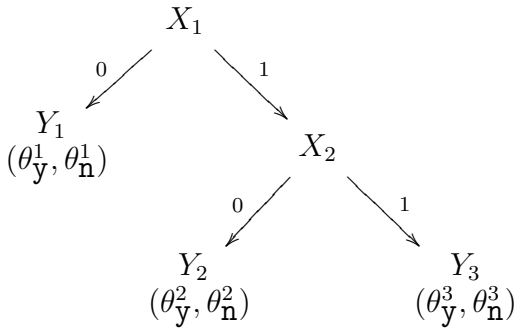

Figure 4-1 A probabilistic decision tree.

the decision tree outputs the probability that the object belongs to a particular class. Let $\theta^i = (\theta^i_1, \ldots, \theta^i_s)$, where θ^i is a list of probabilities or a *parameter vector* associated with a leaf i and θ^i_j is the probability of category j at the leaf i. Note also that $\sum^s_j \theta_j = 1$.

Consider a binary decision tree in Figure 4-1, which has X_1, X_2 for attributes (with values 0 and 1), Y_1, Y_2, Y_3 for leaves, and $\mathtt{y}$(es), $\mathtt{n}$(o) for categories. Associated with each leaf is $\theta_{\mathtt{y}}$ and $\theta_{\mathtt{n}}$, representing the probability of a respective category therein.

Since the probability of observing the data $y_1, \ldots, y_m$ given a decision tree M with parameters $\theta^1, \ldots, \theta^m$ is $P(y^m \mid \theta, M)$, the number of bits needed to encode the data is then $-\log P(y^m \mid \theta, M)$[15]. In general, the description-length of the data $y_1, \ldots, y_m$ under model M (together with bits required to encode parameters) is known to come to:

$$L(y^m \mid M) = -\log P(y^m \mid \hat{\theta}, M) + \frac{k}{2} \log m + O(1) \tag{4-5}$$

Li (1998)

Write $\hat{\theta}$ for the MLE (maximum likelihood estimate) of θ and let $\hat{\theta}^i = (\hat{\theta}^i_1, \ldots, \hat{\theta}^i_s)$. $O(1)$ is a function such that $\lim_{m\to\infty} O(1) = c$ and can safely be ignored here. k is the number of free parameters. Equation 4-5 is regarded as an approximation to the stochastic complexity as formulated in Rissanen (1997).

In addition to the length of describing data, we need to take into account the length of encoding the decision tree itself. One approach proposed in Quinlan and Rivest (1989) is to translate the tree into a string of 1's and 0's (1 for the node and 0 for the leaf) and take the description-length of the string as that of the tree. Or more simply, if we assume that

[15]Throughout the chapter, we use the base 2 logarithm unless otherwise stated.

$\mathcal{M}$ is finite and the probability distribution over it is uniform, then the length of encoding a model M is $L(M) = \log \mid M \mid$. However we take a more straightforward approach to finding the length of the decision tree as will be explained in the following section.

4.4.2 Pruning with MDL

Now finding a subtree with the minimum description length is tantamount to finding an optimal pruning for a given decision tree. Essentially we take a approach based on Yamanishi (1997) and Rissanen (1997).

Consider a decision tree $\mathcal{T}$, with a set $\mathcal{H}$ of subtrees of $\mathcal{T}$ and a set $\mathcal{A}$ of attributes. Then a best pruning for $\mathcal{T}$ given the data $y^1, \ldots, y^m$ is one whose description length equals $min_{M \in \mathcal{H}}\{L(y^m : M)\}$. Based on Equation 4-5, we define the the length of describing data at the node u by:

$$I(u) = -\sum_{j=1}^{K} F_u(j) \log \hat{P}_u(j) + \frac{k}{2} \log N$$

where N is the number of cases that reached u, $F_u(j)$ is the frequency of category j at u, K is the number of categories, $\hat{P}_u(j)$ is the maximum likelihood estimate of the category j at u, k is the number of free parameters at u, so this would be $K - 1$. The model length of u is given as follows:

$$l_u(m) = \begin{cases} -\log P_0 & \text{if } u \text{ is a leaf} \\ -\log P_1 + l_u(a) & \text{otherwise} \end{cases}$$

where $l_u(a)$ is the length of a splitting attribute a at u, which is given as $l_u(a) = \log \mid A \mid$, for a set A of possible attributes, which a is part of. If the attribute is continuous-valued, then the cost of encoding a threshold, $\log(r)$, is added to $l_u(a)$, where r is the number of possible thresholds at u. P_1 is the probability of observing a nonterminal node in $\mathcal{T}$ and P_0 that of observing a leaf in $\mathcal{T}$. Note that $P_1 + P_0 = 1$. Thus for a tree in Figure 4-1, $P_0 = \frac{3}{5}$, $P_1 = \frac{2}{5}$, $l(X_1) = l(X_2) = \log \mid \mathcal{A} \mid$.

As in Li (1998), we take a dynamic programming approach to finding a subtree with the minimal description length, which basically consists of working up the tree by eliminating daughters of u if the following condition is met:

$$-\log P_0 + I(u) \leq -\log P_1 + l(u) + \sum_{v \in D(u)} L(v),$$

where $D(u)$ denotes a set of daughter nodes of u. A precise algorithm is given in Table 4-5.

Table 4-5 A pruning algorithm based on MDL

MDL-Prune(u)
`/* input:root node of a tree */`
begin
if u is a leaf **then**
  set $L(u) = -\log P_0 + I(u)$
  return $L(u)$
else
  $L(u) = \sum_{v \in D(u)}$ MDL-Prune(v)
  where $D(u)$ is a set of daughter nodes of u.
  if $-\log P_0 + I(u) \leq -\log P_1 + l(u) + L(u)$ **then**
¿ remove every $v \in D(u)$
  endif
  return $\min\{-\log P_0 + I(u), -\log P_1 + l(u) + L(u)\}$
endif
end

4.5 Evaluation

Now let us find out whether C4.5 driven CBS is effective in reducing training data. To that end, we ran a series of experiments, using a corpus containing news articles from a Japanese economics daily (Nihon-Keizai-Shimbun-Sha, 1997). The corpus had 894 articles, randomly selected from issues that were published during the year. Each sentence in the articles was tagged with one of the discourse relations at the subrelation level (i.e.

Table 4-6 Performance of a non-sampling C4.5 with various pruning options, as determined by running 10-fold cross validation on the entire data set. The baseline here is C4.5 without pruning. 'REP' refers to C4.5 coupled with reduced error pruning, and 'MDL' C4.5 with MDL pruning. The figures denote error rates averaged over 10 runs (folds).

BASELINE	REP	MDL
40.9%	38.2%	35.4%

Table 4-7 The impact of features on performance. The follow figures show what happens to C4.5 when a particular feature is removed. NONE means running C4.5 with no feature removed. 'EndCueAB' means that both EndCueA and EndCueB are removed when running the classifier, and similarly for 'LenSenAB.' 'EndCueAB' leads to the most notable differences in performance.

feature removed	no pruning	REP
NONE	40.9%	38.2%
EndCueA	41.2%	38.6%
EndCueB	42.1%	39.8%
LenSenA	41.9%	39.0%
LenSenB	41.3%	38.2%
LenText	40.4%	36.9%
LocPar	40.9%	37.7%
LocSen	41.1%	38.3%
LocWithinPar	41.1%	38.5%
EndCueAB	42.4%	39.9%
LenSenAB	41.6%	38.0%

CONSEQUENTIAL, ANTITHESIS, etc.). However, in experiments, we translated each sub-relation into either NON-ELABORATION or ELABORATION, to give some lift to the kappa score.[16] Furthermore, we explicitly asked coders not to tag a paragraph initial sentence for a discourse relation, for we found that coders rarely agree on their classifications.[17] Paragraph-initial sentences were dropped from the evaluation corpus. This had left us with 7,851 sentences, of which 39% are labeled ELABORATION and 61% NON-ELABORATION.

Table 4-6 lists performance (in error rate) of C4.5 with and without reduced error pruning (henceforth, REP), and also when coupled with MDL pruning. Performance was measured by using 10-fold cross validation, where the corpus is divided evenly into 10 blocks of data and 9 blocks are used for training and the remaining one block is held out for testing. The figures in the table were error rates averaged over the 10 runs. MDL

[16] What we did was to put APPOSITIVE, COMPLEMENTARY, and EXPANSION under ELABORATION and everything else under NON-ELABORATION.

[17] Another reason behind this move is that we had the feeling that paragraph initial sentences participate in discourse relations not at the sentence level, but at the paragraph level: they typically summarize or represent the paragraph which they introduce, and whatever discourse relation they participate in also tends to affect the whole paragraph. Recall that ITDR is not about rhetorical relations among paragraphs, or textual units beyond sentences.

pruning produces performance about 10% smaller in error rate than the baseline, and also performs better than C4.5 with REP. Also one thing to note about the baseline is that we do not use here a frequency based baseline which works by simply picking up a most frequent relation as correct, because we are concerned with how sampling affects performance of various forms of C4.5. Besides, the distribution of classes may well vary from one sampled set to another.

Table 4-7 shows the impact of features on performance of C4.5, which is more visible when pruning is turned off. Note that the figures there indicate performance of C4.5 in the *absence* of a particular feature or a set of features. We find that some of the features such as LenSens, and EndCues have more notable effect on performance than others. Among them, EndCues seem most effective for discriminating between the two classes, that is, elaboration and non-elaboration. By contrast, neither LenText nor LocPar has apparent positive effect on performance: indeed, their removal leads to a *better* performance when coupled with REP.

We tested various sampling strategies on the corpus and measured performance by 10-fold cross validation. Note that our concern here is not so much with the question of which strategy performs better than a non-sampling C4.5 as with finding out whether they could deliver a learner that works with less data and its performance comparable to a fully trained C4.5. Thus exactly to what degree they perform better than C4.5 is not much of a concern nor issue here.

In CBS and randomization, one starts with 10% of the data randomly drawn from the training blocks and sequentially examines samples from the rest of the set until all the data are examined. A sample is selected on the basis of the vote entropy of decisions by classifiers generated. Each time a sample is selected, a decision tree is built on the sample together with the data acquired so far, and tested on the held-out data. Since a single run of the sampling procedure failed to gather much data, we repeatedly ran the procedure on the same data set until we used up almost all the data for training: each run starts with the set of samples accumulated in the previous runs, and sequentially examines what is left to be selected. The number of repetitions was empirically determined and set to ten.

Performance was measured each time that we added a set of ten samples to the training set, and figures were averaged, point by point, over 10 folds to give a summary graph for a particular sampling strategy. For CBS and randomization, we set $k = 10$ and $g = 1$, *i.e.*, ten committee members and the entropy gain of 1. However, we did not make use of what

Dagan and Engelson (1995) call the 'temperature,' as the instability of the decision tree classifier apparently led to a generation of sufficiently diverse committee members.

Figure 4-2 shows performance of random sampling (top), bootstrapped CBS (center), and randomized CBS (bottom), as they are applied to C4.5. The pruning is not used. The horizontal line (*target line*) indicates the 10-fold cross-validated error rate of C4.5 trained on the entire data set, which is 40.9% for cases in Figure 4-2. Thus if a given method breaks the target line, it means that it achieves better performance with less than the whole data. In addition, each graph contains a regression line generated by the loess smoother (MathSoft, 1998), which gives a rough idea of what is happening with a particular method. We consider here three different target lines, C4.5s with no pruning, REP, and MDL and examine which sampling approach works best under which target line.

In Figure 4-2, bootstrapped CBS starts to drop below the target line around 2,000 and consistently stay under the line. With random sampling, however, the pattern is not so evident. Though its error rate slowly drops with the samples, the performance widely zigzags along the target line even with a substantial number of data. Randomized CBS (the bottom panel) fares even worse. Its performance hovers above the target line and remains so until the end.

While Figure 4-2 does witness a general tendency for performance of a sampling based system, whether it is random sampling, bootstrapped CBS, or randomized CBS, to improve with the training data, it is apparent that the growth of the training data has non-linear effects on performance, which makes an interesting contrast with probabilistic classifiers like HMM, whose performance improves linearly as the training data grow. The reason may have to do with the instability of the decision tree classifier. A learning algorithm is said to be *instable* if small changes to the training set cause large changes in the learned classifier.

Figure 4-3 shows results for the same three methods with REP, the confidence level being set to 25%. We find that regardless of which sampling method we would use, the pruning has notable effects on performance, which improves by a few percent compared to methods in Figure 4-2, which employ no pruning. The general pattern appears similar to that in Figure 4-2, with differences between random sampling and bootstrapped CBS still visible though somewhat subdued compared to the previous figure. Bootstrapped CBS requires less data for training to break the target line than the other two. Randomized CBS

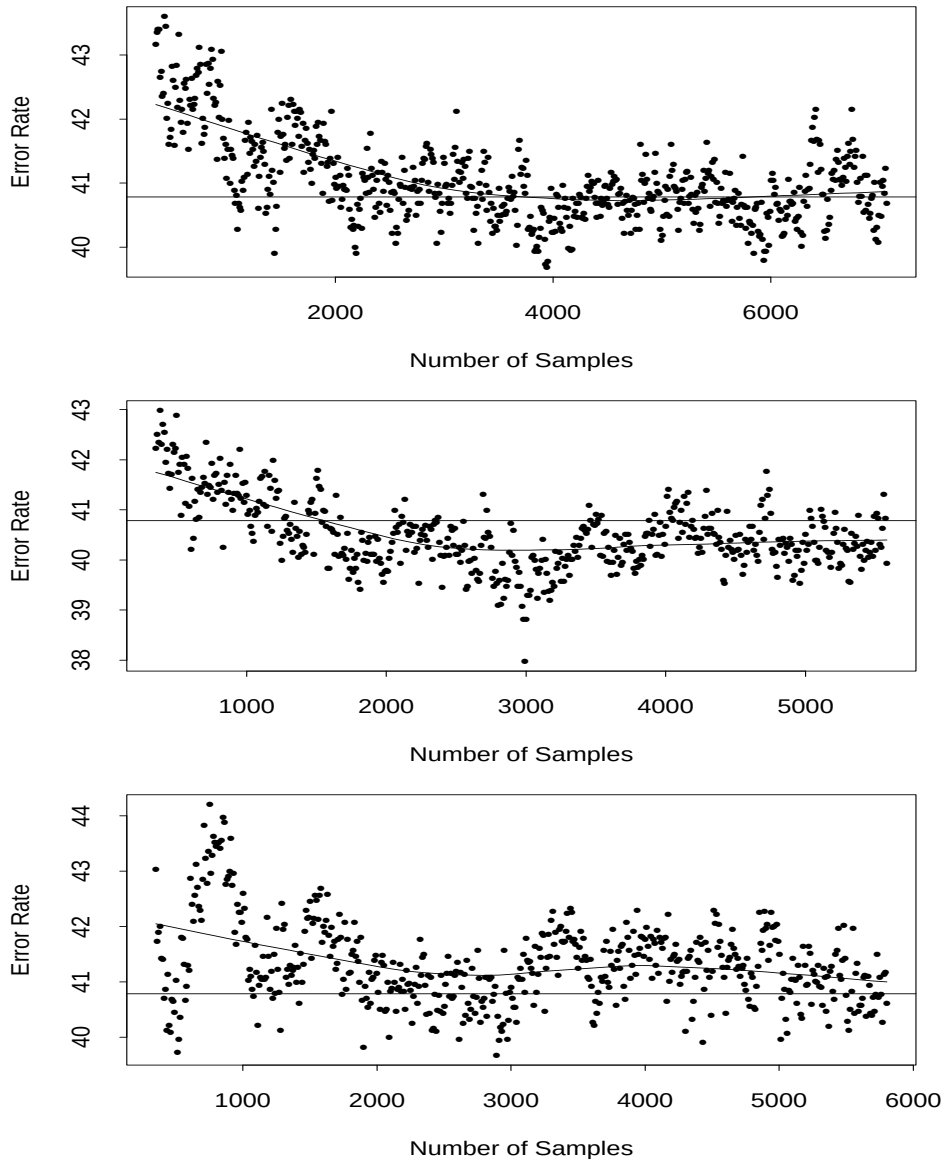

Figure 4-2 Performance of random sampling (top), bootstrapped CBS (center), and randomized CBS (bottom) with no pruning.

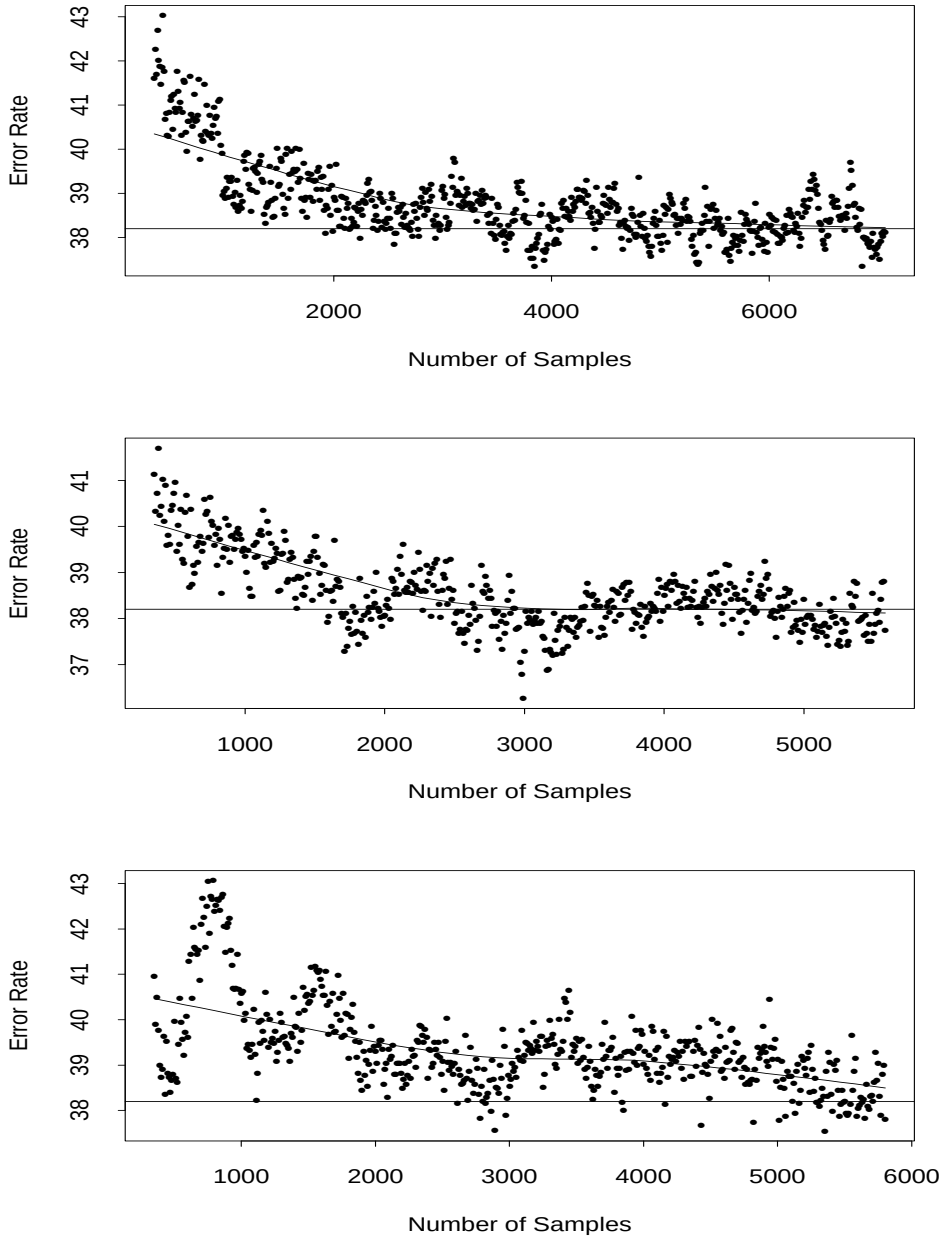

Figure 4-3 Performance of random sampling (top), bootstrapped CBS (center), and randomized CBS (bottom) with reduced error pruning.

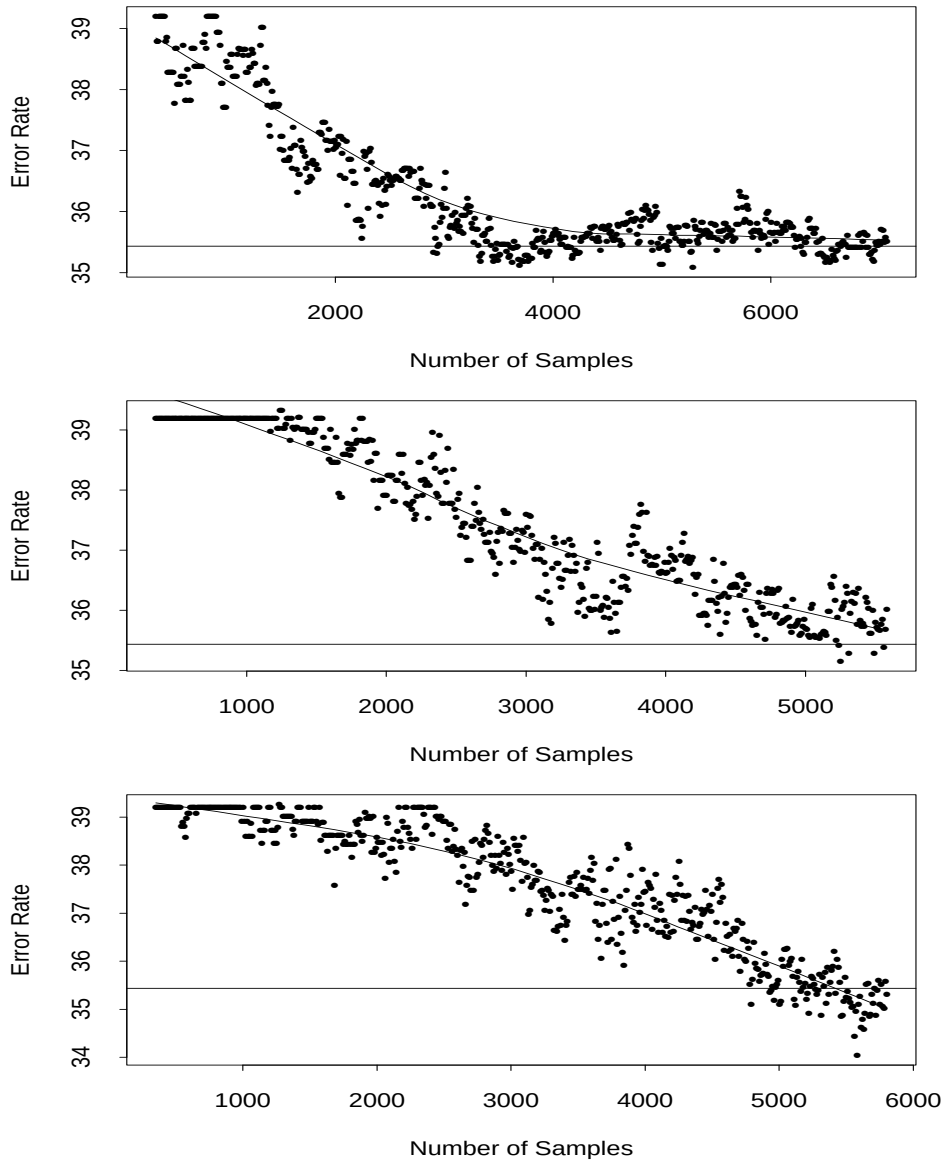

Figure 4-4 Performance of random sampling (top), bootstrapped CBS (center), and randomized CBS (bottom) with MDL pruning.

requires more than 5,000 samples to outperform the target line. Note that the target line here represents a non-sampling C4.5 with REP, whose error rate on the corpus is 38.2%.

We have a quite different picture with the MDL pruning (Figure 4-4). Random sampling converges at a much faster rate than the other two, leveling off at around 4,000 with the error rate of about 36.0%. By contrast, it takes almost the whole data (more than 5,000 samples) for bootstrapped CBS and randomized CBS to beat the target line, or a non-sampling C4.5 with MDL pruning (whose averaged performance on the corpus is at 35.4%). Figure 4-4 shows that whatever difference may exist between bootstrapping and randomization has little influence on MDL-based C4.5, which gives by far the most significant reduction in error rate, improving performance by as much as 10% compared to C4.5 with REP in Figure 4-3.

One curious result of the experiments is that randomized CBS is persistently worse than other methods. While the exact cause of the behavior is not clear, it may be due to noise in the training data, which could be substantial. Dietterich (2000). reports that in situations with substantial classification noise, randomization tends to perform worse than bagged C4.5, which is a close analogue of bootstrapped CBS. In situations with little classification noise, on the other hand, randomization could be superior to bagged C4.5.

In sum, the experiments found that bootstrapped CBS is effective with C4.5 with no pruning as well as with REP, allowing the latter to achieve the level of performance comparable to a fully-trained C4.5, with less than half as much data. With MDL based C4.5, however, bootstrapped CBS is found to be not as effective, which is significantly outperformed by random sampling. MDL based C4.5, coupled with random sampling, achieves by far the largest reduction in error rate with training data about 50% of the corpus.

Recall once again that our primary concern in the study is not with how much better sampling based approaches perform against the target lines, but how little data need to be fed into the systems before they break the target lines. The experiments demonstrate that it is indeed possible to cut back on training data for a decision tree learner without compromising its performance. Another point we learned from the experiments is that a same sampling approach may not always deliver best results: bootstrapped CBS works best for C4.5 with no pruning and with REP, and not for MDL C4.5, while random sampling works best for MDL C4.5, and not for C4.5 with no pruning or with REP. So

Table 4-8 The granularity of discourse relations and classification performance in relation to the Kappa statistic. The 3-way taxonomy includes three major relations, logical, sequence, and elaboration, and the 8-way taxonomy includes all the relations at the lowest level such as consequential, antithesis, etc.

taxonomy	no pruning	REP	κ
2-way	40.9%	38.2%	0.63
3-way	59.7%	58.2%	0.56
8-way	75.2%	74.4%	0.36

which sampling method is most effective pretty much depends on what learning algorithm is used for a task.

4.6 Summary

This chapter presented an empirical comparison of some alternative approaches to minimizing data for training decision tree classifiers towards the automatic detection of discourse relations. We studied several combinations of sampling methods and pruning algorithms. Among the sampling methods considered here are bootstrapped CBS, randomized CBS, and random sampling, and for pruning algorithms, we considered the reduced error pruning and MDL pruning. CBS is motivated by the idea that data can be somehow discriminated in terms of usefulness for improving model parameters, and one can reduce training data by dismissing those not useful. How useful a given example is depends on how much an ensemble of classifiers disagree in its classification.

The experiments show that bootstrapped CBS does have a positive effect on performance of C4.5 with no pruning as well as with REP. However the effects are all but gone on classifiers with MDL pruning. Randomized CBS did not perform well in any of the experiments conducted. The MDL pruning, when used with random sampling, gave the best performance in terms of accuracy and convergence rate. With 50% of the training data, it reached the error rate of around 36.0%, which amounts to some 10% improvement over a decision tree with REP trained on the entire data.

As for CBS, we were unable to find the effectiveness comparable to that observed for Bayesian classifiers (Dagan & Engelson, 1995). Perhaps one possible place to look at for an explanation is noise contained in the train data, which could be large.

In this connection, it is interesting to look at Table 4-8, which shows how the granularity of discourse relations and associated kappa scores relate to performance of C4.5.[18] Apparently, the larger the granularity, the higher the kappa statistic, and the better the classifier's performance, which may suggest that the kappa may indeed dictate the performance of the classifier. So somehow fixing the agreement among coders may lift performance of the classifier.

However, low kappa rates from the experiments indicate that discourse relations are not the sort of things that naturally come to mind for the most people. Therefore, a practical and perhaps most sensible way to boost the agreement is not by fixing the annotation scheme and training lay people on that, but rather hiring one coder-linguist trained in rhetorical theories and letting her work on the annotation alone.

[18]Though we did not elaborate on it in the paper, there is nothing in principle that prevents the decision tree from going beyond the simple binary classification.

Chapter 5 MDL and Boosting in Learning Rhetorical Relations

In this chapter, we will make an experimental comparison of two methods known in the literature to improve learning effectiveness; one is AdaBoost, a particular form of boosting whose goal is to construct an ensemble of diverse classifiers and combine them to produce a single high-accuracy classifier (Freund & Schapire, 1996); and the other is MDL as laid out in the previous chapter. Let us begin by introducing the reader to some basic ideas of boosting.

5.1 Boosting

We will focus on a particular boosting method called *adaptive boosting* or AdaBoost (Freund & Schapire, 1996; Dietterich, 2000), which is widely recognized as most effective for improving performance of learning algorithms. AdaBoost works by repeatedly training classifiers and then combining them into a single composite (and highly accurate) classifier. The method was tested on various kinds of tasks and learning algorithms and proved to be effective. One property of AdaBoost is that it directs classifiers generated later in the boosting process to concentrate on more difficult or exceptional cases that are misclassified by ones generated earlier, thus deriving hypotheses from very different subsets of the training data. A final composite hypothesis AdaBoost outputs takes a weighted majority over its component hypotheses to determine a classification. One obvious effect of AdaBoosting is that by combining many hypotheses, it significantly reduces the random variability (or variance) of classification, thus contributing to reduced error rate.

Fig. 5-1 describes the AdaBoost algorithm (Freund & Schapire, 1996). AdaBoost takes as input a set of example-label pairs. It begins by initializing the probability distribution of examples, assigning them a uniform probability, i.e., an equal chance of being chosen. In the next step, AdaBoost calls **WeakLearn**, some learning algorithm, (e.g. a decision tree) in a series of rounds, and trains and tests the algorithm on samples selected according to the probability distribution D_t (the distribution on the t-th round). Define the error as

Table 5-1 AdaBoost (Freund & Schapire, 1996)

Input sequence of m examples $(x_1, y_1), \ldots, (x_m, y_m)$ with labels $y_i \in Y$
weak learning algorithm **WeakLearn**
integer T specifying number of iterations

Initialize $D_1(i) = 1/m$ for all i

Do for $t = 1, 2, \ldots, T$

1. Call **WeakLearn**, with the distribution D_t.
2. Get back a hypothesis $h_t : X \rightarrow Y$.
3. Calculate the error of $h_t : \epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$.
4. If $\epsilon_t > 1/2$, then set $T = T - 1$ and abort.
5. Set $\beta_t = \epsilon_t/(1 - \epsilon_t)$.
6. Update the distribution:
   $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(i) = y_i \\ 1 & \text{otherwise} \end{cases}$
   where Z_t is a normalization constant.

Output the final hypothesis:
$h_{fin}(x) = \arg\max_{y \in Y} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t}$

---

the sum of probabilities of misclassified samples and the update weight β_t as the ratio of non-error to error. Use β_t to update the distribution D_t to D_{t+1}, which results in decreasing the probability of correctly classified examples, thus a less chance of being picked up again. Updating the distribution will force a learner on the next round to work on examples harder than ever. After the T-th round, output a final hypothesis which is an aggregation of hypotheses built so far. Note that the final hypothesis takes *weighted* votes over weak hypotheses; a vote of a hypothesis with less error is weighted heavier that that of a more error-prone hypothesis.

Table 5-2 Classification tasks.

task	relations to be identified
1	consequential, antithesis, additive, contrast, initiation, appositive, complementary, expanding
2	logical, sequence, elaboration
3	{logical, sequence}, elaboration

5.2 Experiments

We conducted experiments on a corpus of news articles randomly selected from a Japanese financial paper (Nihon-Keizai-Shimbun-Sha, 1997). The corpus contained 894 articles with the total of 13,563 sentences, each of which was hand-annotated with one of the eight discourse relations in Table 4-2, except for paragraph-initial sentences, which we had removed from the corpus, because people rarely agree on their relation types. Those that are mistakenly left untagged were also removed. This had left us with 7,851 sentences.

In experiments, we prepared three learning tasks, differing in number of classes to be identified (Table 5-2). The tasks differ only in the way discourse relations are grouped into classes. In the first task, all the eight relations at the bottom of the taxonomy are used, while the second makes use of three classes LOGICAL, SEQUENCE and ELABORATION. The third task was to distinguish between elaboration and non-elaboration and was constructed by grouping LOGICAL and SEQUENCE into one class and ELABORATION into another. In the following, we refer to tasks with the eight, three, and two classes as *eight-class*, *three-class* and *two-class problem*, respectively.

5.3 Results and Discussion

Table 5-4 shows performance of C4.5 (Quinlan, 1993), AdaBoost C4.5 and MDL C4.5 on the three classification tasks as measured by using 10-fold cross-validation. C4.5 served

Table 5-3 Results of statistical tests for differences in 10-fold performance. In the table, '2c' refers to the two-class task, '3c' refers to the three class task, '8c' refers to the eight-class task. '+' indicates significance at the 1 % confidence level, '−' represents no significance, and '†' near 10 % significance. Figures given in parentheses are p-values. The testing method is two-sided standard t test.

NULL HYPO.	2c	3c	8c
MDL = BOOST	† (0.147)	−	−
C4.5 = MDL	+	† (0.104)	+
BOOST = C4.5	+	† (0.104)	+

as a base classifier. We have made some modifications to C4.5 to build AdaBoost C4.5 and MDL C4.5. Refer to Section 4.3.4 for the explanation of features used for C4.5 here. As a boosting algorithm, we used AdaBoost.M1 (Freund & Schapire, 1996). The number of rounds was set to 50 and N examples were chosen according to each round's distribution, where N is the size of the original training set.

Figures in Table 5-4 are an error rate averaged over 10 cross-validation folds. The baseline C4.5 used the default pruning option, i.e., pessimistic pruning, with the confidence level set to 25%. It is found that MDL's performance is roughly comparable to AdaBoost, though performance of either method declines as the number of classes increases. On the two-class problem, MDL achieves 9% reduction in error rate as compared against the baseline C4.5 while AdaBoost achieves 6%. On other tasks, the difference in performance is slightly smaller; 3% on the three-class problem and 6% on the eight-class problem. As shown in Table 5-3, on all the tasks, both methods perform significantly better than C4.5 with the default pruning. No significant differences, however, are found between MDL and AdaBoost except that the two-class task had differences near the 10% confidence level. Figure 5-1 gives a breakdown of the cross-validated performance of MDL C4.5 and AdaBoost C4.5. In the panels, the error rate of AdaBoost on each of 10 folds is matched against that of MDL on the corresponding fold. We see a clear pattern in the left panel, where MDL has more wins than AdaBoost, while they seem to break even in others.

Now the right panel in Figure 5-2 shows how performance of MDL and AdaBoost on the two-class problem scales with the number of training samples. The horizontal dimension

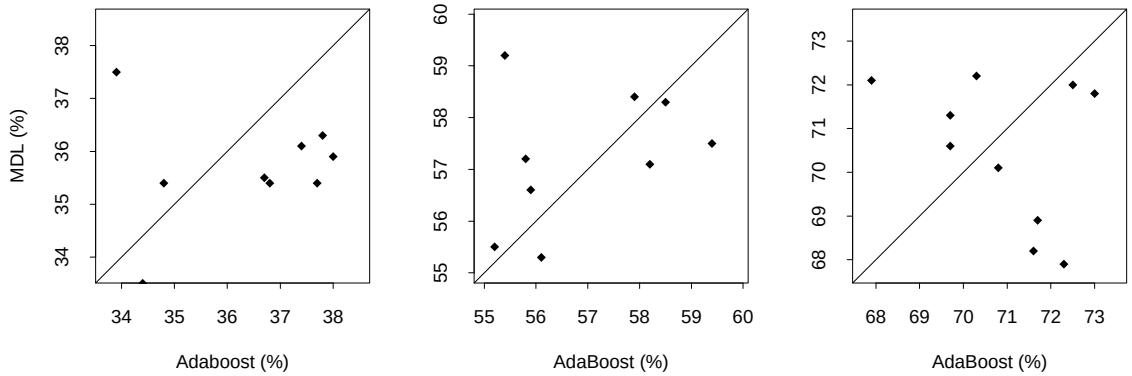

Figure 5-1 Performance (error rate) of MDL as compared against AdaBoost on the two-class problem (left), three-class problem (center), and eight-class problem (right). The vertical dimension represents the error rate of MDL, the horizontal dimension represents that of AdaBoost.

Table 5-4 10-fold cross-validated performance of C4.5, AdaBoost C4.5 and MDL C4.5 on the three tasks.

task	C4.5 (baseline)	AdaBoost	MDL
two-class	38.6%	36.4%	35.4%
three-class	58.2%	56.9%	56.9%
eight-class	74.3%	70.9%	70.5%

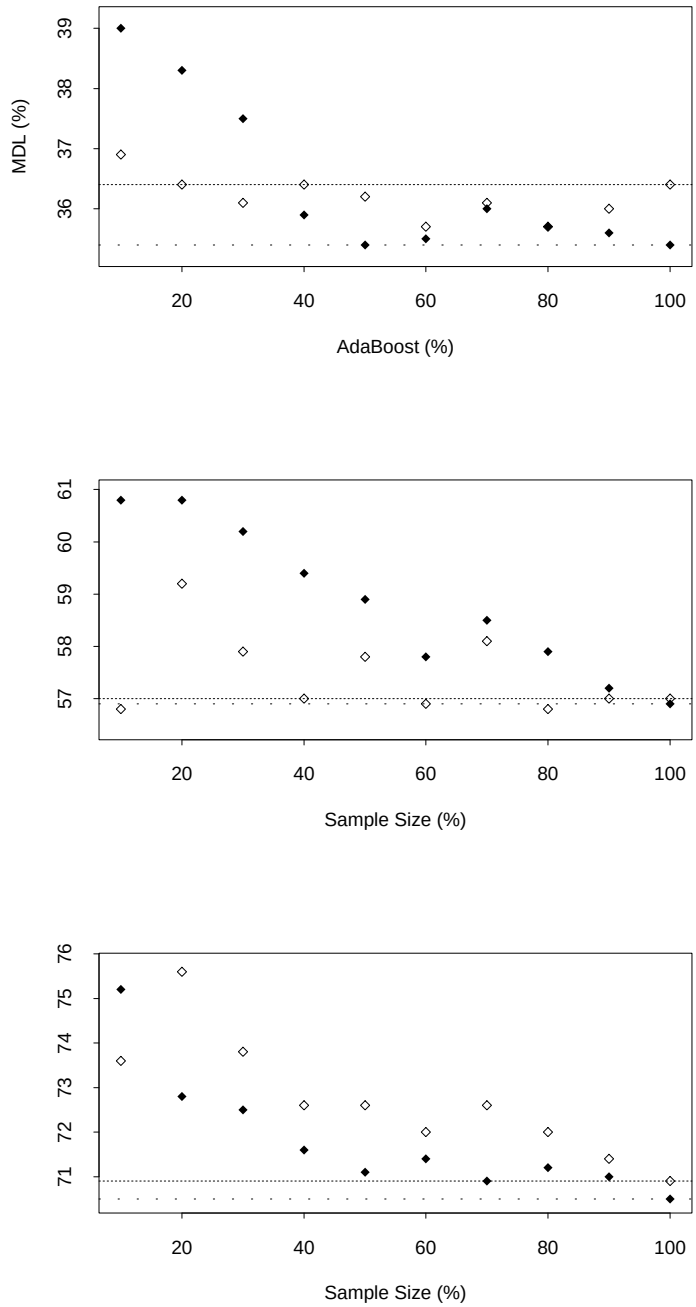

Figure 5-2 Learning curves of MDL and AdaBoost for two-class (left), three-class (center), and eight-class problem (right). Each point is the average of 10-fold cross-validation scores for a particular sample size. The filled diamond represents MDL and the white diamond AdaBoost.

Table 5-5 Difference in tree size (number of nodes) between C4.5 default pruning and MDL

Task	Unpruned	Default Pruning	MDL
two-class	2821.3	1756.7	14.0
three-class	3725.4	2939.3	14.0
eight-class	4287.2	3646.6	25.8

Table 5-6 Agreement scores for seven articles from Nikkei 95. Parenthetical figures indicate error rate of MDL C4.5 (averaged over 10 folds). (Nihon-Keizai-Shimbun-Sha, 1995)

task	3 humans + MDL C4.5	3 humans
two-class	0.613 (0.39)	0.631
three-class	0.574 (0.55)	0.558
eight-class	0.345 (0.66)	0.355

represents the ratio of samples randomly chosen from the training set to the total number of training samples, which is about 7,000, and the vertical dimension gives the error rate. The center and left panels show results for the three- and eight-class problems, respectively. The dominant pattern seems to be that MDL starts below AdaBoost, but picks up quickly as the number of samples increases, and the rate of improvement is much faster than AdaBoost.

Table 5-5 lists tree sizes (in number of nodes) under different pruning methods. MDL achieves more than 99% reduction of an decision tree with the default pruning.

Furthermore, roughly, the total time required to complete a sequence of the 10-fold cross-validation procedure for the two-class problem (including the disk I/O), was found to be 34 seconds for MDL C4.5 and 15,840 seconds (4 hrs. and 24 min.) for AdaBoost C4.5 on a Pentium III 550 MHz machine with 1GB RAM. The three-class task took 46 seconds for MDL and 21,000 seconds (5 hrs. and 50 min.) for AdaBoost, and the eight-class task took 63 seconds for MDL and 17,640 seconds (4 hrs. and 56 min.) for AdaBoost. Thus MDL runs significantly faster than AdaBoost.

Although the error rate is generally high for all the tasks, it appears that humans are not much better off when put to the same classification tasks. Table 5-6 gives two set of agreement scores, one among humans alone and other among humans and a machine, as measured by the *kappa statistic* (Siegel & Castellan, 1988), whose value ranges from 0 (no agreement) to 1 (perfect agreement). We randomly chose seven articles from a corpus of news paper articles in Nikkei 95 (Nihon-Keizai-Shimbun-Sha, 1995) and asked three (post graduate level) human subjects to annotate the articles with the set of eight labels from ITDR. The two- and three-class tasks had original hand-coded labels collapsed into two (*elaboration*, *sequence*), and into three (*logical*, *sequence*, *elaboration*), respectively. For a machine coder, we used an MDL C4.5 classifier trained on 477 articles (5221 examples) from the same corpus, which had the 10-fold cross-validated error rate at 0.39.

As Table 5-6 shows, even with this much error, in all the tasks, the machine's agreement with humans is comparable to that among humans alone, which suggests that in practice the machine could be used in place of humans. The results also suggest a possible cause of generally high error rates. Though there is no universally accepted rules for rating kappa scores, the data with the kappa score of less than 0.67 is believed to be marginally reliable, meaning a considerable lack of consistency in producing data. Thus the poor performance of the algorithm may have been caused not so much by having wrong or non-adequate features for representation but by using data containing a large amount of inconsistencies in classification.

5.4 Summary

We have made an experimental comparison of MDL and AdaBoost, two methods known to improve learning algorithms. We have found that as far as the present domain is concerned, MDL performs as well as AdaBoost and tends to outperform the latter in some cases when applied to the C4.5 algorithm. We suspect that relatively poor performance of either method may be attributable to a substantial classification noise that the training set may contain. Experiments also found that MDL does not fare well when the training data set is small. AdaBoost, however, does not seem to be affected as much by the paucity of the training data.

One of the topics which we did not address in the present chapter and are interested in pursuing in the future is use of MDL as a statistical measure of information. It is observed

in the literature (Li, 1998) that MDL can be regarded as an improved notion of entropy and thus could be put to use in place of information gain when selecting an attribute.

From a standpoint of computational linguistics, it is interesting to note that machine learning techniques such as ones investigated here can be used for exploratory purposes. In discourse studies, where there is no systematic method available to choose one classification scheme over another other than relying on one's intuition, machine learning furnishes us with powerful tools by which to discriminate among possible classification schemes otherwise hard to distinguish.

Chapter 6 Unsupervised Approach to Text Summarization

It has been argued in the literature that one of the merits of discovering rhetorical structure is its potential use for text summarization. The previous chapter demonstrated that the supervised method approaches human performance in identifying rhetorical relations. However, very low agreement for human judgments could put into jeopardy the whole enterprise of the systematic discovery and exploitation of rhetorical structure for summarization. In light of this, it makes sense and an interesting exercise to look for and pursue another possible avenue for summarization research.

6.1 Introduction

Supervised approaches to summarization (Chuang & Yang, 2000; Kupiec et al., 1995; Berger & Mittal, 2000; Marcu, 1999c) typically make use of human-made summaries or extracts to find features or parameters of summarization algorithms, while unsupervised approaches (Zechner, 1996; Carbonell & Goldstein, 1998; Luhn, 1958; Gong & Liu, 2001) determine relevant parameters without regard to human-made summaries. One of the problems with the former approach has to do with its underlying assumption that human-made summaries are reliable enough to be used as "gold standards" for automatic summarization. Recently, research on human summarization has witnessed some conflicting results about the validity of the assumption.

Nomoto and Matsumoto (1997), for instance, asked a large group of university students to identify 10% sentences in a text which they believe to be most important, which was drawn from one of various domains in a news paper corpus. They reported a rather modest result of around 25% (kappa) agreement among students on their choices. Also Salton et al. (1999) reports low inter-subject agreement on paragraph extracts from encyclopedias. On the other hand, there have been some reports to the contrary. Marcu (1997) found 71% (percent) agreement among judges on sentence extracts from expository articles; Jing et al. (1998) found quite high percent agreement (96%) for extractions from TREC articles.

It is not known at present what factors are involved in influencing the reliability of summarization and therefore we do not know whether there is any principled way of eliciting reliable summaries from humans.

Another problem associated with the approach concerns the portability of a summarization system: deploying the system in a new domain usually requires one to collect a large amount of data, which need to be manually annotated, and then train the system. Besides being costly, the annotation work is known to be quite labor-intensive, which prompted, for example, Marcu (1999c) to address the problem of automating the construction of summarization corpora.

The first part of the chapter describes the approach to text summarization which strives to overcome the issues of portability and the quality of human made summaries mentioned above. We begin by arguing for what we call the *information-centric* approach for summary evaluation. We will then explain mechanisms that drive the summarizer, and evaluate our approach using a data set known as BMIR-J2, under the information-centric paradigm.

6.2 Information-Centric Approach to Evaluation

Like previous extract-based approaches (Luhn, 1958; Edmundson, 1969; Kupiec et al., 1995), we define a summary as a set of sentences extracted verbatim from a text, which cover major substance of that text.

However, the present approach significantly differs from previous work in taking an *information-centric* approach to evaluation. We evaluate summaries, not in terms of how well they match human-made extracts (Kupiec et al., 1995; Edmundson, 1969), nor in terms of how much time it takes for humans to make relevance judgments on them (Mani et al., 1998), but in terms of how well they represent source documents in usual IR tasks such as document retrieval and text categorization. We are interested in asking whether it is possible to replace documents altogether by corresponding summaries and do as well with them in IR tasks. Thus an ideal summary would be one whose rank order in retrieved documents is same as that of its source document, or whose assigned category is same as that of its source document. This notion of summary as a perfect surrogate of its

source, while left unexplored in research on summarization, permits a simple and objective characterization of how well summaries represent the original documents.[1]

There have been arguments against extractive summarization in the computational linguistics literature, because extracts generally lack fluency or cohesion. However, Morris et al. (1999) found in a reading comprehension study that humans were able to perform as well reading 20-30% extracts as the original full texts and expert-created abstracts. Humans were able to capture enough of the information from extracts so that they could perform as if they had read their full-length versions.

6.3 The Diversity-Based Summarization

If we agree that the problem of summarization is one of finding a subset of sentences in text which in some way represents its source text, then a natural question to ask is, 'what are the properties of text that should be represented or retained in a summary?' Katz's work Katz (1996) is enlightening in this regard. In his work on language model (Katz, 1996), he made an important observation that the numbers of occurrences of content words in a document do not depend on the document's length, that is, the frequencies per document of individual content words do not grow proportionally with the length of a document. Where is the missing mass that accounts for the discrepancy between the document length and frequencies of content words? He resolves the apparent puzzle by showing that it is the number of *different* content words in text that increases with the document length.

Katz's observation illuminates two important properties of text: *redundancy* and *diversity*. The former relates to how repetitive concepts (or content words) are, the latter relates to how many different concepts there are in the text. While much of the prior work on summarization capitalize on redundancy to find important concepts in the text or its relevance to the query, few of them take an issue with the problem of diversity. One exception is Carbonell and Goldstein (1998), who added the diversity component to a criterion for sentence selection, which they call maximal marginal relevance or MMR. MMR selects a sentence in such a way that it is both relevant to the query and has the least similarity to sentences selected previously. Mani et al. (1998) report that MMR-based summarization

[1]One might consider the information-centric evaluation here an extreme form of extrinsic evaluation (Sparck Jones & Gallier, 1995; Mani et al., 1998).

ranks among the best in the 1998 SUMMAC conference. Radev et al. (2000) also makes use of some dissimilarity measure for ranking sentences for multi-document summarization. Gong and Liu (2001) presents yet another attempt to use an MMR-like feature in summarization.

6.3.1 The Method

The above discussion motivates a summarization strategy which takes seriously diversity as well as redundancy of concepts in text. We will show how to construct a *generic* single-document summarizer along this direction. Roughly, the summarizer consists of the following two operations:

Find-Diversity	Find diverse topic clusters in text.
Reduce-Redundancy	From each topic cluster, identify the most important sentence and take that sentence as a representative of the area.

Finally the summarizer outputs a set of sentences identified by Reduce-Redundancy as a summary for the text. (The summarizer could invoke some post-summarization procedures such as putting sentences in textual order.) The term "topic cluster" here is to be understood as a set of sentences which are mutually similar by some criterion but may not be necessarily contiguous. Let us look at each of the operations in details.

6.3.1.1 Find-Diversity.

Find-Diversity is built upon the K-means clustering algorithm extended with MDL (Rissanen, 1997; Li, 1998). The algorithm presented here is an MDL-version of X-means (Pelleg & Moore, 2000). X-means itself is an extension of the popular K-means clustering algorithm with an added functionality of estimating K, the number of clusters which otherwise needs to be supplied by the user. We call our adaptation of X-means 'X^M means.'

For the remainder of the chapter, borrowing in part notations from Pelleg and Moore (2000), we denote by μ_j the coordinates of the centroid with the index j, and by $\mathbf{x_i}$ the coordinates of the i-th data point. (i) represents the index of the centroid closest to the data point i. $\mu_{(j)}$, for example, denotes the centroid associated with the data point j. c_i denotes a cluster with the index i.

K-means is a hard clustering algorithm that produces a clustering of input data points into K disjoint subsets. It dynamically redefines a clustering by relocating each centroid to the center of mass of points associated with it and re-associating the centroid with points closest to it.

K-means starts with some randomly chosen initial points. As noted in Bradley and Fayyad (1998), Pelleg and Moore (2000), a bad choice of initial centers can have adverse effects on performance in clustering. In experiments described later in the chapter, following an advice by Bradley and Fayyad (1998), we repeatedly ran K-means with random initial points and selected a best clustering solution from solutions generated, on the basis of *distortion*, a measure for the tightness of a cluster. A best solution is one that minimizes distortion.

We define distortion as the averaged sum of squares of Euclidean distances between objects of a cluster and its centroid. Thus for some clustering solution $S = \{c_1, \ldots, c_k\}$, its distortion is given by:

$$D(S) = \sum_{i}^{k} V(c_i),$$

where

$$V(c_i) = \frac{1}{\mid c_i \mid} \sum_{j} (\mathbf{x}_j - \mu_{(i)})^2.$$

Here c_i denotes a cluster, $\mathbf{x}_j$ is a multidimensional point in c_i, $\mu_{(i)}$ represents the centroid of c_i, and $\mid \cdot \mid$ is the cardinality function.

One problem with K-means is that the user has to supply the number of clusters, and it is known that it is prone to searching local minima (Pelleg & Moore, 2000). X-means overcomes these problems by globally searching the space of centroid locations to find the best way of partitioning the input data. X-means resorts to a model selection criterion known as the Bayesian Information Criterion (BIC) to decide whether to split a cluster. The splitting happens when the information gain from splitting a cluster as measured by BIC is greater than the gain for keeping that cluster as it is.

Let us graphically illustrate this situation. Figure 6-1 shows a K-means solution with four centroids (large dots), which cover four distinct regions of the data space. The split operation examines each of the four clusters, breaks each of them in two, running the regular K-means on each local cluster region with $K = 2$, and decides whether the splitting is worthwhile in terms of BIC. As mentioned above, each call to K-means involves repeated runs of itself with randomly chosen initial centers and selecting the best clustering solution

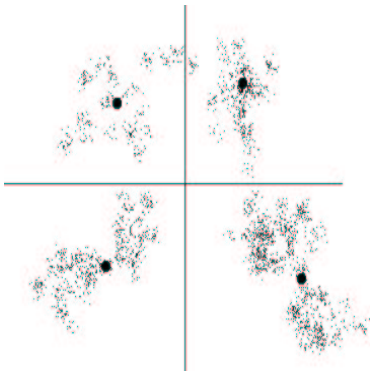

Figure 6-1 The initial state with four regions.

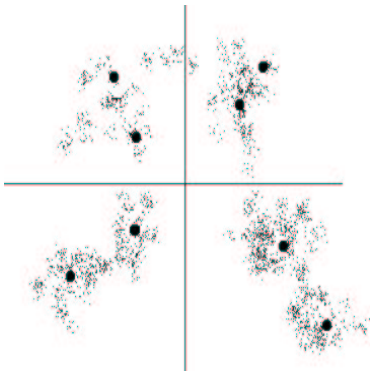

Figure 6-2 Each local cluster splits into two sub-regions.

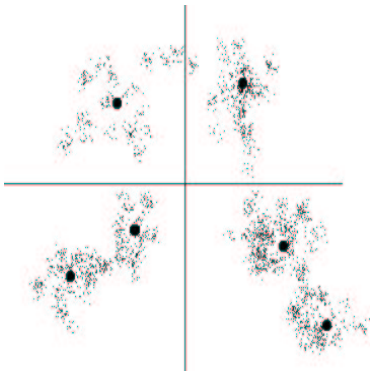

Figure 6-3 BIC determines that some of sub-regions are not worth keeping.

Table 6-1 The X^M-means Algorithm. c_0 here stands for the entire data space, L for the description length. c_i^j indicates a cluster originating from a cluster indexed with j. 2-means indicates K-means with $K = 2$.

X^M-means(c_0,$K_{\max}$)
begin
$C = \phi$
$(c_1^0, c_2^0) = 2\text{-}means(c_0)$
$C = C \cup \{c_1^0, c_2^0\}$
$k = 2$
while $k < K_{\max}$ and there is no convergence
    begin
    $S = \{c : c \in C, L(2\text{-}means(c)) < L(c)\}$
    if S is not empty **then**
      c_{best}=arg $\min_{c \in S}$ $L(2\text{-}means(c))$
      $(c_1^k, c_2^k) = 2\text{-}means(c_{best})$
      $C = C \backslash \{c_{best}\} \cup \{c_1^k, c_2^k\}$
      $k = k + 1$
    endif
    end
end

---

from those runs. (In experiments described later, we performed 200 runs of K-means and selected from those runs a solution with the least distortion.) Figure 6-2 shows a state where each local cluster splits into two sub-regions by K-means. In Figure 6-3, BIC determines that the upper left and upper right regions are not worth breaking up, leavings us with six clusters.

Our modification to X-means consists in replacing BIC by Minimum Description Length Principle or MDL, a well-studied general criterion for model selection.

In general, given the data $x_1, \ldots, x_m$, MDL contends that a model that allows a shortest description for the data is most likely to have given rise to them. (A model here is thought of as the probability distribution of a variable X, where $X = \mathbf{x}_i$.) In MDL, the length

of a description of data is given as the sum of bits required to encode a model and bits required to encode the data given the model. The best hypothesis, i.e. a model, h for $x^m = x_1, \ldots, x_m$ is then expressed as follows:

$$h_{best}(x^m) = \arg\min_{M \in \mathbf{M}} L(x^m : M),$$

where $L(y^m : M) = L(x^m \mid M) + L(M)$, $\mathbf{M}$ is the set of possible models, and $L(x)$ is a description-length of x. $L(x^m \mid M)$ denotes the description length of data, given the model M, which is the sum of the maximum log-likelihood estimate of $P(x^m \mid M)$ and the coding length of parameters involved.

Let us assume identical hyper-spherical Gaussian distributions for input data. Also let each data point represent a multi-dimensional encoding of the sentence in text, i.e. a vector of weights of index terms in the sentence. Then the probability that a given multidimensional data point $\mathbf{x}_i$ belongs to a cluster $c_{(i)}$ can be defined as the product of the probability of observing $c_{(i)}$ and the multivariate normal density function of $\mathbf{x}_i$, with the covariance matrix $\Sigma = \sigma^2 I$.[2] Therefore we have

$$\hat{P}(\mathbf{x}_i) = \frac{R_{(i)}}{R} \cdot \frac{1}{\sqrt{2\pi}\hat{\sigma}^U} \exp\left(-\frac{1}{2\hat{\sigma}^2}\|\mathbf{x}_i - \mu_{(i)}\|^2\right),$$

where $R_{(i)} = |c_{(i)}|$, $\|\cdot\|$ is the Euclidean norm, R is the total number of input data points, U is the number of dimensions, and $\hat{\sigma}^2$ is the maximum likelihood estimate of the variance such that:

$$\hat{\sigma}^2 = \frac{1}{R-K}\sum_i \|\mathbf{x}_i - \mu_{(i)}\|^2.$$

Therefore the maximum log-likelihood of the cluster c_j is:

$$\begin{aligned}\hat{l}(c_j) &= \log \textstyle\prod_{i \in c_j} \hat{P}(\mathbf{x}_i) \\ &= \sum_{i \in c_j}\left(\log\frac{1}{\sqrt{2\pi}\hat{\sigma}^U} - \frac{1}{2\hat{\sigma}^2}\|\mathbf{x}_i - \mu_{(i)}\|^2 + \log\frac{R_{(i)}}{R}\right),\end{aligned}$$

which is equivalent to:

$$\begin{aligned}&-\frac{R_{(i)}}{2}\log(2\pi) - \frac{R_{(i)} \cdot U}{2}\log(\hat{\sigma}^2) - \frac{R_{(i)} - K}{2} \\ &+ R_i \log R_i - R_i \log R.\end{aligned}$$

[2]Thus we assume that the features are statistically independent and have the same variance σ^2. See, for instance, Duda et al. (2001), for details.

Now define the probability of observing the model M by:

$$P(M) = \frac{1}{|\mathbf{M}|}.$$

Then the description-length of the model M is: $L(M) = -\log|\mathbf{M}|$. We note that $|M| = \frac{(K^N - K)}{K!} + 1$, where K represents $K_{\max}$ and N denotes the number of points one wishes to cluster. $|M|$ corresponds to the number of clusterings of N points into up to $K_{\max}$ clusters. But we can drop it from the MDL formula since it is invariant for any choice of M. This brings us to the final form of MDL:

$$L(D : M) = -\sum_{c_i}^{K} \hat{l}(c_i) + \frac{k}{2} \log R$$

where M is a model, D is a set of data points in the input space, c_i is a region or cluster as demarcated by M, R is the size of the input data, and k is the number of free parameters in M. The number of parameters here is simply the sum of $K - 1$ cluster probabilities, $U \cdot K$ centroid coordinates, and one variance estimate.

Table 6-1 shows an algorithm for X^M means. It takes as input the entire data region (represented by a single cluster c_0) and the maximum number K_{max} of clusters one likes to have. It begins then by running 2-means (i.e. K-means with $K = 2$) on the entire region, giving birth to two subregions (clusters) c_0^1 and c_0^2. Add those to C. Set k, the number of clusters in C, to 2. If $k < K_{max}$ and there is no convergence, then run 2-means on each local region found in C and test if the splitting produces a pair of child regions whose MDL is smaller than that of the parent. If that is the case, store that information in S. Find a region in S whose splitting results in the smallest MDL. Replace the region with those of children. Increment k by 1. Again test if the stopping conditions are met. If not, repeat the whole process. The idea here is that regions which are not represented well by current centroids will receive more attention with an increased number of centroids in them. The algorithm searches the entire data space for the best region to split.

6.3.1.2 Reduce-Redundancy.

For Reduce-Redundancy, we will use a simple sentence weighting model by Zechner (1996) (call it the *Z-model*), where one takes the weight of a given sentence as the sum of $tf \cdot idf$ values of index terms in that sentence. Let the weight W of sentence s be given by:

$$W(s) = \sum_{x \in s} \underbrace{(1 + \log(tf(x))) \cdot \log(N/df(x))}_{\text{tf.idf}} \tag{6-1}$$

Table 6-2 The classification of queries in BMIR-J2. Figures under the PRIMARY (SECONDARY) heading indicates the number of queries that fall under a given type.

TYPE	EXPLANATION	PRIMARY	SECONDARY
A	morphological analysis, the use of thesaurus	14	0
B	the analysis of collocation involving numbers, e.g. inequality, range	3	1
C	syntactic analysis	10	1
D	semantic, discourse analysis	9	2
E	world/common sense knowledge	4	3
F	semantic, world/common sense knowledge, reasoning	10	3

where x denotes a index term, $tf(x)$ is the frequency of term x in a document, $idf(x)$ is the inverse document frequency of x. In the Z-model, sentence selection proceeds by: determining the weights of sentences in the text; sorting them in a decreasing order; and finally selecting top sentences. Our implementation of Z-model further normalized sentence weight for length.

Reduce-Redundancy applies the Z-model to each one of the clusters identified by Find-Diversity and finds out a sentence with the best $W(s)$ score, which it takes as representative of the cluster. Note that this strategy of picking up best scoring sentences is designed to minimize the loss of the resulting summary's relevance to a potential query. This is in contrast to an approach by Radev et al. (2000), where the centroid of a cluster is selected instead.

6.4 Test Data and Evaluation Procedure

6.4.1 BMIR-J2

BMIR-J2 (Benchmark for Japanese IR system version 2) —developed jointly by a Japanese academic society and a government-funded research consortium —represents a test collection of 5,080 news articles in Japanese, all of them published in 1994 (Nichi-Gai Associates, 1995). Articles were collected from such diverse domains as economy, engineering, and industrial technology, but they all came from a single news paper source. The collection comes with a set of 60 queries and the associated list of answers indicating

which article is relevant to which query to what degree. The degree of relevance falls into one of three categories: *A*, *B*, and *C*. *A* indicates a perfect match to the query, *B* some relevance and *C* no relevance. All of the articles were manually labeled for relevance, and labelings were reviewed by the project committee, comprising researchers from industry and academia. The collection also features a classification of queries based on sorts of language technologies potentially required to process them (Table 6-2). As can be seen from the table, the classification gives us a general idea of how difficult the task of retrieval using a given query is. For instance, to properly identify documents relevant to a query of type *A* requires the morphological analysis of the query, which involves tokenization, lemmatization and assignments of grammatical categories, and also the possible use of a thesaurus, and to deal with a query of type *B*, one needs some way of analyzing collocation involving numbers, which is more difficult than an *A*-type query. To find documents relevant to a query of type *F*, it is claimed, one has to make reference to common sense or knowledge about the world and also some reasoning, which is the hardest of all. *C*, *D*, and *E*-type queries come in between.

Moreover, a set of queries prepared by BMIR-J2 comprises a primary set of 50 queries, for each of which there are five or more relevant documents, and a secondary set of 10 queries, each having one to four relevant documents. The description of a query in BMIR-J2 contains two types of information; query words/phrases used for the retrieval of documents and a short explanation of search needs the query is intended to fulfill. In experiments, we used the primary set of queries.

6.4.2 Experiment Setup and Procedure

We have conducted experiments using BMIR-J2. Our interest was in finding out whether the diversity-based summarization as formulated here is superior to relevance-based summarization, which represents a class of summarization methods broadly characterized as creating a summary from a list of sentences in the document ranked according to their salience or their likelihood of being included in a summary. As a representative of the class, we used *Z*-model described earlier, which can be thought of as a baseline model for relevance-based approaches. Another point of using *Z*-model is that since the model is already part of our diversity-based scheme, experiments would reveal true effects of the diversity component on performance. We could see how much the diversity component contributes to performance by comparing it to an approach without one.

We treated summaries as if they were stand-alone documents, and performed usual document retrieval sessions using them: which is to retrieve documents for a particular query and rank them according to the cosine similarity to the query. We scored performance in F-measure where for given P(precision) and R (recall),

$$F = \frac{2 \cdot P \cdot R}{P + R}.$$

We performed two sets of experiments which differ in what relevance scheme is employed. One scheme, which we call the *strict relevance scheme* (SRS), takes only A-labeled documents as relevant to the query, while another, called the *moderate relevance scheme* (MRS), takes both A- and B-labeled documents as relevant. BMIR-J2 recommends the latter scheme.

In each experiment, we ran a number of summarizers set at a particular compression rate, which include Z-model, a diversity-based summarizer with the standard K-means (hereafter, DBS/K), and a diversity-based summarizer with X^M-means (hereafter, DBS/X^M). In addition, largely for the purpose of reference, we ran a lead based summarizer (LEAD), which works by the simple idea of selecting the initial portion of text as a summary. DBS/K, which is identical to DBS/X^M except for the diversity component, was introduced to determine the effects of MDL on K-means. To obtain a given compression level α for DBS/X^M, we set K_{max} to the corresponding number of sentences in text: e.g. for the text of 10 sentences, $K_{max} = 5$ for $\alpha = 50\%$. Similarly for DBS/K. Let us note here that the number of sentences DBS/X^M picks for a summary may be less than K_{max}, as it is up to MDL to determine the number of clusters and it could settle for less than K_{max}. However, on BMIR-J2, DBS/X^M was found to return a K_{max} long summary for most of the time.

One feature specific to the present test domain is the use of a Japanese tokenizer ChaSen (Matsumoto et al., 1999), which breaks up sentences into words, each labeled with relevant morphological information such as part of speech. ChaSen is reported to have the accuracy of over 98% (Matsumoto et al., 1999). For index terms, we used everything except for punctuation marks, non-linguistic symbols, particles such as case marker. Furthermore, we did not use any stop-list except for those elements already excluded from the set of index terms.

Inspired in part by a finding (Morris et al., 1999) that 20% and 30% extracts are reasonably informative and comparable to the full-length text in the reading comprehension

setting, we looked at compression rates ranging from 10% to 50%. We ran a test procedure which consists of two steps.

1. At each compression rate, run Z-model, DBS/K, DBS/X^M, and LEAD on the entire BMIR-J2 collection, to produce respective pools of extracts.

2. For each query from BMIR-J2, perform a search on each pool generated, and score performance with the uninterpolated average F-measure.

Since we have two relevance schemes to consider, we did the testing under each scheme, which brought the total of retrieval sessions to somewhere around 1,200.

One problem with the use of a summary as a surrogate of its full length source in document retrieval is that the condensation process usually destroys statistical properties of a source text such as term frequency: thus for instance, terms X and Y, which may have different frequencies in the source, could end up having the same number of occurrences in a summary, which would leave us with no way of distinguishing between them in terms of term weight. One way to go about the problem is to extrapolate frequencies of index terms in a summary in order to estimate their true frequencies in its source, which we did using the following formula from Katz (1996).

$$\begin{aligned} E(k \mid k \geq m) &= \sum_{r \geq m} \left(\frac{p_r}{\sum_{j \geq m} p_j} \right) \cdot r \\ &= \frac{\sum_{r \geq m} p_r r}{\sum_{r \geq m} p_r}, \end{aligned} \tag{6-2}$$

where p_r denotes the probability of a given word occurring r times in the document and $m \geq 0$. Formula 6-2 (= (6.4) in Katz (1996)) estimates the average number of occurrences of a word in the documents, each of which contains at least m occurrences of that word. With Formula 6-2 it is possible to estimate the average frequency in the source of a word observed in a summary. For example, if we observe m occurrences of a word w in a summary, its expected frequency in its source text is given as $E(k \mid k \geq m)$.

In our experiments, we restricted ourselves to index terms with two or more occurrences in the document, so their extrapolated estimates would be $E(k \mid k \geq 2)$.

The *df* values of index terms in a summary are obtained directly from a pool of summaries.

Table 6-3 Average Performance of Z, DBS/K, DBS/X^M and LEAD under SRS. 'Full' represents a full-length document retrieval system, which runs a query on full-length documents. Figures below are in F-measure. α indicates compression rate.

α	FULL	Z	DBS/K	DBS/XM	LEAD
10%	0.230	0.203	0.211	0.227	0.230
20%	0.230	0.231	0.208	0.225	0.244
30%	0.230	0.225	0.220	0.220	0.260
40%	0.230	0.240	0.222	0.225	0.252
50%	0.230	0.234	0.227	0.235	0.258

Table 6-4 Average Performance of Z, DBS/K, DBS/X^M and LEAD under MRS.

α	FULL	Z	DBS/K	DBS/XM	LEAD
10%	0.170	0.145	0.185	0.206	0.178
20%	0.170	0.178	0.191	0.208	0.187
30%	0.170	0.194	0.209	0.223	0.203
40%	0.170	0.214	0.213	0.234	0.221
50%	0.170	0.227	0.228	0.233	0.225

6.5 Results and Discussion

Tables 6-3 and 6-4 give us a detailed picture of how Z0, DBS/K, DBS/X^{M}and LEAD compare among themselves under the two relevance schemes, i.e., moderate and strict relevance schemes. How much of a text is selected is determined by a particular compression rate set for a summary. Note however that the idea of compression rate has little consequence for a full-length retrieval system (or FULL), as it is meant to use an entire body of text, not any part of it, to determine its relevance to a query, In the tables, we put performance of FULL along with those of others for the sake of comparison. All the figures reported there are averaged F-measures over the primary queries.

Let us turn to Table 6-3. We are somewhat struck by the absence of differences in performance among the systems, which makes an interesting contrast with their performance under MRS, where the differences are more apparent. Table 6-5 and 6-6 look at p-values for pair-wise differences in performance among the summarizers. The null hypothesis here is that there is no difference in how any two systems perform. We tested the hypothesis

Table 6-5 Significance scores (P-values) for SRS. The asterisk indicates 5% significance level. We are concerned here about how much a given pair of summarizers differ in their performance on the primary queries. L denotes LEAD, X DBS/X^M K DBS/K and Z the Z model. L:X reads 'L is compared to X.'

	10%	20%	30%	40%	50%
L:X	0.9067	0.3978	0.0779	0.0666	0.1662
L:Z	0.1712	0.6704	0.0189*	0.4802	0.1992
L:K	0.2832	0.0784	0.2470	0.1609	0.1058
K:Z	0.5953	0.4751	0.8732	0.0605	0.3744
Z:X	0.2778	0.8460	0.7368	0.1842	0.8722
K:X	0.2577	0.1788	0.9952	0.7924	0.1773

Table 6-6 Significance scores (P-values) for MRS. Refer to Table 6-5 for legends.

	10%	20%	30%	40%	50%
L:X	0.0017*	0.0116*	0.0253*	0.0778*	0.1368
L:Z	0.0005*	0.6834	0.3490	0.3494	0.6801
L:K	0.4276	0.6217	0.8193	0.3511	0.5448
K:Z	0.0001*	0.5791	0.5931	0.8746	0.7410
Z:X	0.0000*	0.2033	0.0002*	0.0154*	0.0469*
K:X	0.0007*	0.0017*	0.6106	0.0061*	0.3517

by running a two-tailed t-test on each pair of summarizers. Table 6-5 gives results for SRS and Table 6-6 those for MRS. P-values breaking 5% level are marked with an asterisk.

What we find in the tables is that in MRS most of p-values at 10% are statistically significant, meaning that there is a substantial difference in performance among the systems, while in SRS none of them are, indicating that the systems perform more or less equally there. Take LEAD and DBS/X^M. Their results for MRS find that the two summarizers produce significantly different performance at compression rates from 10% to 30%, whereas in SRS the difference between the two is marginal at best.

The MRS results also show a particularly significant difference in performance between DBS/K and Z at compression rate of 10%, whose p-value is as small as 0.0001. In terms of machinery they employ, their difference comes down to the use of clustering in DBS/K and non-use of it in Z. Therefore, it is arguable that whatever difference there is in performance has been caused by clustering.

Notice also a general superiority of DBS/X^M over Z in MRS. We believe the reason has to do with a particular way DBS/X^M pulls out sentences: unlike Z, which looks at the entire text space for summary sentences, DBS/X^M is designed to search each of topic clusters or subtexts for a potential sentence to be included in summary. As a consequence, Z is more likey than DBS/X^M to fail to notice those sentences which though not on primary subjects of the text, may still be relevant to a query. We believe that B-labeled documents typically contain sentences of this kind, dealing with secondary concerns of the text which are marginally relevant to a query.

Another point about DBS/X^M and Z-model is that their differences in performance become smaller with the compression rate. This is because the Z-model selects more of the diverse sentences in the text, as the compression rate increases.

Table 6-7 and Table 6-8 break down performance of DBS/X^M by query type under SRS and MRS, respectively. The general picture is that in either scheme, DBS/X^M performs best on average for queries of type B, and moderately well for queries of type A and C. The results are somewhat contrary to our expectation since a retrieval with a query of type B is supposedly more difficult than with an A-type query.

(Gong & Liu, 2001)(G&L, hereafter) is worthy of some mention here, as it provides an interesting alternative to the present approach. In it, the authors discuss two approaches to representing the diversity of content in text: one involves selecting a representative

Table 6-7 Breakdown of average performance of DBS/X^M by query type under SRS.

query	10%	20%	30%
A	0.230	0.230	0.233
B	0.381	0.372	0.399
C	0.245	0.248	0.248
D	0.161	0.162	0.151
E	0.078	0.072	0.074
F	0.040	0.037	0.043

Table 6-8 Breakdown of average performance of DBS/X^M by query type under MRS.

query	10%	20%	30%
A	0.139	0.141	0.145
B	0.273	0.279	0.333
C	0.193	0.197	0.213
D	0.101	0.101	0.116
E	0.081	0.080	0.085
F	0.074	0.070	0.090

sentence in such a way that it may not contain any term mentioned in a pool of sentences already chosen; and another relies on singular value decomposition (SVD).

Now let us see how they compare to DBSs. As G&L find no significant difference in performance between SVD and non-SVD summarizers, we try here a simpler, non-SVD summarizer, which takes the following steps to create a summary.

1. Rank sentences in a text, according to how similar they are to the document they belong to, using a weighting scheme such as tfidf.
2. Add a top scoring sentence to a summary.
3. Remove terms mentioned in the sentence from the document.
4. Halt if the summary reaches a predefined number of sentences. If not, go to step 1.

We consider three summarizers which G&L call ATC, ATN, and ANN. Which are reported in G&L to consistently produce performance better than or equal to SVD based

Table 6-9 Average Performance of ATC, ATN, and ANN in SRS.

α	ATC	ATN	ANN
10%	0.072	0.110	0.139
20%	0.062	0.150	0.150
30%	0.113	0.211	0.211
40%	0.182	0.209	0.209
50%	0.221	0.219	0.219

summarizers. ATC and ATN both make use of what they call the augmented term weighting scheme, along with the inverse document frequency or idf.[3] ATC further normalizes term weights by the length of sentence, a feature not shared by ATN. ANN also adopts the augmented scheme but does not take into account idf or normalization.

Table 6-9 gives results for SRS and 6-10 those for MRS. In either table, we find that ATC, ATN, and ANN are all defeated by DBSs, and somewhat surprisingly also by Z and LEAD. The results indicate that going for a summary with as mutually distinct sentences as possible is not a good idea when working on IR tasks such as one here. Note that G&L's approach does not allow lexical overlap among sentences in summary. This assumption may have caused havoc to the summarizers.

To sum up, we have examined how the diversity based methods (DBSs) compare in performance to the relevance based model (Z-model) and LEAD under two paradigms, SRS and MRS. It was found that they differ more widely in MRS than in SRS: DBSs consistently outperform Z and LEAD in MRS, but does not in SRS, indicating that the diversity based methods are more sensitive to marginally relevant documents than the relevance based model.

[3]For a given term i, its augmented weight is given as $0.5 + 0.5 * TF(i)/MAX\_TF$, where $TF(i)$ denotes the frequency of term i in a text, MAX_TF denotes the maximum term frequency for terms in that text. Note that they give a somewhat unorthodox definition for the inverse document frequency, which they define as $\log(N/n(i))$. N is the total number of *sentences* in a text and $N(i)$ the number of *sentences* that contain i.

Table 6-10 Average Performance of ATC, ATN, and ANN in MRS.

α	ATC	ATN	ANN
10%	0.015	0.109	0.122
20%	0.051	0.152	0.152
30%	0.100	0.191	0.191
40%	0.141	0.234	0.214
50%	0.186	0.227	0.227

6.6 Summary

We have proposed a novel summarization paradigm where evaluation does not rely on matching extracts against human-made summaries but measuring the loss of information in extracts in terms of retrieval performance. Under this scheme, the diversity based summarization (DBS/X^M) was found to be superior to relevance based (tfidf based) summarization i.e., Z and LEAD, a standard baseline for text summarization.

We have seen that the improvement by the diversity component is more prominent under the moderate relevance scheme (MRS) than the strict relevance scheme (SRS). But this is just what is expected of performance of the diversity-based approach, because under the moderate relevance scheme, the system has to be more sensitive to marginally relevant sentences.

Chapter 7 Learning Human-Created Summaries

7.1 Introduction

In the following, we are going to turn to an interesting question of how well the diversity based summarizer models subjective judgments by humans on which sentence to include in a summary. In particular, we will explore how DBS compares to a supervised approach for summarization, which is a natural choice when the training data are available. To our knowledge, no prior work on automatic summarization addressed the question of how supervised and unsupervised approaches compare in performance when they are set to the same task. A general perception seems to be that a learning based approach works better, since it is able to exploit information about the grand truth provided by humans, which is not available to an unsupervised approach. But it is well known that learning based approaches to summarization inherently suffer from the problem of "lack of uniqueness of a summary" (Mani et al., 1998) and the cost of creating a corpus large enough to train algorithms on. Besides, it is far from clear whether the supervised approach is superior to unsupervised approach in our current concern.

Against this backdrop, we set out to empirically investigate how supervised and unsupervised approaches compare, by directly matching one against the other in the same summarization task.

7.2 Decision Tree Based Summarization

As a baseline for learning based paradigm, we consider C4.5 decision tree algorithm (Quinlan, 1993), with some modifications to accommodate an MDL based pruning.[1] Another modification involved the way the decision tree classifies data. In order to deal with a variable length summary, it was modified to output a class probability rather than a label, which allows us to rank sentences according to the probabilistic strength.

[1]We had a promising result from our earlier work (Nomoto & Matsumoto, 2000) that the extension of the decision tree with MDL pruning does improve performance to the degree comparable to AdaBoost (Freund & Schapire, 1996). See Appendix for technical details about harnessing C4.5 with MDL.

Given a set of sentences (encoded in features) that comprise a document, the decision tree outputs the probabilities of each sentence being included in a summary, which are exploited to generate a summary of a particular length. A feature description of a sentence contains information such as length, position, and linguistic cues. Listed in the following are a set of features used to encode a sentence. (Some of them are devised to specifically deal with Japanese as we use a Japanese corpus for evaluation.)

`<LocSen>` Location of sentence X, defined by:

$$\frac{\#S(X) - 1}{\#S(Last\_Sentence)}$$

'$\#S(X)$' denotes an ordinal number indicating the position of X in a text, starting with one, e.g., $\#S(k\text{th\_sentence}) = k$. 'Last_Sentence' refers to the last sentence in a text. `LocSen` takes values between 0 and $\frac{N-1}{N}$. N is the number of sentences in the text.

`<LocPar>` Location of paragraph in which sentence X occurs, given by:

$$\frac{\#Par(X) - 1}{\#Last\_Paragraph}$$

'$\#Par(X)$' denotes an ordinal number indicating the position of a paragraph containing X. '#Last_Paragraph' is the position of the last paragraph in a text, represented by the ordinal number.

`<LocWithinPar>` Location of sentence X within a paragraph in which it appears.

$$\frac{\#S(X) - \#S(Par\_Init\_Sen)}{Length(Par(X))}$$

'Par_Init_Sen' refers to the initial sentence of a paragraph in which X occurs, 'Length(Par(X))' denotes the number of sentences that occur in that paragraph. `LocWithinPar` takes continuous values ranging from 0 to 1. A paragraph initial sentence would have 0 and a paragraph final sentence 1.

`<LenText>` Text length in Japanese characters, i.e., *kana*, *kanji*.

`<LenSen>` Sentence length in *kana*/*kanji*.

Following some work in Japanese linguistics, we assume that sentence endings are relevant for identifying semantic relations among sentences. Some of the sentence endings refer

Table 7-1 Classes of sentence final forms.

code	description
1	non-past tense verbs/verbal adjectives
2	past tense verbs/verbal adjectives
3	non-past tense copula
4	nominals
5	non linguistic symbols
6	sentence final particles
0	none of above

to inflectional categories of verbs such as PAST/NON-PAST, INTERROGATIVE, and others to morphological categories like nouns and particles, e.g. question-markers. Along with Ichikawa (1990), we identified a set of sentence-ending cues and marked a sentence as to whether it contains a cue from the set.[2] Included in the set are inflectional forms of the verb and the verbal adjective, PAST/NON-PAST, morphological categories such as COPULA, and NOUN, parentheses (quotation markers), and sentence-final particles such as -ka. We use the following attribute to encode a sentence-ending form.

`<EndCue>` encodes one of sentence-ending forms described above. It is a discrete valued feature. The value ranges from 0 to 6. See Table 7-1 for the description of codes.

Finally, one of two class labels, 'Select' and 'Don't Select', is assigned to a sentence, depending on whether it is to be included in a summary. The 'Select' label is for a sentence which would be included in a summary, and the 'Don't Select' label for other cases.

One problem with the use of a decision tree as a summarizer is that it is not able to rank sentences in the document, which would be required for the generation of a variable-length summary. To deal with this we modified the way the decision tree classifies data, i.e. we re-designed it to produce, instead of a class label, the probability of a sentence being of 'Select' class, which is given as the ratio of instances of class 'Select' to the total number of instances found at a leaf node the decision tree associates with a sentence, whether or not the sentence is actually labeled as 'Select.' The ratio was further modified using the Laplace's law. Consult Appendix A.3 for further details.

[2]Word tokens are extracted by using ChaSen, a Japanese morphological analyzer which is reported to achieve the accuracy rate of over 98% (Matsumoto et al., 1999).

Table 7-2 The test data.

text type	#sentences	#paragraphs	#articles
column	17.04	4.28	25
editorial	22.32	7.48	25
news	17.60	6.40	25

7.3 Clustering based Summarization

A specific approach we adopt for the unsupervised paradigm basically follows the tripartite model we discussed in the previous chapter, namely, the diversity based summarizer (DBS/X^M), which operates by breaking a text into topically related groups of sentences and then selecting sentences from each such group. As in the previous chapter, the approach here has each sentence in the text represented as a vector of tf.idf weights representing the importance of terms in the sentence.[3] In the following, we write 'DBS' to mean DBS/X^M.

7.4 Evaluation

7.4.1 The Test Data

We asked each of 112 naive subjects (students at graduate and undergraduate level) to extract 10 % of sentences in a text which he or she considers most important in making its summary. The number of choices to make varied from two to four, depending on the length of a text. The age of subjects varied from 18 to 45. The experiments used 75 texts from three different genres (25 for each); COLUMN, EDITORIAL and NEWS WIRE. The texts were of about the same size in terms of character counts and the number of paragraphs, and were selected randomly from articles that appeared in a Japanese economics daily in 1995 (Nihon-Keizai-Shimbun-Sha, 1995). Table 8-1 provides some information on the corpus from which extraction tests are constructed. For the ease of reference, we refer to the judgments data for columns as 'G1,' those for editorials as 'G2' and those for news wire texts as 'G3.' And we refer to them collectively as 'JFD-1995.'

[3]See Equation 6-1.

Table 7-3 Human agreement in κ on summary extraction.

	column	editorial	news wire	average
$K1$	0.225 (24)	0.202 (25)	0.338 (25)	0.255
$K2$	0.355 (24)	0.336 (25)	0.455 (25)	0.383
$K3$	0.468 (23)	0.441 (25)	0.589 (25)	0.500
$K4$	0.556 (22)	0.541 (22)	0.686 (25)	0.598
$K5$	0.669 (15)	0.680 (15)	0.770 (21)	0.714

The number of subjects assigned to one extraction task varied from 4 to 9. However, 96% of the time, we had over 6 subjects working on a same task. The average number of subjects per text was 7.33. What we find in Table 8-1 is somewhat discouraging as there is only marginal agreement among subjects.

Table 7-3 shows genre-wise agreement (in κ statistic) among humans on summary extraction.[4] It also shows how the κ statistic is affected by varying agreement thresholds. By agreement threshold, we mean the minimum number of votes a sentence need to get in order to qualify for a summary sentence: thus if the agreement threshold is set to 2 (indicated by $K2$), we will assume as a summary sentence, one with two or more subjects voting for it and consider everything else a non-summary sentence. Note that at $K1$, any sentence with one or more votes is qualified for a summary sentence (we abuse the term slightly and call a corpus of sentences so marked '$K1$.') and similarly for other cases. The parenthetical figures are the number of texts from a particular genre which contain one or more sentences with the number of votes above or equal to a given threshold. Note that we have less and less relevant texts as the threshold goes up: more and more texts are thrown out because none of the sentences they contain meets the rising threshold. Note that the agreement increases with the threshold, as we might expect.

The average agreement among subjects was 0.255 at $K1$. Which is in a way consistent with Salton et al. (1999), who report a low inter-subject agreement on paragraph extracts from encyclopedias, and also with Gong and Liu (2001), who find a low inter-subject agreement in news wire summarization. While there are some work (Marcu, 1999b; Jing

[4]The kappa statistic (κ) represents one of measures of agreement for nominally scaled data, which takes the form: $\frac{P(A)-P(E)}{1-P(E)}$. It is the ratio $P(A)$ of pairwise agreement among subjects adjusted for the expected agreement ratio $P(E)$ (Siegel & Castellan, 1988).

et al., 1998), which find high agreement rates, their success may be attributed to particulars of texts used, as suggested by Jing et al. (1998). Thus, the question of whether it is possible to establish an ideal summary based on agreement is far from clear. In the face of this, it would be interesting and perhaps more fruitful to explore another view on summary: that the variability of a summary is the norm rather than the exception.

One consequence of the view is that a sentence marked as important by any *one* of the subjects becomes a potential summary sentence, however marginal it may be. We decided to adopt this view, and consequently regarded as correct, every sentence marked as important by one or more subjects.

7.4.2 Procedure

We have measured performance of the decision tree based summarizer by the usual 10-fold cross validation, whereby we split the test data into 10 blocks, reserve one for the test and use others for training. Table 7-4 gives some idea of what the test data which we call JFD-1995 looks like: the number of positive instances plummets and the value of κ climbs with the agreement threshold. (A 'positive' instance refers to that humans marked as important enough to be included in a summary.) Table 7-5 lists performance in binary (i.e., positive versus negative) classification, averaged over ten folds, of the decision tree on $K1$ to $K5$. Note that Table 7-5 is concerned with the accuracy of classification, not summarization. As shown in the table, the accuracy of classification for the test data was 60% without the pruning (CF=25%) but increased to 63% with MDL. Though the figures are low, either is well above the baseline performance, which is 51%.

Table 7-6 shows genre-wise precision (averaged over 10 folds) of the decision tree classifier (again, not the summarizer version of it) at $K1$ to $K5$. Precision here is defined as the ratio of the number of sentences correctly labeled correct over the total number of sentences the decision tree labeled correct (or 'Select'). We refer to this particular use of the term as *micro-precision* to distinguish it from its normal use, i.e., precision across all labels or categories. A dominant pattern seems to be that performance of the classifier drops with the agreement threshold, regardless of the domain it is working with.

For DBS, the only training involved was the estimation of the document frequency of a term used in sentence weighting, which was done using a collection of 14,391 articles selected from the same news paper corpus the test data came from. Also for the purpose

Table 7-4 JFD-1995. N represents the number of sentences in JFD-1995.

	N	positive	negative
$K1$	1424	707	717
$K2$	1424	392	1032
$K3$	1424	236	1188
$K4$	1424	150	1274
$K5$	1424	72	1352

Table 7-5 Effect of MDL on a two-class decision tree. Figures in the table indicate error rates. The baseline here labels everything as negative. The testing and training data are constructed with $K = 1$.

	C4.5 (CF=100%)	C4.5(CF=25%)	MDL pruned	baseline
$K1$	0.411	0.396	0.369	0.49

of comparison, we prepared other sorts of unsupervised methods. Among them is LEAD, which works simply by selecting an initial portion of text.

7.5 Results and Discussion

Tables 7-7 through 7-11 show performance in F-measure of summarizers at varying compression rates, where $F = \frac{2 \cdot P \cdot R}{P+R}$ for precision P and recall R.[5] (Each performance figure

Table 7-6 Classificatory accuracy in micro-precision

K	column	editorial	news wire
1	0.488	0.516	0.480
2	0.265	0.276	0.286
3	0.157	0.167	0.173
4	0.101	0.098	0.118
5	0.049	0.038	0.068

[5]Some caveat about recall. Giving recall as the ratio of correct sentences to the total number of sentences in text would unfairly underrate it for summaries with low compression rate. Imagine that you wish to have a summary 10% the size of its source text. Assume in addition that 50% of the source are marked

Table 7-7 Genre-wise performance of DT, DBS, and LEAD on $K1$.

	column (G1)			editorial(G2)			news wire(G3)		
α	DT	DBS	LEAD	DT	DBS	LEAD	DT	DBS	LEAD
10%	0.573	0.694	0.667	0.507	0.637	0.773	0.707	0.707	0.787
20%	0.510	0.711	0.630	0.520	0.650	0.720	0.600	0.687	0.740
30%	0.491	0.594	0.569	0.547	0.617	0.600	0.571	0.636	0.678
40%	0.514	0.583	0.551	0.541	0.611	0.557	0.548	0.631	0.598
50%	0.541	0.596	0.584	0.543	0.628	0.554	0.546	0.631	0.591

indicates an F score averaged over 10 cross validation folds, with α indicating compression rate.)

It is interesting to note that while DBS prevails over DT and LEAD in G1, it is outperformed by LEAD on G3. On G2, DBS tends to pick up with larger compression rates. At $\alpha \geq 30\%$, DBS performs superior to either DT or LEAD, apparently regardless of level of agreement or K, with the notable exception of $K5$, where LEAD dominates DBS and DT. Also we find that on the news data set, LEAD significantly outperforms DBS and DT on $K1$ through $K5$.

Now the question is why we don't have one particular summarizer consistently outperforming others across the data sets. LEAD works best on G3 but not on G2 or on G1, while DBS works best on G1 and not on other domains. Why is this so?

To get some insight into possible relations between domain and performance of summarizers, we split a text into 10 equal-sized contiguous segments of sentences and counted votes won collectively by sentences in each segment. Each panel in Figure 7-1 and 7-2 shows on the horizontal axis the position of a given segment, and on the vertical axis a normalized count of votes for that segment, i.e., the ratio of votes it has won over the total count of votes. Thus for a text of 20 sentences, the first segment starts with the initial

as correct. The problem is, even if you are right about every sentence you put in the 10% summary, you are still left with recall of 25%. That is, there is no way that a 10% summary could achieve 100% recall. A way out would be to use the following denominator for recall.

$$D = \begin{cases} \alpha \cdot |T| & \text{if } \alpha \cdot |T| < |C_T| \ (\alpha > 0) \\ |C_T| & \text{otherwise} \end{cases} \tag{7-1}$$

α denotes compression ratio, $|T|$ is the length of text T and C_T the number of correct sentences in T. We use the modified recall throughout the section.

Table 7-8 Genre-wise performance of DT, DBS, and LEAD on $K2$.

	column (G1)			editorial(G2)			news wire(G3)		
α	DT	DBS	LEAD	DT	DBS	LEAD	DT	DBS	LEAD
10%	0.376	0.392	0.416	0.227	0.383	0.520	0.560	0.493	0.627
20%	0.355	0.406	0.408	0.273	0.442	0.473	0.456	0.475	0.560
30%	0.349	0.372	0.358	0.280	0.417	0.368	0.428	0.485	0.517
40%	0.410	0.415	0.323	0.320	0.412	0.328	0.429	0.506	0.484
50%	0.408	0.413	0.345	0.348	0.422	0.350	0.425	0.494	0.472

Table 7-9 Genre-wise performance of DT, DBS, and LEAD on $K3$.

	column (G1)			editorial(G2)			news wire(G3)		
α	DT	DBS	LEAD	DT	DBS	LEAD	DT	DBS	LEAD
10%	0.208	0.229	0.253	0.168	0.213	0.341	0.444	0.385	0.500
20%	0.212	0.272	0.261	0.203	0.296	0.312	0.374	0.365	0.473
30%	0.237	0.282	0.235	0.203	0.317	0.273	0.350	0.424	0.444
40%	0.304	0.307	0.206	0.213	0.300	0.233	0.342	0.407	0.386
50%	0.302	0.295	0.220	0.240	0.302	0.238	0.333	0.367	0.351

Table 7-10 Genre-wise performance of DT, DBS, and LEAD on $K4$.

	column (G1)			editorial(G2)			news wire(G3)		
α	DT	DBS	LEAD	DT	DBS	LEAD	DT	DBS	LEAD
10%	0.191	0.142	0.194	0.126	0.130	0.295	0.412	0.357	0.460
20%	0.211	0.181	0.207	0.129	0.211	0.269	0.340	0.336	0.441
30%	0.193	0.215	0.186	0.118	0.217	0.214	0.284	0.366	0.390
40%	0.234	0.218	0.156	0.145	0.196	0.181	0.264	0.317	0.313
50%	0.225	0.203	0.152	0.188	0.195	0.166	0.266	0.290	0.268

Table 7-11 Genre-wise performance of DT, DBS, and LEAD on $K5$.

	column (G1)			editorial(G2)			news wire(G3)		
α	DT	DBS	LEAD	DT	DBS	LEAD	DT	DBS	LEAD
10%	0.144	0.147	0.175	0.093	0.113	0.273	0.402	0.271	0.441
20%	0.163	0.158	0.167	0.098	0.159	0.222	0.328	0.260	0.392
30%	0.136	0.153	0.136	0.071	0.134	0.162	0.276	0.268	0.311
40%	0.188	0.154	0.107	0.070	0.117	0.141	0.228	0.228	0.246
50%	0.165	0.136	0.099	0.093	0.137	0.128	0.212	0.213	0.204

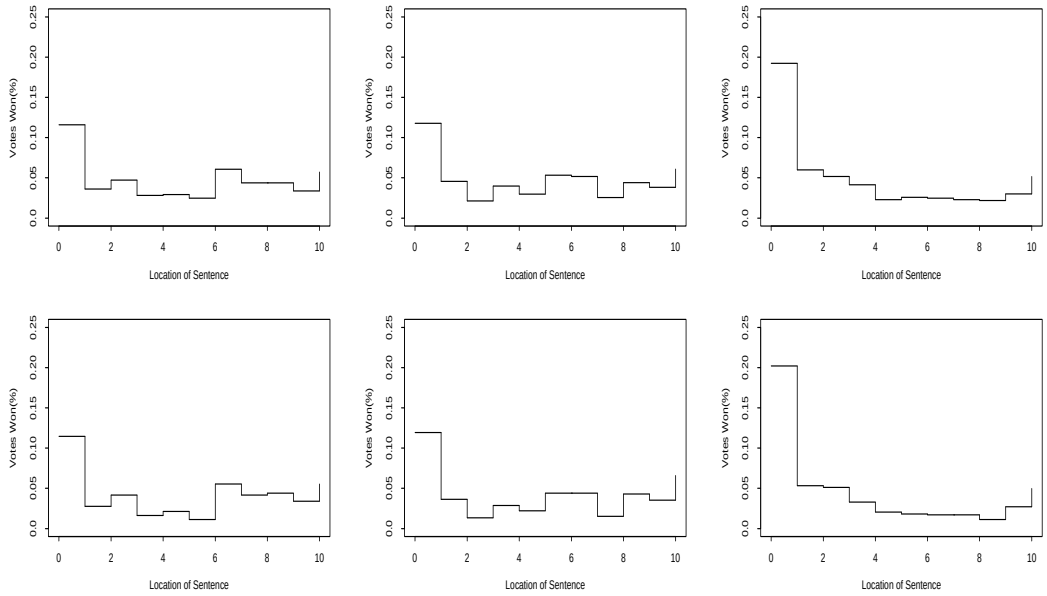

Figure 7-1 Each panel here shows the ratio of votes won by a block of sentences over the total number of votes cast, for various Ks and domains. The rows (from top to bottom) represent $K1$ and $K2$. The columns represent various domains.

sentence and continues until the second, followed by the second segment which starts with the third sentence and lasts until the fourth, and so on.

In Figure 7-1 we find two rows each consisting of three panels. The top row gives results for $K1$ texts, namely, those texts where any sentence with one or more votes is marked as correct, and the bottom row results for $K2$ texts. Each panel in a row shows how the popularity changes over successive segments of text from a particular domain: left panel represents G1, center G2 and right G3. Analogously, Figure 7-2 gives results for $K3$ (top), $K4$ (center), and $K5$ (bottom).

What is striking about these results is that the distribution of votes (or DOV for short) has a shape specific to a particular domain and remains stable over $K1$ to $K5$: the DOV for G1 on $K5$, for instance, remains similar to that on $K1$, with characteristic features in tact such as high peak at the beginning and lower hill towards the end, though it looks like a somewhat meager version of the latter. The DOV for the news articles (G3), in contrast, has a sharp peak at the beginning and then tapers off, features shared by all

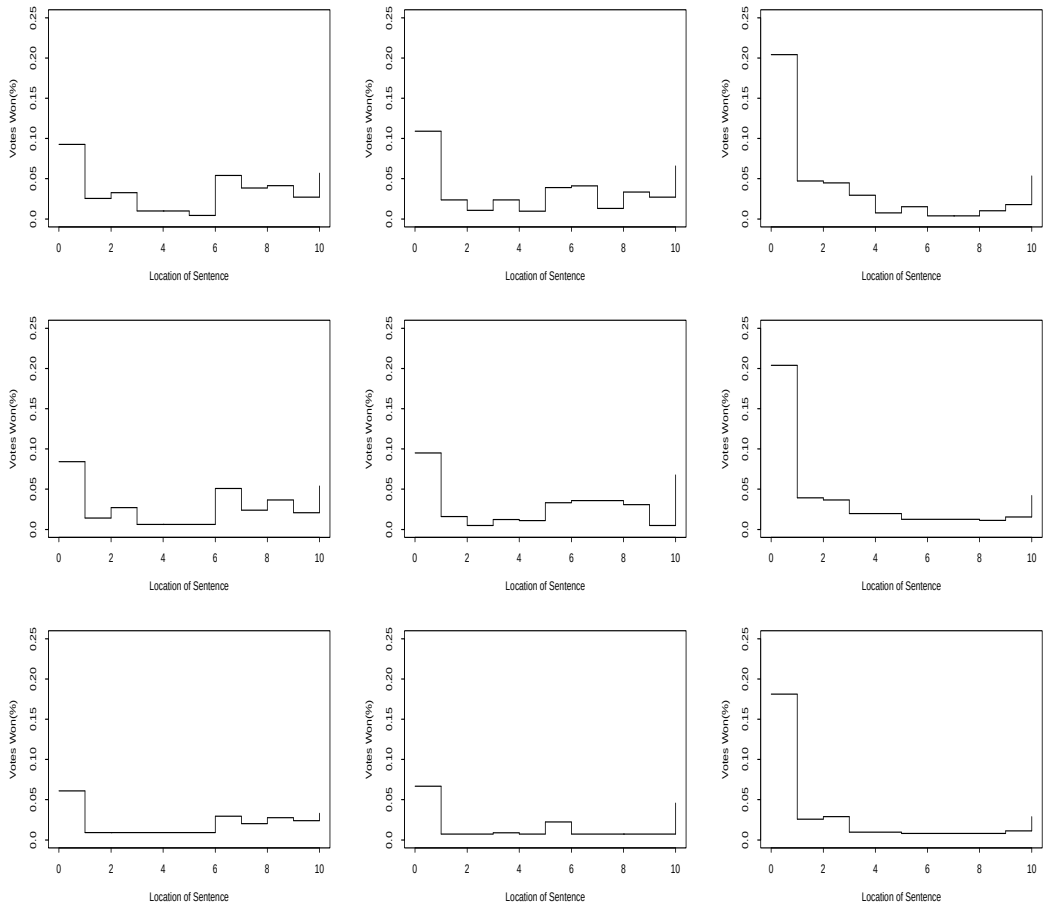

Figure 7-2 Continued from Fig.7-1. We are looking at results for $K3$ (top row), $K4$ (center row), and $K5$ (bottom row).

of its versions for $K2$ to $K5$. The DOV of the editorials or G2 looks similar to that for the columns, featuring a text-initial peak and bumps at the end but distinguishes itself by having a ragged hump in the middle.

Interestingly enough, DOVs help us see why we do not have a single summarizer prevailing on all of the domains, i.e., G1 to G3. Take DBS. As we noted earlier, DBS generally outperforms DT and LEAD in the columns domain. It happens probably because votes in columns are more evenly distributed or more widely scattered over segments than, say, in news articles. Therefore, it pays more to search various parts of text for candidate summary sentences, as DBS does, than to look at a particular region of text (which is what LEAD does). By contrast, if one turns to the editorial domain, the converse holds, that is, it pays more to stick to one particular region than to look everywhere, as text initial segments take most of votes. This is why LEAD beats DBS in that domain.

It is more difficult to explain why DT is performing as it does. DT tends to fare worse than DBS and LEAD. However, at $K4$ on the columns domain, DT is generally better off than the other two. It is not clear at the moment why DT does not do as well on the editorial domain as on the columns domain, despite the similarity of the corresponding DOVs.

7.6 Summary

We ran experiments using summary judgments elicited from human subjects, and found DBS and LEAD to be superior in performance to DT, which is somewhat unexpected given that DT is allowed to use through learning information on human judgments, an issue we continue to discuss in chapter 8. Then we asked whether the level of agreement among humans on their judgments as well as the type of domain have any effect on performance of the summarizers. The results suggest that the type of domain, or a particular shape it has for the distribution of votes (DOV) is more likely to affect how the summarizers would perform. In chapter 9 we will return to the issue of DOVs, and examine an approach to summarization that directly exploits DOVs by modeling them through probability distributions.

Chapter 8 Supervised Ranking in Summarization

8.1 Introduction

In chapter 7, we found that an unsupervised method based on clustering (DBSX^M) sometimes better approximates human created extracts than a supervised, decision tree based approach (ProbDT). That appears somewhat contradictory given that a supervised approach should be able to exploit human supplied information about which sentence to include in an extract and which not to, whereas an unsupervised approach blindly chooses sentences according to some selection scheme. An interesting question is, why this should be the case.

The reason may have to do with variation in human judgments on sentence selection for a summary. In a study to be described below, we asked students to select 10% of a text which they find most important for making a summary. If they agree perfectly on their judgments, then we will have only 10% of a text selected as most important. However, what we found was that about a half of text were marked as important, indicating that judgments vary widely among humans.

Curiously, though, (Nomoto & Matsumoto, 2001a) also found that ProbDT fares much better when tested on data exhibiting high agreement among humans than an unsupervised, clustering based system, i.e., DBSX^M or DBS for short. Their finding suggests that there are indeed some regularities (or biases) to be found (and exploited).

Results so far on human summaries illuminate two aspects to judgments humans make when summarizing texts; their judgments vary but exhibit some consistent biases which could be usefully exploited. The issue is then how we might model them in some coherent framework.

In this chapter, we will take a look at a possible integration of ProbDT and DBS as a way of resolving apparently conflicting properties of human summaries. We will show how coupling both paradigms provides us with a better way of approximating human judgments than either of the two considered alone. To our knowledge, no prior work on summarization, e.g., Kupiec et al. (1995) explicitly tackled the issue of the variability inherent in human

judgments in summarization. The details of DBS and ProbDT can be found either in previous chapters or appendices.

8.2 Combining ProbDT and DBS

Combining ProbDT and DBS is a straightforward business. All that it involves is to replace Reduce-Redundancy with ProbDT. Thus instead of picking up a sentence scoring highest in tfdif, DBS/ProbDT attempts to find a sentences with the highest score for $P(Select \mid \vec{u}, \mathrm{DT})$. We consider here two variations on ProbDT. One extends a DT with MDL based pruning, and the other relies on a particular decision tree known as Subspace Splitting Decision Tree, or SSDT. We start with the MDL version.

8.2.1 MDL-DT

MDL-DT stands for a decision tree with MDL based pruning. It strives to optimize the decision tree by pruning the tree in such a way as to produce the shortest (minimum) description length for the tree. The description length refers to the number of bits required for encoding information about the decision tree. MDL ranks, along with Akaike Information Criterion (AIC) and Bayes Information Criterion (BIC), as a standard criterion in machine learning and statistics for choosing among possible (statistical) models. As shown empirically in Nomoto and Matsumoto (2000) for discourse domain, pruning DT with MDL significantly reduces the size of tree, while not compromising performance.

8.2.2 SSDT

SSDT or Subspace Splitting Decision Tree represents another form of decision tree algorithm.(Wang & Yu, 2001) The goal of SSDT is to discover patterns in highly biased data, where a target class, i.e., the class one likes to discover something about, accounts for a tiny fraction of the whole data. Note that the issue of biased data distribution is particularly relevant for summarization, as a set of sentences to be identified as *wis* usually account for a very small portion of the data.

SSDT begins by searching the entire data space for a cluster of positive cases and *grows* the cluster by adding points that fall within some distance to the center of the cluster. If the splitting based on the cluster offers a better Gini index than simply using one of the attributes to split the data, SSDT splits the data space based on the cluster, that is, forms

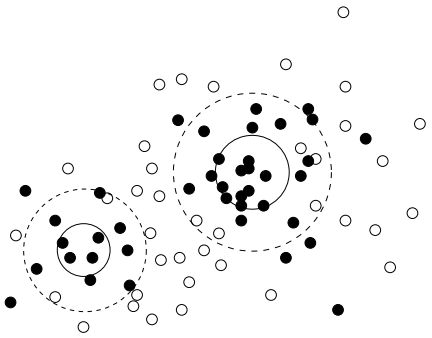

Figure 8-1 SSDT in action. Filled circles represent positive class, white circles represent negative class. SSDT starts with a small spherical cluster of positive points (solid circle) and grows the cluster by 'absorbing' positive points around it (dashed circle).

one region outside of the cluster and one inside.[1] It repeats the process recursively on each subregions spawned until termination conditions are met. Figure 8-1 gives a snapshot of SSDT at work. SSDT locates some clusters of positive points, develops spherical clusters around them.

With its particular focus on positive cases, SSDT is able to provide a more precise characterization of them, compared, for instance, to C4.5.

8.3 Test Data and Procedure

We asked 112 Japanese subjects (students at graduate and undergraduate level) to extract 10% sentences in a text which they consider most important in making a summary.[2] The number of sentences to extract varied from two to four, depending on the length of a text. The age of subjects varied from 18 to 45. We used 75 texts from three different categories (25 for each category); column, editorial and news report. Texts were of about the same size in terms of character counts and the number of paragraphs, and were selected randomly from articles that appeared in a Japanese financial daily (Nihon-Keizai-Shimbun-Sha, 1995). There were, on average, 19.98 sentences per text.

[1]For a set S of data with k classes, its Gini index is given as: $Gini(S) = 1 - \sum_i^k p_i^2$, where p_i denotes the probability of observing class i in S.

[2]The test data we will be using is same as one we described in Chapter 7, i.e., JFD-1995. We are repeating here the description of how it came about for the convenience's sake only.

Table 8-1 Test Data. N denotes the total number of sentences in the test data. The figures under K mean the minimum number of votes a sentence need to get in order to be considered a *wis* (positive) sentence.

K	N	positive	negative
1	1424	707	717
2	1424	392	1032
3	1424	236	1188
4	1424	150	1274
5	1424	72	1352

The kappa agreement among subjects was 0.25. The result is in a way consistent with Salton et al. (1999), who report a low inter-subject agreement on paragraph extracts from encyclopedias and also with Gong and Liu (2001) on a sentence selection task in the cable news domain. While there are some work (Marcu, 1999b; Jing et al., 1998) which do report high agreement rates, their success may be attributed to particulars of texts used, as suggested by Jing et al. (1998). Thus, the question of whether it is possible to establish an ideal summary based on agreement is far from settled, if ever. In the face of this, it would be interesting and perhaps more fruitful to explore another view on summary, that the variability of a summary is the norm rather than the exception.

In the experiments that follow, we decided not to rely on a particular level of inter-coder agreement to determine whether or not a given sentence is *wis*. Instead, we used *agreement threshold* to distinguish between *wis* and non-*wis* sentences: for a given threshold K, a sentence is considered *wis* (or positive) if it has at least K votes in favor of its inclusion in a summary, and non-*wis* (negative) if not. Thus if a sentence is labeled as positive at $K = 1$, it means that there are one or more judges taking that sentence as *wis*. In particular, we write 'K1' to mean a version of JFD-1995 with $K = 1$. We examined K from 1 to 5. (On average, seven people are assigned to one article. However, one would rarely see all of them unanimously agree on their judgments.)

Table 8-1 shows how many positive/negative instances one would get at a given agreement threshold. On $K1$, out of 1424 instances, i.e., sentences, 707 of them are marked positive and 717 are marked negative, so positive and negative instances are evenly spread across the data. On the other hand, on $K5$, there are only 72 positive instances. This means that there is less than one occurrence of *wis* case per article.

In the experiments below, each probabilistic rendering of the DTs, namely, C4.5, MDL-DT, and SSDT is trained on the corpus, and tested with and without the diversity extension (Find-Diversity). When used without the diversity component, each ProbDT works on a test article in its entirety, producing the ranked list of sentences. A summary with compression rate γ is obtained by selecting top γ percent of the list. When coupled with Find-Diversity, on the other hand, each ProbDT is set to work on each cluster discovered by the diversity component, producing multiple lists of sentences, each corresponding to one of the clusters identified. A summary is formed by collecting top ranking sentences from each list.

Evaluation was done by 10-fold cross validation. For the purpose of comparison, we also ran the diversity based model as given in Chapter 6 and a tfidf based ranking model (call it Z model), which simply ranks sentences according to the tfidf score and selects those which rank highest. Also recall from Chapter 6 that the diversity based model (DBS) consists in Find-Diversity and the tfidf based ranking model, called Reduce-Redundancy.

8.4 Results and Discussion

Tables 8-2-8-6 show performance of each ProbDT and its combination with the diversity (clustering) component. It also shows performance of Z model and DBS. In the tables, the slashed 'V' after the name of a classifier indicates that the relevant classifier is diversity-enabled, meaning that it is coupled with the diversity extension. Notice that each decision tree here is a ProbDT and should not be confused with its non-probabilistic counterpart. Also worth noting is that DBS is in fact Z/V, that is, diversity-enabled Z model.

Returning to the tables, we find that for most of the times, the diversity component has clear effects on ProbDTs, significantly improving their performance. All the figures are in F-measure, i.e., $F = \frac{2*P*R}{P+R}$. In fact this happens regardless of a particular choice of ranking model, as performance of Z is also boosted with the diversity component. Not surprisingly, effects of supervised learning are also evident: diversity-enabled ProbDTs generally outperform DBS (Z/V) by a large margin. What is surprising, moreover, is that diversity-enabled ProbDTs are superior in performance to their non-diversity counterparts (with a notable exception for SSDT on $K1$), which suggests that selecting marginal sentences is an important part of generating a summary.

Table 8-2 Performance at varying compression rates for $K1$. MDL-DT denotes a summarizer based on C4.5 with the MDL extension. DBS (=Z/V) denotes the diversity based summarizer. Z represents the Z-model summarizer. Performance figures are in F-measure. 'V' indicates that the relevant classifier is diversity-enabled. α indicates compression rate. Note that DBS =Z/V.

α	C4.5	C4.5/V	MDL-DT	MDL-DT/V	SSDT	SSDT/V	DBS	Z
0.2	0.371	**0.459**	0.353	**0.418**	0.437	**0.454**	0.429	0.231
0.3	0.478	**0.507**	0.453	**0.491**	**0.527**	0.517	0.491	0.340
0.4	0.549	**0.554**	0.535	**0.545**	**0.605**	0.553	0.529	0.435
0.5	**0.614**	0.600	0.585	**0.593**	**0.639**	0.606	0.582	0.510

Another observation about the results is that as one goes along with a larger K, differences in performance among the systems become ever smaller: on $K5$, Z performs comparably to C4.5, MDL, and SSDT either with or without the diversity component. The decline of performance of the DTs may be caused by either the absence of recurring patterns in data with a higher K or simply the paucity of positive instances. At the moment, we do not know which is the case here.

It is curious to note, moreover, that MDL-DT is not performing as well as C4.5 and SSDT on $K1$, $K2$, and $K3$. The reason may well have to do with the general properties of MDL-DT. Recall that MDL-DT is designed to produce as small a decision tree as possible. Therefore, the resulting tree would have a very small number of nodes covering the entire data space. Consider, for instance, a hypothetical data space in Figure 8-2. Assume that MDL-DT bisects the space into region A and B, producing a two-node decision tree. The problem with the tree is, of course, that point x and y in region B will be assigned to the *same probability* under the probabilistic tree model, despite the fact that point x is very close to region A and point y is far out. This problem could happen with C4.5, but in MDL-DT, which covers a large space with a few nodes, points in a region could be far apart, making the problem more acute. Thus the poor performance of MDL-DT may be attributable to its extensive use of pruning.

Table 8-3 Performance on $K2$

α	C4.5	C4.5/V	MDL-DT	MDL-DT/V	SSDT	SSDT/V	DBS	Z
0.2	0.381	**0.441**	0.343	**0.391**	0.395	**0.412**	0.386	0.216
0.3	0.420	**0.441**	0.366	**0.418**	0.404	**0.431**	0.421	0.290
0.4	0.434	**0.444**	0.398	**0.430**	0.415	**0.444**	0.444	0.344
0.5	0.427	**0.447**	0.409	**0.437**	0.423	**0.439**	0.443	0.381

Table 8-4 Performance on $K3$

α	C4.5	C4.5/V	MDL-DT	MDL-DT/V	SSDT	SSDT/V	DBS	Z
0.2	0.320	**0.354**	0.297	**0.345**	0.328	**0.330**	0.314	0.314
0.3	0.300	**0.371**	0.278	**0.350**	0.321	**0.338**	0.342	0.349
0.4	0.297	**0.357**	0.298	**0.348**	0.325	**0.340**	0.339	0.337
0.5	0.297	**0.337**	0.301	**0.329**	0.307	**0.327**	0.322	0.322

Table 8-5 Performance on $K4$

α	C4.5	C4.5/V	MDL-DT	MDL-DT/V	SSDT	SSDT/V	DBS	Z
0.2	0.272	**0.283**	0.285	**0.301**	0.254	**0.261**	0.245	0.245
0.3	0.229	**0.280**	0.234	**0.284**	0.249	**0.267**	0.269	0.269
0.4	0.238	**0.270**	0.243	**0.267**	0.236	**0.248**	0.247	0.247
0.5	0.235	**0.240**	0.245	**0.246**	0.227	**0.233**	0.232	0.232

Table 8-6 Performance on $K5$

α	C4.5	C4.5/V	MDL-DT	MDL-DT/V	SSDT	SSDT/V	DBS	Z
0.2	**0.242**	0.226	**0.252**	0.240	0.188	**0.189**	0.191	0.191
0.3	0.194	**0.220**	0.197	**0.231**	0.171	**0.206**	0.194	0.194
0.4	0.184	**0.189**	0.189	**0.208**	**0.175**	0.173	0.173	0.173
0.5	0.174	**0.175**	0.176	**0.191**	0.145	**0.178**	0.167	0.167

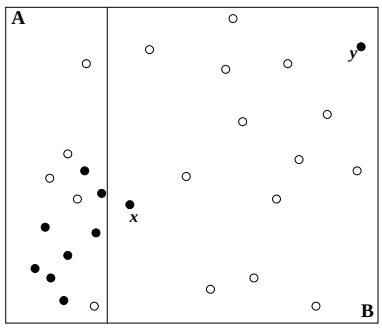

Figure 8-2 Hypothetical Data Space

8.5 Summary

As a way of exploiting human biases towards an increased performance of the summarizer, we have explored approaches to embedding supervised learning within a general unsupervised framework. In the paper, we focused on the use of decision tree as a plug-in learner. We have shown empirically that the idea works for a number of decision trees, including C4.5, MDL-DT and SSDT. Coupled with the learning component, the unsupervised summarizer based on clustering significantly improved its performance on the corpus of human created summaries. More importantly, we found that supervised learners perform better when coupled with the clustering than when working alone. We argued that that has to do with the high variation in human created summaries: the clustering component forces a decision tree to pay more attention to sentences marginally relevant to the main thread of the text.

While ProbDTs appear to work well with ranking, it is also possible to take a different approach: for instance, we may use some distance metric in instead of probability to distinguish among sentences. It would be interesting to invoke the notion like *prototype modeler* (Kalton, Langely, Wagstaff, & Yoo, 2001) and see how it might fare when used as a ranking model.

Moreover, it may be worthwhile to explore some non-clustering approaches to representing the diversity of contents of a text, such as Gong and Liu (2001)'s summarizer 1 (GLS1, for short), where a sentence is selected on the basis of its similarity to the text it belongs to, but which excludes terms that appear in previously selected sentences. While our preliminary study indicates that GLS1 produces performance comparable and even superior to DBS on some tasks in the document retrieval domain, we have no results available at the moment on the efficacy of combining GLS1 and ProbDT on sentence extraction tasks.

Finally, we note that the test corpus used for evaluation is somewhat artificial in the sense that we elicit judgments from people on the summary-worthiness of a particular sentence in the text. Perhaps, we should look at naturally occurring abstracts or extracts as a potential source for training/evaluation data for summarization research. Besides being natural, they usually come in large number, which may alleviate some concern about the lack of sufficient resources for training learning algorithms in summarization.

Chapter 9 Bayesian Learning in Text Summarization

9.1 Introduction

Consider Figure 9-1. What is shown there is the proportion of the times that sentences at particular locations are judged as relevant to summarization, or worthy of inclusion in a summary. Each panel shows judgment results on 25 Japanese texts of a particular genre; columns (G1K3), editorials (G2K3) and news stories (G3K3). All the documents are from a single Japanese news paper, and judgments are elicited from some 100 undergraduate students. While more will be given on the details of the data later (Section 9.3.2), we can safely ignore them here.

Each panel has the horizontal axis representing location or order of sentence in a document, and the vertical axis the proportion of the times sentences at particular locations are picked as relevant to summarization. Thus in G1K3, we see that the first sentence (to appear in a document) gets voted for about 12% of the time, while the 26th sentence is voted for less than 2% of the time.

Curiously enough, each of the panels exhibits a distinct pattern in how votes are spread across a document: G1K3 has the distribution of votes (DOV) with sharp peaks around 1 and 14; in G2K3, the distribution is peaked around 1 and has some dull swell around 19; in G3K3, the distribution is sharply skewed to the left, indicating that the majority of votes went to the initial section of a document – an interesting parallel to *news schema* (Van Dijk, 1988). A natural question is then, can we somehow exploit the DOV to the service of summarization? In this chapter, we will be talking about about how we could do this under a Bayesian modeling framework, which allows us to explicitly represent and make use of the DOV by way of Dirichlet posterior (Congdon, 2003).

9.2 Bayesian Model of Summaries

Let us begin by some preliminaries. Take a document D of n sentences. Suppose that we want to do extractive summarization on D, and knew the probability that each i-th

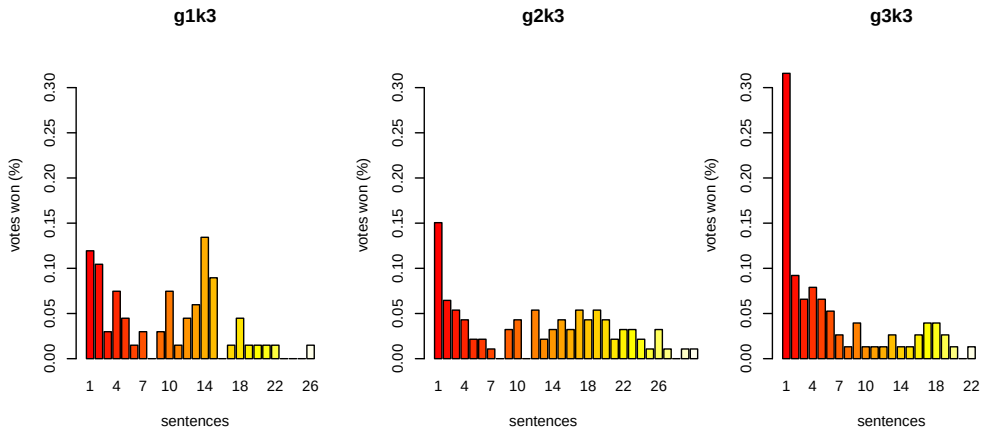

Figure 9-1 Genre-by-genre vote distribution

$$\Pr(s; D, l) = \begin{cases} \frac{1}{Z} \prod_{i \in s} \Pr(Y = i | D, \mathcal{S}, \mathcal{V}, \mathcal{U}) & \text{if } |s| = l \text{ and } s \text{ contains no duplicates.} \\ 0 & \text{otherwise} \end{cases} \tag{9-2}$$

sentence ($1 \leq i \leq n$) is judged as relevant to (and thus included in) a possible summary, which we might just as well represent by $\Pr(Y = i|D)$, i.e., the probability of a multinomial variable Y taking i as its value, given D. Then we could regard a summary as a set of random draws $Y^{(1)}, \ldots, Y^{(k)}$ from this multinomial distribution, i.e.,

$$Y^{(j)}|D \sim \text{Mult}(\xi) \tag{9-1}$$

where $\xi = (\xi_i, \ldots, \xi_n)$, $\xi_i = \Pr(Y = i|D)$, and $\sum_i \xi_i = 1$. '$X \sim Z$' is read as 'X is distributed as according to Z.' 'Mult(ξ)' represents a multinomial distribution with parameters $\xi_1, \ldots \xi_n$. $Y^{(j)}$ represents the j-th random draw. To make the DOV part of the story, we extend the distribution model 9-1 to include, among others, a parameter that represents it. In particular we define the probability distribution of summaries with length l, for document D, by Equation 9-2, where $\mathcal{S}$ represents whatever information we already have about sentences; $\mathcal{V}$ stands for the DOV; $\mathcal{U}$ is a hyperparameter explained later; and Z is a normalizing constant. For convenience, we write y_i to mean '$Y = i$,' hereafter.

Now one useful, and perhaps enlightening way to look at the DOV in the current framework is by analogy to collaborative filtering (CF) (Yu, Tresp, & Yu, 2004); part of

the deal with CF is to invoke and exploit a notion of *user profile*, which usually means a set of particular ratings or choices made by the user – here we could think of it as representing particular positional preferences a person may have when working on a summary. We could then take the probability of a given sentence i being part of a summary as the sum of ξ_i over user profiles that gave rise to the DOV, which is an amalgam of them. Since however we know nothing about each individual profile, we marginalize over them, which would give us:

$$\Pr(y_i|D,\mathcal{S},\mathcal{V},\mathcal{U}) = \int \Pr(y_i|D,\mathcal{S},\mathcal{V},\theta,\mathcal{U}) \Pr(\theta\,|\mathcal{V},\mathcal{U})\, d\theta. \tag{9-3}$$

Here $\theta = (p_1,\ldots,p_n)$ stands for some user profile with $\sum_i p_i = 1$. (Think of p_i as a probability measure indicating how much a particular person, whose profile θ is, is inclined to pick the i-th sentence as part of a summary.) Note also that equation 9-3 gives a ξ_i averaged over a possibly infinite number of profiles. As finding an exact solution to the integral in equation 9-3, however, is a practical impossibility, we will fall back to a somewhat degenerate, though efficient, approximation method suggested by Yu et al. (2004) and Cowans (2004), where one seeks a solution at a particular θ, namely, posterior mode or MAP (maximum a posterior) estimate of θ – which translates into expectation $E[\theta|\mathcal{V},\mathcal{U}]$ in the Bayesian paradigm, which happens to permit an exact solution. (See Appendix A.4.) As for $\mathcal{V}$, we take it as as a vector $(v_1,\ldots,v_n)$ of observed counts of the times a sentence at each i-th place ($1 \le i \le n$) is picked by people. We take $\mathcal{U}$ as a vector $(u_1,\ldots,u_n)$ of uniform parameters, where $u_i > 0$. Therefore equation 9-3 reduces to:

$$\begin{aligned} &\Pr(y_i|D,\mathcal{S},\mathcal{V},\mathcal{U}) \\ &\quad\approx \Pr(y_i|D,\mathcal{S},\mathcal{V},\mathcal{U},\hat{\theta}) \Pr(\hat{\theta}\,|\mathcal{V},\mathcal{U}) \\ &\quad= \Pr(y_i|D,\mathcal{S},\hat{\theta}) \Pr(\hat{\theta}\,|\mathcal{V},\mathcal{U}) \end{aligned} \tag{9-4}$$

where $\hat{\theta} = E[\theta|\mathcal{V},\mathcal{U}]$.

We have yet to work out the likelihood function and the prior that are part of equation 9-4. Let $\Pr(\theta|\mathcal{V},\mathcal{U})$ be given by $\text{Dirichlet}(\theta|\mathcal{V},\mathcal{U})$, i.e., a Dirichlet posterior of θ, given $\mathcal{V}$ and $\mathcal{U}$.[1] By noting that $\text{Dirichlet}(\theta|\mathcal{V},\mathcal{U}) = \text{Dirichlet}(\theta|\mathcal{V}+\mathcal{U})$, we have

$$\Pr(\theta|\mathcal{V},\mathcal{U}) = \text{Dirichlet}(\theta|\mathcal{V}+\mathcal{U}).$$

[1]The Dirichlet distribution is known as a 'natural' prior for a multinomial distribution which ξ_i is (Congdon, 2003).

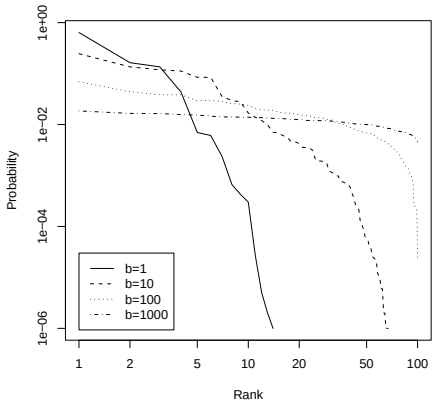

Figure 9-2 Ziph plot of random samples from Dirichlet distributions with $\lambda = 1, 10, 100,$ and 1000.

We may rewrite $\mathcal{U}$ as $\lambda\mathbf{u}$ for some positive scalar λ, such that $u_i = 1/n$ and $\mathbf{u} = (u_1, \ldots, u_n)$. λ is known to influence the shape of distribution; i.e., small λ tends to produce a distribution over $\mathbf{u}$ sharply peaked around some u's (MacKay & Peto, 1994).

Figure 9.2 shows a Ziphian plot of random samples from Dirichlet distributions $\Pr(\mathbf{p}|\lambda\mathbf{u})$ with various values for λ ('b' in the figure), where $\mathbf{p} = (p_1, \ldots, p_{100})$, and $\mathbf{u}$ is given as before except that we have $u_i = 1/100$.[2] On the x-axis we have an array of parameters p_i ranked by the magnitude of their probabilities, which are given on the y-axis. It is readily seen in the figure that a larger value of λ gives a more leveled distribution.

As for the likelihood function in equation 9-4, i.e., $\Pr(y_i|D, \mathcal{S}, \mathcal{V}, \mathcal{U}, \hat{\theta})$, it could be any probability function. Indeed, any discriminative (pattern) classifier will do, as long as it produces the probability of y_i conditioned on D, $\mathcal{S}$ and $\hat{\theta}$.

Now here is how we may work with a classifier. We begin by randomly resampling data with replacement from the training data $\mathcal{S}$ according to $\hat{\theta}$. We then move on to train a classifier on the resampled data, and run it on the test document D to find, for each sentence in D, its probability of being a 'pick' sentence. Because resampling does not in general lead to a unique set of data, we take $\Pr(y_i|D, \mathcal{S}, \hat{\theta})$, for a given classifier f, to be

[2]Here $p_1, \ldots, p_{100}$ are generated using the GNU Scientific Library which samples and renormalizes each value from a Gamma distribution with $\alpha = \lambda u_i$ and $\beta = 1$.

the average likelihood f produces for $Y = i$ over 100 samples drawn from Mult($\hat{\theta}$). Note an interesting parallel to boosting (Freund & Schapire, 1996); the Dirichlet posterior forces classifiers to attend to some specific parts of text, and ignore others. How narrowly focused they are is largely controlled by the value of λ.

Having equation 9-4 worked out, we are ready to deliver a summarizer based on it, which we call the 'Bayesian summarist' or simply 'BAYS.' For document D and compression length l, the Bayesian summarist produces a summary or a subset of D according to:

$$\mathcal{G}(D, l) = \arg\max_{s \subseteq D} \Pr(s; D, l), \tag{9-5}$$

to which dynamic programming provides a straightforward solution.

9.3 Working with Bayesian Summarist

9.3.1 Pattern Classifiers

We couple BAYS with some of the well known classifiers to see how it leverages their performance. In particular we compare two versions of each classifier; one with BAYS and one without. Classifiers we study here include: Adtree (alternating decision tree) (Freund & Mason, 1999), Naive Bayes, Kstar (an instance-based classifier with a particular distance measure) (Cleary & Trigg, 1995), and C45 (classical decision tree) (Quinlan, 1993). We used Weka implementations of the algorithms (with default settings) in experiments described below (Witten & Frank, 2000).

The attributes are broadly intended to represent some aspects of a sentence in a document, an object of interest here. Thus for each sentence ψ, its encoding involves reference to the following set of attributes or features. 'LocSen' gives a normalized location of ψ in the text, i.e., a normalized distance from the top of the text; likewise, 'LocPar' gives a normalized location of the paragraph in which ψ occurs, and 'LocWithinPar' records its normalized location within a paragraph. Also included are a few length-related features such as the length of text and sentence. Furthermore we brought in some language specific feature which we call 'EndCue.' It records the morphology of a linguistic element that ends ψ, such as inflection, part of speech, etc.

In addition, we make use of the weight feature ('Weight') for a record on the importance of ψ based on tf.idf. Let $\psi = w_1, \ldots, w_n$, for some word w_i. Then the weight $W(\psi)$ is

Table 9-1 N represents the number of sentences in G1K3 to G3K3. K is an agreement threshold. $K = n$ means that sentences with votes $\geq n$ are marked positive.

K	N	Positive	Negative	κ
3	1424	236	1188	0.500

given as:

$$W(\psi) = \sum_{w} (1 + \log(\mathrm{tf}(w))) \cdot \log(N/\mathrm{df}(w)).$$

Here 'tf(w)' denotes the frequency of word w in a given document, 'df(w)' denotes the 'document frequency' of w, or the number of documents which contain an occurrence of w. N represents the total number of documents used to define 'df(w).'

The 'Pos' feature is also among a battery of features used here, intended to record the position or textual order of ψ, given by how many sentences away it occurs from the top of text, starting with 0.

Finally we note that each classifier does a binary (positive/negative) classification, but runs in 'distributional' mode, meaning that it produces the probability of a sentence being positive, i.e., a pick sentence, not the category it thinks it belongs to.

9.3.2 Test Data and Procedure

We created three pools of texts, columns, editorials and news stories from a Japanese financial paper (Nihon-Keizai-Shimbun-Sha, 1995), each consisting of 25 articles, and asked 112 Japanese students to extract 10% worth of sentences from each text which they consider most important in creating a summary; the number of sentences to extract varied from two to four, depending on the length of text. (The corpus is something we dubbed 'JFD-1995' in previous chapters.) We ran experiments on each pool. On average, we had about seven people working on each text. Sentences are marked 'positive' if three or more people nominated them for inclusion in a summary, and 'negative' otherwise. For convenience, let us call the data set for columns G1K3, the editorials G2K3 and the news stories G3K3. Additional details are found in Table 9-1, which also gives the κ statistics, a measure of agreement among humans on their judgments.

Table 9-2 ADTREE on G1K3

	NONBAYS			BAYS		
α	TP	FP	PR	TP	FP	PR
0.10	0.1507	0.0473	0.3958	0.1924	0.0482	0.3958
0.15	0.2271	0.1040	0.3056	0.2271	0.1085	0.2708
0.20	0.2549	0.1612	0.2458	0.2840	0.1598	0.2653
0.25	0.2549	0.2109	0.1938	0.3049	0.2069	0.2090
0.30	0.3361	0.2614	0.2001	0.4090	0.2502	0.2344

Table 9-3 NB on G1K3

	NONBAYS			BAYS		
α	TP	FP	PR	TP	FP	PR
0.10	0.0903	0.0646	0.1875	0.0833	0.0650	0.1875
0.15	0.1597	0.1217	0.1875	0.1993	0.1123	0.2500
0.20	0.2965	0.1616	0.2493	0.2896	0.1644	0.2389
0.25	0.3417	0.2020	0.2333	0.3764	0.2018	0.2424
0.30	0.4042	0.2540	0.2237	0.3382	0.2677	0.1876

9.4 Results and Discussion

Tables 9-2 through 9-13 show how Bayesian summarists perform on G1K3, G2K3, and G3K3. The tables list results in three categories, true positive (TP), false positive (FP) and precision (PR), at compression rates (α) of interest. The figures thereof indicate performance averaged over leave-one-out cross validation folds. TP denotes the ratio of true

Table 9-4 KSTAR on G1K3

	NONBAYS			BAYS		
α	TP	FP	PR	TP	FP	PR
0.10	0.1458	0.0534	0.3542	0.1333	0.0555	0.3333
0.15	0.2757	0.1013	0.3472	0.2722	0.1001	0.3472
0.20	0.3590	0.1534	0.2847	0.3069	0.1548	0.2778
0.25	0.3937	0.1955	0.2597	0.2965	0.2077	0.2111
0.30	0.4368	0.2522	0.2340	0.3604	0.2587	0.2076

Table 9-5 C45 on G1K3

	NONBAYS			BAYS		
α	TP	FP	PR	TP	FP	PR
0.10	0.1389	0.0541	0.3125	0.1958	0.0453	0.4583
0.15	0.2028	0.1070	0.2917	0.2931	0.0967	0.3611
0.20	0.2458	0.1604	0.2514	0.3069	0.1582	0.2757
0.25	0.3167	0.1954	0.2437	0.4056	0.1854	0.2931
0.30	0.3271	0.2570	0.2108	0.4333	0.2497	0.2369

Table 9-6 ADTREE on G2K3

	NONBAYS			BAYS		
α	TP	FP	PR	TP	FP	PR
0.10	0.1660	0.0605	0.3667	0.2287	0.0491	0.5067
0.15	0.3013	0.0917	0.4033	0.3093	0.0896	0.4233
0.20	0.3893	0.1381	0.3627	0.4247	0.1289	0.4060
0.25	0.4620	0.1839	0.3343	0.4840	0.1791	0.3493
0.30	0.4880	0.2362	0.2931	0.5373	0.2278	0.3189

Table 9-7 NB on G2K3

	NONBAYS			BAYS		
α	TP	FP	PR	TP	FP	PR
0.10	0.1487	0.0651	0.3267	0.1573	0.0663	0.2933
0.15	0.2680	0.0994	0.3567	0.2320	0.1080	0.3033
0.20	0.3687	0.1426	0.3393	0.2833	0.1583	0.2707
0.25	0.4467	0.1871	0.3203	0.3033	0.2170	0.2184
0.30	0.5200	0.2337	0.3010	0.3940	0.2593	0.2286

Table 9-8 KSTAR on G2K3

	NONBAYS			BAYS		
α	TP	FP	PR	TP	FP	PR
0.10	0.1407	0.0655	0.3067	0.2153	0.0534	0.4733
0.15	0.2147	0.1101	0.2800	0.2980	0.0956	0.3800
0.20	0.3633	0.1434	0.3367	0.3420	0.1484	0.3140
0.25	0.4233	0.1936	0.3000	0.4373	0.1889	0.3152
0.30	0.4747	0.2404	0.2808	0.4573	0.2436	0.2726

Table 9-9 C45 on G2K3

	NONBAYS			BAYS		
α	TP	FP	PR	TP	FP	PR
0.10	0.1853	0.0575	0.4133	0.2220	0.0501	0.5133
0.15	0.2633	0.0987	0.3667	0.2800	0.0964	0.3800
0.20	0.2947	0.1552	0.2833	0.3840	0.1417	0.3513
0.25	0.3447	0.2079	0.2501	0.4307	0.1938	0.3010
0.30	0.3960	0.2578	0.2338	0.4793	0.2400	0.2831

Table 9-10 ADTREE on G3K3

	NONBAYS			BAYS		
α	TP	FP	PR	TP	FP	PR
0.10	0.3513	0.0186	0.8200	0.3513	0.0186	0.8200
0.15	0.4960	0.0457	0.7133	0.5027	0.0462	0.7067
0.20	0.5640	0.0856	0.5967	0.5973	0.0833	0.6067
0.25	0.6340	0.1404	0.4953	0.6407	0.1421	0.4900
0.30	0.6573	0.1920	0.4240	0.6607	0.1948	0.4153

Table 9-11 NB on G3K3

	NONBAYS			BAYS		
α	TP	FP	PR	TP	FP	PR
0.10	0.3313	0.0228	0.7600	0.3613	0.0164	0.8400
0.15	0.4427	0.0572	0.6267	0.4493	0.0584	0.6200
0.20	0.5493	0.0920	0.5667	0.5227	0.0983	0.5333
0.25	0.6273	0.1431	0.4860	0.5693	0.1590	0.4293
0.30	0.6607	0.1912	0.4227	0.6640	0.1916	0.4227

Table 9-12 KSTAR on G3K3

	NONBAYS			BAYS		
α	TP	FP	PR	TP	FP	PR
0.10	0.3213	0.0240	0.7400	0.3647	0.0165	0.8400
0.15	0.4413	0.0608	0.6067	0.4613	0.0560	0.6400
0.20	0.5460	0.0960	0.5467	0.5560	0.0956	0.5500
0.25	0.5960	0.1560	0.4433	0.6227	0.1503	0.4647
0.30	0.6627	0.1977	0.4113	0.6627	0.1992	0.4080

Table 9-13 C45 on G3K3

	NONBAYS			BAYS		
α	TP	FP	PR	TP	FP	PR
0.10	0.3080	0.0245	0.7400	0.3647	0.0165	0.8400
0.15	0.4660	0.0511	0.6667	0.4827	0.0485	0.6867
0.20	0.5127	0.0975	0.5367	0.5607	0.0877	0.5867
0.25	0.5573	0.1547	0.4420	0.6307	0.1442	0.4840
0.30	0.5707	0.2093	0.3700	0.7140	0.1831	0.4500

positives retrieved over their total, FP the ratio of true negatives retrieved over their total. TP together with FP defines a point on the ROC (Receiver Operating Characteristic), and they are included here to give a general idea of what it looks like. Since however the ROC in general does not allow an easy and accurate comparison among competing systems, we fall back here to working with PR. Thus when we say classifier X outperforms classifier Y, what we mean is that X performs better on PR than Y. PR is defined by the ratio of hits (positive sentences) over the number of sentences retrieved.

In each table, figures to the left of the vertical line are for classifiers without BAYS and those to the right are for classifiers with BAYS. Let us note that we run cross validation on a document basis, which means we test on one document, using the rest for training. After a number of dry runs, we settled for the following settings for λ: $\lambda = 10$ for G1K3 and G2K3, and $\lambda = 20$ for G3K3.

Results for G3K3 (Tables 9-10-9-13), which contains news stories, find that BAYS significantly boosts performance of the classifiers. Its effect is more pronounced when they are run with smaller compression rates; it tends to fade off as the compression rate increases, except for C45 where BAYS remains effective throughout. On G2K3 (editorials; Tables 9-6-9-9), BAYS is generally effective except for Naive Bayes (NB), which registers no improvement at all. The Bayesian Kstar comfortably outperforms its non-Bayesian version at 0.10 and 0.15, but loses to the latter with larger compression rates. However, in G1K3 (columns; Tables 9-2-9-5), Bayesian classifiers are doing just about as good as their non-Bayesian counterparts except for the Bayesian C45, which enjoys a huge increase in performance.

Table 9-14 Performance in precision of LEAD on G1K3, G2K3 and G3K3.

α	PR (G1K3)	PR (G2K3)	PR (G3K3)
0.10	0.3000	0.4533	0.8400
0.15	0.2800	0.3733	0.6200
0.20	0.2413	0.2920	0.4867
0.25	0.2340	0.2497	0.4233
0.30	0.2024	0.2205	0.3780

Now why is BAYS not as effective on G1K3 as on G2K3 and G3K3? Some insight into the puzzle comes from the vote distributions in Figure 9-1. Compared to G3K3 and G2K3, G1K3 has the DOV somewhat ragged with sudden dips and rises. We suspect this is what makes the focused learning with BAYS less effective on G1K3. Also worthy of mention is that C45 responds remarkably well to the Dirichlet resampling, providing a huge boost in performance across the three domains.

Finally let us note that Bayesian summarists generally compare favorably to the lead based summarizer (LEAD); C45 outdoes the latter by a large margin, and Kstar to a lesser extent. However, how much better BAYS's perform than LEAD apparently depends on the domain they work on; we see them comfortably beat LEAD on G3K3 and G2K3, and moderately so on G1K3. An obvious loser here is NB or Naive Bayes, whose performance, BAYS-enabled or not, lags far behind not only those of competing classifiers but also of LEAD. Why it fails is something we do not have a ready answer to, but apparently NB does not work well with the biased statistics that BAYS gives rise to. It is interesting to note how its performance drops when coupled with the BAYS; it fares better without the enhancement most of the time.

9.5 Concluding Remarks

We have discussed a Bayesian approach to text summarization and gains it delivers to a usual variety of summarization models based on pattern classification. We also identified some of the properties of domains on which the Bayesian model tends to fail. A possible

future direction would include exploring the use of gradient algorithms for automatically finding optimal values of λ (MacKay & Peto, 1994).

Chapter 10 Conclusions

10.1 Summing up

Before we conclude, let us run through some of the major points of the thesis. The discourse part of the thesis focused on formulating learning based approaches towards recovering a linguistic model of discourse for a possible use with text summarization. In chapter 2, we constructed a DT based generic discourse parser which one could adapt for any language given a training corpus. Among features used for a DT parser, we found that distance-, location- and length-related features contribute most to performance of the parser.

Chapter 3 addresses the issue of annotating texts with concepts from discourse theories such as Rhetorical Structure Theory (RST) (Mann & Thompson, 1987a). Initially we worked with RST, but it soon became clear that human annotators disagree widely on rhetorical relations they provide. This made us turn to an alternative scheme which we call 'ITDR' (Ichikawa, 1990), where rhetorical relations are explicitly tied with discourse connectives, therefore easier to mark a text for. All this eventually resulted in improved agreement among humans.

Chapter 6 marks a major shift in the sort of approach we take to summarization; we found out from previous chapters that judgments on discourse relations are not reliably elicited from humans, which could compromise the integrity of a linguistic corpus they are part of. Chapter 6 responds by exploring a unsupervised approach to summarization, which we call the diversity based summarization or DBS. DBS proved particularly effective in retrieving marginally relevant documents.

Chapter 7 turns an eye on an question of how well DBS serves as a model of humans extracted summaries. Curiously enough, we found that DBS generally outperforms DT. It is curious because the DT has to its avail information on various aspects of data, such as location, length, etc. A closer look at the results reveals, however, that each of the data sets we are working with has the distribution of votes (DOV) with a shape specific to it; votes are more widely spread across the whole text in columns and editorials than in news

articles. We argued that they would provide some useful clues to understanding why the summarizers are behaving as they do.

Picking up on this issue, the paper explores a Bayesian approach in chapter 9, where we aim at explicitly encoding and exploiting DOVs through a statistical modeling. It turns out that the Bayesian approach, when coupled with a usual variety of pattern classifiers, significantly improves performance of the classifiers.

10.2 Gold standard for summarization?

One of the long-standing issues in summarization is an apparent variability of human judgments; people disagree on what they like to see in a summary as often as, possibly more often than they agree, which is amply demonstrated in the literature and also by the present study (chapter 8–9).

A unique feature of the Bayesian approach espoused in chapter 9 lies in the view it takes that summarization is a particular form of collaborative filtering (CF), wherein we regard a summary as a set of sentences favored by a particular user or a group of users just like any other things people may have particular preference for, such as CDs, books, paintings, emails, news articles, etc. Importantly, under CF, we would not be asking, what is the 'correct' or gold standard summary for document X? – the question that consumed much of the past research on summarization.

Indeed the fact that there could be as many summaries as angles to look at the text from may favor a CF view of summary: that what constitutes a good summary may vary and depend on a particular set of profile data we are working with, which may come from a single user or a group of users with various interests and concerns. There are some recent efforts along the similar lines. One notable is the Pyramid scheme (Nenkova & Passonneau, 2004) where one does not declare a particular human summary a absolute reference to compare summaries against, but rather makes every one of multiple human summaries at hand bear on evaluation; Rouge (Lin & Hovy, 2003) represents another such effort. The Bayesian summarist (chapter 9) represents yet another wherein one strives to find a summary most typical of those created by humans.

One could talk about a particular translation being right or wrong, but probably not with a summary, as it involves a subjective interpretation (and prioritizing) of events described in the text, and could come out quite unlike any of the corresponding reference

(or ‘correct’) summaries we happen to have. Perhaps a lesson to take home is, ‘no standard’ is a gold standard for summarization.

APPENDIX A.

A.1 Split Criterion

The *gain* criterion measures the effectiveness of partitioning a data set T with respect to a test X, and is defined as follows.

$$gain(X) = info(T) - info_X(T)$$

Define $info(T)$ to be an entropy of T, that is, the average amount of information generated by T. Then we have:

$$info(T) = -\sum_{j=1}^{k} \frac{freq(C_j, T)}{\mid T \mid} \times \log_2 \frac{freq(C_j, T)}{\mid T \mid}$$

$freq(C, T)$ is the number of cases from a class C divided by the sum of cases in T. Now $info_X(T)$ is the average amount of information generated by partitioning T with respect to a test X. That is,

$$info_X(T) = \sum_{i=1}^{n} \frac{\mid T_i \mid}{\mid T \mid} \times info(T_i)$$

Thus a good classifier would give a small value for $info_X(T)$ and a large value for $info_X(T)$.

The *gain ratio* criterion is a modification to the *gain* criterion. It has the effect of making a splitting of a data set less intense.

$$gain\ ratio(X) = gain(X)/split\ info(X)$$

where:

$$split\ info(X) = -\sum_{i=1}^{n} \frac{\mid T_i \mid}{\mid T \mid} \times \log_2 \frac{\mid T_i \mid}{\mid T \mid}$$

The ratio decreases with an increase in the number of splits.

Table A-1 Assignments Matrix

	1	2	$\cdots$	j	$\cdots$	m	
1	n_{11}	n_{11}	$\cdots$	n_{1j}	$\cdots$	n_{1m}	S_1
2	n_{21}	n_{22}	$\cdots$	n_{2j}	$\cdots$	n_{2m}	S_2
$\vdots$				$\vdots$			$\vdots$
i	n_{i1}	n_{i2}	$\cdots$	n_{ij}	$\cdots$	n_{im}	S_i
$\vdots$				$\vdots$			$\vdots$
N	n_{N1}	n_{N2}	$\cdots$	n_{Nj}	$\cdots$	n_{Nm}	S_N
	C_1	C_2	$\cdots$	C_j	$\cdots$	C_m	

A.2 The Kappa Statistic

The kappa coefficient (K) of agreement measures the ratio of observed agreements to possible agreements among a set of raters on category judgments, correcting for chance agreement:

$$K = \frac{P(A) - P(E)}{1 - P(E)} \tag{A-1}$$

where $P(A)$ is the proportion of the times that raters agree and $P(E)$ is the proportion of the times that we would expect them to agree by chance. $K = 1$ if there is complete agreement among the raters. $K = 0$ if there is no agreement other than that which is expected by chance. Consider a set of k raters and a group of N objects, each of which is to be assigned to one of m categories. Each of the raters assigns each object to a category. We represent the assignments data as an $N \times m$ matrix (Table A-1), where the value (n_{ij}) at each $\text{cell}_{i,j}$ ($0 < i \leq N$, $0 < j \leq m$) denotes the number of raters assigning the ith object to the jth category. Let C_j be the total number of times that objects are assigned to the jth category, i.e., $C_j = \sum_{i=1}^{N} n_{ij}$. We give S_i by Def. A-2.

$$S_i = \frac{\sum_{j=1}^{m} \binom{n_{ij}}{2}}{\binom{k}{2}} = \frac{1}{k(k-1)} \sum_{j=1}^{m} n_{ij}(n_{ij} - 1) \tag{A-2}$$

Table A-2 Probabilistic Classification with DT. $\vec{u}$ is a vector representation of sentence u. α is a smoothing function. $t(\vec{u})$ is some leaf node assigned to $\vec{u}$ by DT.

$$P(\mathit{Select} \mid \vec{u}, \mathrm{DT}) = \alpha\left(\frac{\text{the number of ``Select'' sentences at } t(\vec{u})}{\text{the total number of sentences at } t(\vec{u})}\right)$$

S_i measures the proportion of pairwise agreements among the raters on category assignments for a particular object i. Agreement frequencies n_{ij} on each row must sum to k, the total number of raters. Note that $0 < S_i \leq 1$. $S_i = 1$ when there is total agreement among the raters for a given category j on the ith row. $P(A)$, or the proportion of the times that the raters agree, is given as the average of S_i across all objects (Def. A-3).

$$P(A) = \frac{1}{N}\sum_{i=1}^{N} S_i \tag{A-3}$$

The probability that a category is chosen at random is estimated as $p_j = C_j/Nk$. Then, the probability that any two raters agree on the jth category by chance would be p_j^2. $P(E)$ is defined as the sum of chance agreement for each category (Def A-4), representing the overall rate of agreement by chance.

$$P(E) = \sum_{j=1}^{m} p_j^2 \tag{A-4}$$

The values of $P(A)$ and $P(E)$ are then combined to give the kappa coefficient K.

A.3 Supervised Ranking with Probabilistic DT

One technical problem associated with the use of a decision tree as a summarizer is that it is not able to rank sentences, which it must be able do, to allow for the generation of a variable-length summary. In response to the problem, we explore the use of a probabilistic decision tree as a ranking model. First, let us review some general features of probabilistic decision tree (ProbDT, henceforth) (Yamanishi, 1997; Rissanen, 1997).

ProbDT works like a usual decision tree except that rather than assigning each instance to a single class, it distributes each instance among classes. For each instance x_i, the

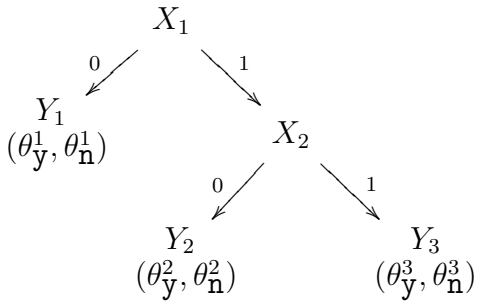

Figure A-1 Probabilistic Decision Tree (Same as Figure 4-1).

strength of its membership to each of the classes is determined by $P(c_k \mid x_i)$ for each class c_k.

Consider a binary decision tree in Fig A-1. Let X_1 and X_2 represent non-terminal nodes, and Y_1 and Y_2 leaf nodes. '1' and '0' on arcs denote values of some attribute at X_1 and X_2. $\theta^i_{\mathtt{y}}$ and $\theta^i_{\mathtt{n}}$ represent the probability that a given instance assigned to the node i is labeled as `yes` and `no`, respectively. Abusing the terms slightly, let us assume that X_1 and X_2 represent splitting attributes as well at respective nodes. Then the probability that a given instance with $X_1 = 1$ and $X_2 = 0$ is labeled as `yes` (`no`) is $\theta^2_{\mathtt{y}}$ ($\theta^2_{\mathtt{n}}$). Note that $\sum_c \theta^j_c = 1$ for a given node j.

Now to rank sentences with ProbDT simply involves finding the probability that each sentence is assigned to a particular class designating sentences worthy of inclusion in a summary (call it 'Select' class) and ranking them accordingly. (Hereafter and throughout the rest of the paper, we say that a sentence is *wis* if it is worthy of inclusion in a summary: thus a *wis* sentence is a sentence worthy of inclusion in a summary.) The probability that a sentence u is labeled as *wis* is expressed as in Table A-2, where $\vec{u}$ is a vector representation of u, consisting of a set of values for features of u; α is a smoothing function, e.g., Laplace's law; $t(\vec{u})$ is some leaf node assigned to $\vec{u}$; and DT represents some decision tree used to classify $\vec{u}$.

A.3.1 Features

The following lists a set of features used for encoding a sentence in ProbDT.[1] Most of them are either length- or location-related features.[2]

`<LocSen>` The location of a sentence X defined by:

$$\frac{\#S(X)-1}{\#S(Last\_Sentence)}$$

'$\#S(X)$' denotes an ordinal number indicating the position of X in a text, i.e. $\#S(k\text{th\_sentence}) = k$. 'Last_Sentence' refers to the last sentence in a text. `LocSen` takes values between 0 and $\frac{N-1}{N}$. N is the number of sentences in the text.

`<LocPar>` The location of a paragraph in which a sentence X occurs given by:

$$\frac{\#Par(X)-1}{\#Last\_Paragraph}$$

'$\#Par(X)$' denotes an ordinal number indicating the position of a paragraph containing X. '#Last_Paragraph' is the position of the last paragraph in a text, represented by the ordinal number.

`<LocWithinPar>` The location of a sentence X within a paragraph in which it appears.

$$\frac{\#S(X)-\#S(Par\_Init\_Sen)}{Length(Par(X))}$$

'Par_Init_Sen' refers to the initial sentence of a paragraph in which X occurs, 'Length(Par(X))' denotes the number of sentences that occur in that paragraph. `LocWithinPar` takes continuous values ranging from 0 to $\frac{l-1}{l}$, where l is the length of a paragraph: a paragraph initial sentence would have 0 and a paragraph final sentence $\frac{l-1}{l}$.

`<LenText>` The text length in Japanese character i.e. *kana*, *kanji*.

`<LenSen>` The sentence length in *kana*/*kanji*.

[1] The features are same as those given in chapter 7, repeated here for the convenience's sake

[2] Note that one may want to add tfidf to a set of features for a decision tree or, for that matter, to use features other than tfidf for representing sentences in clustering. The idea is worthy of consideration, but not pursued here.

Table A-3 Linguistic cues

code	category
1	non-past
2	past `/-ta/`
3	copula `/-da/`
4	noun
5	symbols, e.g., parentheses
6	sentence-ending particles, e.g., `/-ka/`
0	none of the above

Some work in Japanese linguistics found that a particular grammatical class a sentence final element belongs to could serve as a cue to identifying summary sentences. These include categories like PAST/NON-PAST, INTERROGATIVE, and NOUN and QUESTION-MARKER. Along with Ichikawa (1990), we identified a set of sentence-ending cues and marked a sentence as to whether it contains a cue from the set.[3] Included in the set are inflectional classes PAST/NON-PAST (for the verb and verbal adjective), COPULA, and NOUN, parentheses, and QUESTION-MARKER `-ka`. We use the following attribute to encode a sentence-ending form.

`<EndCue>` The feature encodes one of sentence-ending forms described above. It is a discrete valued feature. The value ranges from 0 to 6. (See Table A-3 for details.)

Finally, one of two class labels, 'Select' and 'Don't Select', is assigned to a sentence, depending on whether it is *wis* or not. The 'Select' label is for *wis* sentences, and the 'Don't Select' label for non-*wis* sentences.

[3]Word tokens are extracted by using CHASEN, a Japanese morphological analyzer which is reported to achieve the accuracy rate of over 98% (Matsumoto et al., 1999).

A.4 MAP

Let θ, $\mathcal{U}$, $\mathcal{V}$ and $\lambda\mathbf{u}$ be as they are given above, and $E[\,\theta\,|\mathcal{U}\,]$ be an expectation of θ given $\mathcal{U}$. Let $\mathbf{u} = (u_1, \ldots, u_n)$, with $u_i = 1/n$. Note that

$$\begin{aligned} & E[\,\theta\,|\mathcal{U}\,] \\ &= \int \text{Dirichlet}^n(\theta\,|\lambda\mathbf{u})\,\theta\,d^n\,\theta \\ &= \left(\int u_1\,p_1\,dp_1, \ldots, \int u_n\,p_n\,dp_n\right) \\ &= \mathbf{u}. \end{aligned}$$

where $\int p_i dp_i = 1$. Therefore we have

$$\begin{aligned} & E[\,\theta\,|\mathcal{V},\mathcal{U}\,] \\ &= \int \text{Dirichlet}^n(\theta\,|\mathcal{V} + \lambda\mathbf{u})\,\theta\,d^n\,\theta \\ &= \left(\ldots, \frac{v_i + \lambda/n}{\sum_j v_j + \lambda}, \ldots\right) = \hat{\theta}. \end{aligned}$$

where $\hat{\theta} = (\hat{\theta}_1, \ldots, \hat{\theta}_n)$. Thus $\hat{\theta}_i = \dfrac{v_i + \lambda/n}{\sum_j v_j + \lambda}$.

References

Abney, S. (2002). Bootstrapping. In *Proceedings of the 40th annual meeting of the association for computational linguistics.* ACL.

Aone, C., Gorlinsky, J., Larsen, B., & Okurowski, M. E. (1999). A trainable summarizer with knowledge acquired from robust nlp techniques. In I. Main & M. T. Maybury (Eds.), *Advances in automatic text summarization* (p. 71-80). The MIT Press.

Barzilay, R., & Elhadad, M. (1999). Using lexical chains for text summarization. In I. Main & M. T. Maybury (Eds.), *Advances in automatic text summarization* (p. 111-121). The MIT Press.

Berger, A., & Mittal, V. O. (2000). Query-relevant summarization using FAQs. In *Proceedings of the 38th annual meeting of the association for computational linguistics* (p. 294-301). Hong Kong.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference of computational learning theory (colt98)* (p. 92-100). Madison, Wisconsin.

Boguraev, B., & Kennedy, C. (1999). Salience-based content characterization of text documents. In I. Main & M. T. Maybury (Eds.), *Advances in automatic text summarization* (p. 99-110). The MIT Press.

Bradley, P. S., & Fayyad, U. M. (1998). Refining initial points for k-means clustering. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML98)* (p. 91-99). Morgan Kaufmann.

Carbonell, J., & Goldstein, J. (1998). The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21th annual international acm/sigir conference on research and development in information retrieval.* Melbourne, Australia.

Carletta, J., Isard, A., Isard, S., Kowtko, J. C., Doherty-Sneddon, G., & Anderson, A. H. (1997). The Reliability of a Dialogue Structure Coding Scheme. *Computational Linguistics*, *23*(1), 13-31.

Choi, W. S., Cho, J.-M., & Seo, J. (1999). Analysis System of Speech Acts and Discourse Structures Using Maximum Entropy Model. In *Proceedings of the 37th annual conference of the association for computational linguistics* (p. 230-237). College Park, MD: The Association for Computational Linguistics.

Chuang, W., & Yang, J. (2000). Text summarization by sentence segment extraction using machine learning algorithms. In T. Terano, H. Liu, & A. L. P. Chen (Eds.), *Knowledge discovery and data mining* (p. 454-457). Springer.

Cleary, J. G., & Trigg, L. E. (1995). K: An instance-based learner using an entropic distance measure. In *Proceedings of the 12th international conference on machine learning* (p. 108-114).

Cohen, P. R. (1995). *Empirical methods in artificial intelligence.* The MIT Press.

Collins, M., & Singer, Y. (1999). Unsupervisd models for named entity classification. In *Proceedings of joint sigdat conference on empirical methods in natural language processing and very large corpora (emnlp/vlc99)* (p. 100-110). Maryland.

Collins, M. J. (1996). A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of the 34th annual meeting of the association for computational linguistics* (p. 184-191). California, USA.: Association for Computational Linguistics.

Congdon, P. (2003). *Bayesian statistical modelling.* John Wiley and Sons.

Cowans, P. J. (2004). Information Retrieval Using Hierarchical Dirichlet Processes. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (p. 564-565). Sheffield, U.K.: ACM.

Dagan, I., & Engelson, S. (1995). Committee-based sampling for training probabilistic classifiers. In *Proceedings of international conference on machine learning* (p. 150-157).

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, *40*(2), 139-158.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification* (second ed.). John Wiley & Sons, Inc.

Edmundson, H. P. (1969). New Method in Automatic Abstracting. *Journal of the ACM*, *16*(2), 264-285.

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap.* New York: Chapman and Hall.

Engelson, S. P., & Dagan, I. (1996). Minimizing manual annotation cost in supervised training from corpora. In *Proceedings of the 34th annual meeting of the association for computational linguistics* (p. 319-326). University of California,Santa Cruz.

Fox, B. A. (1987). *Discourse structure and anaphora.* Cambridge, UK: Cambridge University Press.

Freund, Y., & Mason, L. (1999). The alternating decision tree learning algorithm,. In *Proc. 16th international conf. on machine learning* (pp. 124–133). Morgan Kaufmann, San Francisco, CA.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proceedings of the thirteenth international conference on machine learning.*

Gong, Y., & Liu, X. (2001). Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th annual international acm/sigir conference on research and development in information retrieval.* New Orleans: ACM-Press.

Grosz, B., & Sidner, C. (1986). Attention, Intentions and the Structure of Discourse. *Computational Linguistics*, *12(3)*, 175-204.

Halliday, M., & Hasan, R. (1990). *Cohesion in english.* New York: Longman.

Haruno, M. (1997). *Kettei-gi o mochiita nihongo kakariuke kaiseki (A Japanese dependency parser using a decision tree).* (Submitted for publication, ATR, Seika)

Hearst, M. A. (1994). Multi-Paragraph Segmentation of Expository Text. In *Proceedings of the 32nd annual meeting of the association for computational linguistics* (p. 9-16). New Mexico, USA.

Ichikawa, T. (1990). *Bunshôron-gaisetsu.* Tokyo: Kyôiku-Shuppan.

Jing, H., Barzilay, R., McKeown, K., & Elhadad, M. (1998). Summarization evaluation methods: Experiments and analysis. In *Aaai symposium on intelligent summarization.* Stanford Univesisty, CA.

Kalton, A., Langely, P., Wagstaff, K., & Yoo, J. (2001). Generalized clustering, supervised learning, and data assignment. In *Proceedings of the seventh international conference on knowledge discovery and data mining (kdd2001).* San Francisco: ACM.

Katz, S. M. (1996). Distribution of content words and phrases in text and language modelling. *Natural Language Engineering*, *2*(1), 15-59.

Krippendorff, K. (1980). *Content Analysis: An Introduction to Its Methodology* (Vol. 5). The Sage Publications, Inc.

Kupiec, J., Pedersen, J., & Chen, F. (1995). A trainable document summarizer. In *Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Developmnet in Information Retrieval* (p. 68-73). Seattle.

Kurohashi, S., & Nagao, M. (1994). Automatic Detection od Discourse Structure by Checking Surface Information in Sentences. In *Proceedings of the 15th intewrnational conference on computational linguistics* (p. 1123-1127). Kyoto,Japan.

Levinson, S. C. (1994). *Pragmatics.* Cambridge University Press.

Li, H. (1998). *A probabilistic approach to lexical semantic knowledge acquistion and structural disambiguation.* Unpublished doctoral dissertation, University of Tokyo, Tokyo.

Lin, C.-Y., & Hovy, E. (2003). Automatic evaluation of summaries using n-gram co-occurance statistics. In *Proceedings of hlt-naacl* (p. 71-78). Edmonton: ACL.

Litman, D., & Passonneau, R. J. (1995). Combining multiple knowledge sources for discourse segmentation. In *Proceedings of the 33rd annual meeting of the association for computational linguistics* (p. 108-115). The Association for Computational Linguistics. (Cambridge, MASS. USA)

Luhn, H. P. (1958). The automatic creation of literary abstracts. In I. Mani & M. T. Maybury (Eds.), *Advances in automatic text summarization,1999* (p. 15-21). The MIT Press. (Reprint)

MacKay, D. J. C., & Peto, L. C. B. (1994). A hierarchical dirichlet language model. *Natural Language Engineering.*

Magerman, D. M. (1995). Statistical Decision-Tree Models for Parsing. In *Proceedings of the 33rd annual meeting of the association for computational linguistics* (p. 276-283). Camridge, MASS. USA.: Association for Computational Linguistics.

Mani, I., House, D., Klein, G., Hirshman, L., Obust, L., Firmin, T., Chrzanowski, M., & Sundheim, B. (1998). *The TIPSTER SUMMAC Text Summarization Evalutation Final Report* (Tech. Rep.). Virginia, USA: MITRE.

Mann, W. C., & Thompson, S. A. (1987a). Rhetorical structure theory. In L. Polyani (Ed.), *The structure of discourse.* Norwood, NJ: Ablex Publishing Corp.

Mann, W. C., & Thompson, S. A. (1987b). *Rhetorical Structure Theory : A Theory of Text Organization* (Technical Report ISI/RS 87-190). ISI.

Marcu, D. (1997). The Rhetorical Parsing of Natural Language Texts. In *Proceedings of the 35th annual meetings of the association for computational linguistics and the 8th european chapter of the association for computational linguistics* (p. 96-102). Madrid, Spain.

Marcu, D. (1998). To build text summaries of high quality, nuclearity is not sufficient. In *The working notes of the aaai-98 spring symposium on intelligent text summarization* (p. 1-8). Stanford: AAAI.

Marcu, D. (1999a). A decision-based approach to rhetorical parsing. In *Proceedings of the 37th annual conference of the association for computational linguistics* (p. 365-372). College Park, MD: The Association for Computational Linguistics.

Marcu, D. (1999b). Discourse trees are good indicators of importance in text. In I. Mani & M. T. Maybury (Eds.), *Advances in automatic text summarization* (p. 123-136). The MIT Press.

Marcu, D. (1999c). The automated construction of large-scale corpora for summarization research. In *Proceedings of the 22nd International ACM/SIGIR Conference on Research and Development in Informational Retrieval* (p. 137-144). Berkeley.

Marcu, D., & Echihabi, A. (2002). An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th annual meeting of the association for computational linguistics.*

Marcus, M., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, *19*(2), 313-330.

MathSoft. (1998). *S-PLUS 5 for UNIX Guide to Statistics.*

Matsumoto, Y., Kitauchi, A., Yamashita, T., & Hirano, Y. (1999). *Japanese morphological analysis system chasen version 2.0 manual* (Tech. Rep.). Ikoma: NAIST. (NAIST-IS-TR99008)

Miike, S., Itoh, E., Ono, K., & Sumita, K. (1994). A Full-text Retrieval System with a Dynamic Abstract Generation Function. *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, 152-159. (Dublin, Ireland)

Morris, A. H., Kasper, G. M., & Adams, D. A. (1999). The effects and limitations of automated text condensing on reading comprehension performance. In I. Main & M. T. Maybury (Eds.), *Advances in automatic text summarization* (p. 305-323). The MIT Press.

Moser, M., & Moore, J. D. (1993). Investigating discourse relations. In B. Boguraev & J. Pustejovsky (Eds.), *Proceedings of workshop "acquisition of lexical knowledge from text" sponsored by special interest group on the lexicon of the association for computational linguistics* (p. 94-97). Ohio State University.

Nenkova, A., & Passonneau, R. (2004). Evaluation Content Selection in Summarization: The Pyramid Method. In *Proceedings of the human language technology conference of the north american chapter of the association for computational linguistics: Hlt-naacl 2004* (p. 145-152). Boston.

Nichi-Gai Associates. (1995). *CD-mainichi shimbun '94: Data shu.* CD-ROM.

Nihon-Keizai-Shimbun-Sha. (1995). *Nihon keizai shimbun 95 nen cd-rom ban.* CD-ROM. (Tokyo, Nihon Keizai Shimbun, Inc.)

Nihon-Keizai-Shimbun-Sha. (1997). *Nihon Keizai Shimbun 97 nen CD-ROM ban.* CD-ROM. (Tokyo, Nihon Keizai Shimbun, Inc.)

Nomoto, T., & Matsumoto, Y. (1997). Data Reliability and Its Effects on Automatic Abstracting. In *Proceedings of the fifth workshop on very large corpora.* Beijing/Hong Kong, China.

Nomoto, T., & Matsumoto, Y. (1998). Discourse parsing: A decision tree approach. In E. Charniak (Ed.), *Proceedings of the sixth workshop on very large corpora* (p. 216-224). Montréal: ACL-SIGDAT.

Nomoto, T., & Matsumoto, Y. (2000). Comparing the minimum description length principle and boosting in the automatic analysis of discourse. In *Proceedings of the Seventeenth International Conference on Machine Learning* (p. 687-694). Stanford University: Morgan Kaufmann.

Nomoto, T., & Matsumoto, Y. (2001a). *The diversity based approach to open-domain text summarization.* (Unpublished Manuscript)

Nomoto, T., & Matsumoto, Y. (2001b). A new approach to unsupervised text summarization. In *Proceedings of the 24th International ACM/SIGIR Conference on Research and Development in Informational Retrieval.* New Orleans: ACM.

Pelleg, D., & Moore, A. (2000). X-means: Extending K-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML2000)* (p. 727-734). Stanford University: Morgan Kaufmann.

Polanyi, L., & Scha, R. (1984). A syntactic approach to discourse semantics. In *Proceedings of the 10th international conference on computaional linguistics and the 22nd annual meetings of the association for computational linguistics* (p. 413-420). Stanford.

Pollock, J. J., & Zamora, A. (1999). Automatic abstracting research at chemical abstracts service. In I. Mani & M. T. Maybury (Eds.), *Advances in automatic text summarization* (p. 43-49). The MIT Press. (Reprint)

Quinlan, J. R. (1993). *C4.5: Programs for machine learning.* Morgan Kaufmann.

Quinlan, J. R., & Rivest, R. L. (1989). Inferring decision trees using the minimum description length principle. *Information and Computation*, *80*, 227-248.

Radev, D. R., Jing, H., & Budzikowska, M. (2000). Centroid-based summarization of multiple documents: sentence extration, utility-based evalutation, and user studies. In *Proceedings of the acl/naal workshop on summarization.* Seattle, WA.

Rino, L. H. M., & Scott, D. R. (1996). *A discourse model for gist preservation* (Tech. Rep. No. ITRI-96-12). ITRI, University of Brighton.

Rissanen, J. (1997). Stochastic complexity in learning. *Journal of Computer and System Sciences*, *55*, 89-95.

Sakuma, M. (Ed.). (1987). *Bunshô-kôzô to yôyaku-bun no shosô.* Tokyo: Kuroshio.

Sakurai, H., & Hisamitsu, T. (1997). Keitaiso Puroguramu ANIMA no Sekkei to Jissoo. In *Jyoohoo shori gakkai zenki zenkoku taikai kooen ronbun shuu* (Vol. 2, p. 57-56). Information Processing Society of Japan.

Salton, G., Singhal, A., Mitra, M., & Buckley, C. (1999). Automatic text structuring and summarization. In I. Mani & M. T. Maybury (Eds.), *Advances in automatic text summarization* (p. 342-355). The MIT Press. (Reprint)

Samuel, K., Carberry, S., & Vijay-Shanker, K. (1998). Diaglogue act tagging with transformation-based learning. In *Proceedings of the 36th annual meeting of the association of computational linguistics and the 17th international conference on computational linguistics* (p. 1150-1156). Montreal, Canada.

Siegel, S., & Castellan, N. J. (1988). *Nonparametric statistics for the behavioral sciences* (Second ed.). McGraw-Hill.

Sparck Jones, K., & Gallier, J. R. (1995). *Evaluating natural language processing systems: an analysis and review.* Springer.

Van Dijk, T. A. (1988). *News as discourse.* Hillsdale, NJ: Erlbaum.

Wang, H., & Yu, P. (2001). SSDT: A scalable subspace-splitting classifier for biased data. In *Proceedings of 2001 ieee international conference on data mining* (p. 542-549). San Jose: IEEE Computer Society.

Witten, I. H., & Frank, E. (2000). *Data mining: Practical machine learning tools and techniques with java implementations.* Morgan Kaufmann.

Yamanishi, K. (1997). Data compression and learning. *Journal of Japanese Society for Artificial Intelligence*, *12*(2), 204-215. (in Japanese)

Yu, K., Tresp, V., & Yu, S. (2004). A Nonparametric Hierarchical Bayesian Framework for Information Filtering. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (p. 353-360). Sheffield, U.K.: ACM.

Zechner, K. (1996). Fast generation of abstracts from general domain text corpora by extracting relevant sentences. In *Proceedings of the 16th international conference on computational linguistics* (p. 986-989). Copenhagen.

PUBLICATIONS

Reviewed Journal Papers

1. Nomoto, T. and Matsumoto, Y. (2003). The Diversity Based Approach to Open-Domain Text Summarization. Information Processing & Management, vol. 39, 363-389.

2. Nomoto, T. and Matsumoto, Y. (2004). Hikensha hantei no yure to youyaku moderu (Exploiting Human Judgments for Automatic Text Summarizatioin: An Empirical Comparison), *Jyoho Shori Gakkai Ronbunshi* (IPSJ Journal), Vol. 45, No. 3, 794-808.

Reviewed Conference Papers

1. Nomoto, T. and Nitta, Y. (1991). Extended Reference and Discourse Representation. Proceedings of the 2nd Japan-Australia Joint Symposium on Natural Language Processing, 261-267. Iizuka, Japan.

2. Nomoto, T. and Nitta, Y. (1993). Resolving Zero Anaphora in Japanese. ACL Proceedings of Sixth European Conference, 315-321. Utrecht, The Netherlands.

3. Nomoto, T. and Nitta, Y. (1994a). A Grammatico-Statistical Approach to Discourse Partitioning. Proceedings of the 15th International Conference on Computational Linguistics, 1145-1149. Kyoto, Japan.

4. Nomoto, T. and Nitta, Y. (1994b). Structuring the Raw Discourse. International Conference on New Methods in Language Processing (NeMLaP), 162-167. Manchester, UK: The University of Manchester Institute of Science and Technology (UMIST).

5. Nomoto, T. (1995). Effects of Grammatical Annotation on the Topic Identification Task. Proceedings of the International Conference on Recent Advances in Natural Language Processing. Tzigov Chark, Bulgaria.

6. Nomoto, T. and Matsumoto, Y. (1996). Exploiting Text Structure for Topic Identification. In E. Ejerhed I. Dagan (Eds.), Proceedings of the Fourth Workshop on Very Large Corpora, 101-112. Copenhagen, Denmark: ACL SIGDAT.

7. Yoshida, S., Nomoto, T., and Takahashi, H. (1997). Investigating Morpho-Syntactic Factors in Human Anaphora Resolution: A Case of Japanese Anaphora. Proceedings of the First International Conference on Cognitive Science,45-50, Seoul, Korea: The Korean Society for Cognitive Science.

8. Nomoto, T. and Matsumoto, Y. (1997). Data Reliability and Its Effects on Automatic Abstracting. Proceedings of the Fifth Workshop on Very Large Corpora. Beijing/Hong Kong, China: ACL SIGDAT.

9. Nomoto, T. and Matsumoto, Y. (1998). Discourse Parsing: A Decision Tree Approach. In E. Charniak (Ed.), The Six Workshop on Very Large Corpora, 216-224. Monteal, Quebec, Cananda: COLING-ACL98, ACl-SIGDAT.

10. Nomoto, T. and Matsumoto, Y. (1999). Learning Discourse Relations with Active Data Selection. In P. Fung and J. Zhou (eds.), Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 158-167, University of Maryland, MD, USA.

11. Nomoto, T. and Matsumoto, Y. (2000). Comparing the Minimum Description Length Principle and Boosting in the Automatic Analysis of Discourse. Proceedings of the Seventeenth International Conference on Machine Learning (ICML00), 687-694, Stanford University.

12. Miyata, T., Matsumoto, Y., Nomoto, T., and Tokunaga,T. (2000). Document Analysis and Summarization Support Environment. presented at Demo Session, The 38th Annual Meeting of Association of Computational Linguistics, Hong Kong.

13. Nomoto, T and Matsumoto, Y. (2001) A New Approach to Unsupervised Text Summarization. Proceedings of the 24th International ACM/SIGIR Conference on Research and Development in Information Retrieval, New Orleans.

14. Nomoto, T and Matsumoto, Y. (2001) An Experimental Comparison of Supervised and Unsupervised Approaches to Text Summarization. Proceedings of the 2001 IEEE International Conference on Data Mining, San Jose.

15. Nomoto, T. and Matsumoto, Y. (2002) Supervised Ranking in Open-Domain Text Summarization. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, ACL.

16. Nomoto, T and Matsumoto, Y.(2002) Modeling (In)Variability of Human Judgments for Text Summarization. Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Tampere, ACM.

17. Nomoto, T. (2003) Predictive Models of Performance in Multi-Engine Machine Translation. Proceedings of Machine Translation Summit IX, New Orleans. IAMT.

18. Nomoto, T. (2004) Multi-Engine Machine Translation with Voted Language Model. Proceedings of the 42th Annual Meeting of the Association for Computational Linguistics, Barcelona, ACL.

Non-Reviewed Papers

1. Nomoto, T. (1995). Kakuritsu moderu ni yoru shudai no jidou chuushutsu (A Probabilistic Approach to Topic Identification), *Shingaku Gihou* (IEICE Tech. Rep.), Vol. 95, No. 168/169, 1-6.

2. Yoshida, S., Nomoto, T., and Takahashi, H. (1995). Zero Daimeishi no shouou ni oyobosu kakujyoshi no eikyou ni tsuite: wa to ga no baai (On how case markers affect the resolution of zero pronouns: a case of *wa* and *ga*), *Nihon Ninchi Kagakukai 12 Kai Taikai Ronbunshuu* (Proc. of the 12th meeting of JCSS), 66-67.

3. Nomoto, T. and Matsumoto, Y. (1996). Tekisuto kouzou wo riyoushita shudai no suitei ni tsuite (Exploiting Text strncture for Topic Identification), *Singaku Gihou* (IEICE Tech. Rep.), Vol. 96, No. 157, 47-54.

4. Yoshida, S., Nomoto, T. and Takahashi, H. (1996). Bunshou kouzou no chigai ga zero daimeishi no shouou ni oyobosu eikyou ni tsuite (On how text structure affects the resolution of zero pronouns), *Nihon Ninchi Kagakukai 13 Kai Taikai Ronbunshuu* (Proc. of the 13th meeting of JCSS), 178-179.

5. Nomoto, T. and Matsumoto, Y. (1997). Ningen no juuyoubun hantei ni motoduita jidou youyaku no kokoromi, *Jyouhou Shori Gakkai Kenkyuu Houkoku* (IPSJ Tech. Rep.), 97-NL-120, 71-76.

6. Yoshida, S., Nomoto, T., and Takahashi, H. (1997). Bun no kinsetsusei to zero daimeishi no shouou (Textual proximity and the resolution of zero pronouns),*Nihon Ninchi Kagakukai 14 Kai Taikai Ronbunshuu* (Proc. of the 14th meeting of JCSS), 118-119.

7. Yoshida, S., Nomoto, T., and Takahashi, H. (1998). Zero daimeishi no shouou ni kansuru segumento kasetsu no seiyaku ni tsuite (The 'Segment Hypothesis' and interpretation of zero pronouns), *Nihon Ninchi Kagakukai 15 Kai Taikai Ronbunshuu* (Proc. of the 15th meeting of JCSS), 52-53.

8. Nomoto, T. (2001) ModDBS-X^M: A diversity based summarizer for DUC. Presented at SIGIR Workshop on Text Summarization, New Orleans.

Book Articles

1. Nomoto, T. and Nitta, Y. (1994b). Structuring the Raw Discourse. In Daniel Jones and Harold Somers, editors, New Methods in Language Processing, UCL Press, London, 1997.

2. Nomoto, T. (1995). Effects of Grammatical Annotation on the Topic Identification Task. In Ruslan Mitkov and Nicolas Nicolov, editors, Recent Advances in Natural Language Processing, John Benjamins Publishing Co., Amsterdam/Philadelphia.