Discourse Parsing: A Decision Tree Approach

Tadashi Nomoto
Advanced Research Laboratory
Hitachi Ltd.
2520 Hatoyama Saitama,
350-0395 Japan
nomoto@harl.hitachi.co.jp

Yuji Matsumoto
Nara Institute of Science and Technology
8916-5 Takayama Ikoma Nara,
630-0101 Japan
matsu@is.aist-nara.ac.jp

Abstract

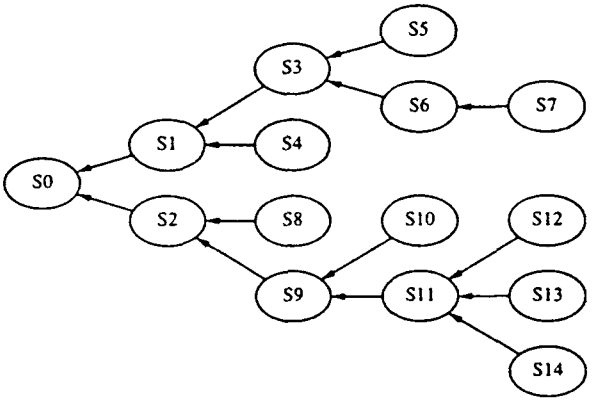
The paper presents a new statistical method for parsing discourse. A parse of discourse is defined as a set of semantic dependencies among sentences that make up the discourse. A collection of news articles from a Japanese economics daily are manually marked for dependency and used as a training/testing corpus. We use a C4.5 decision tree method to develop a model of sentential dependencies. However, rather than to use class decisions made by C4.5, we exploit information on class distributions to rank possible dependencies among sentences according to their probabilistic strength and take a parse to be a set of highest ranking dependencies. We also study effects of features such as clue words, distance and similarity on the performance of the discourse parser. Experiments have found that the method performs reasonably well on diverse text types, scoring an accuracy rate of over 60%.

1 Introduction

Attempts to the automatic identification of a structure in discourse have so far met with a limited success in the computational linguistics literature. Part of the reason is that, compared to sizable data resources available to parsing research such as the Penn Treebank (Marcus et al., 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 and Nagao, 1994; Litman and Passonneau, 1995; Hearst, 1994). The lack of a large-scale corpus has made it impossible to talk about results of discourse studies with the sufficient degree of reliability.

In the work described here, we created a corpus with discourse information, containing 645 articles from a Japanese economic paper, an order of magnitude larger than any previous work on discourse processing. It had a total of 12,770 sentences and 5,352 paragraphs. Each article in the corpus was manually annotated for a discourse dependency relation. We then built a statistical discourse parser based on the C4.5 decision tree method (Quinlan, 1993), which was trained and tested on the corpus we have created. The design of a parser was inspired by Haruno (1997)'s work on statistical sentence parsing.

Figure 1: A discourse tree. 'S' denotes a sentence.



The paper is organized as follows. Section 2 presents general ideas about statistical parsing as applied to the discourse, After a brief introduction to some of the points of a decision tree model, we discuss incorporating a decision tree within a statistical parsing model. In Section 3, we explain how we have built an annotated corpus. There we also describe a procedure of experiments we have conducted, and conclude the section with their results.

2 Statistical Discourse Parsing

First, let us make ourselves clear about what we mean by parsing a discourse. The job of parsing is to find whatever dependencies there are among elements that make up a particular linguistic unit. In discourse parsing, elements one is interested in finding dependencies among correspond to sentences, and a level of unit under investigation is a discourse. We take a naive approach to the notion of a dependency here. We think of it as a relationship between a pair of sentences such that the interpretation of one sentence in some way depends on that of the other. Thus a dependency relationship is not a structural one, but rather a semantic or rhetorical one.

The job of a discourse parser is to take as input

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 1). 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.

$$T_{best} = \arg\max_T P(T \mid D)$$

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:

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

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

2.1 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.

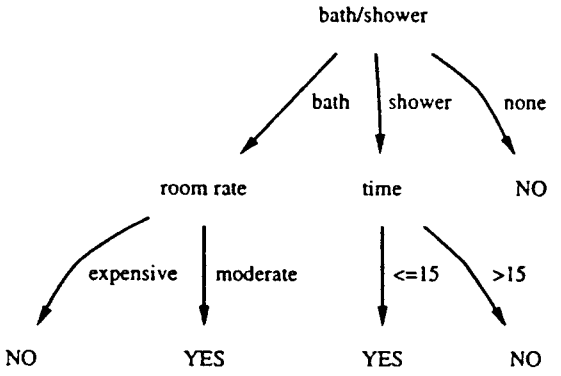
A decision tree method represents one of approaches to classification problems, where features are ranked according to how much they contribute to a classification, and models are then built with features most relevant to that classification. Suppose, for example, that you work for a travel agency and want to find out what features of a hotel are more important for tourists, based on data from your customers like Table 1. With decision tree techniques, you would be able to tell what features are more closely associated with customers' preferences.

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

Table 1: 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

Figure 2: A decision tree for the hotel example.

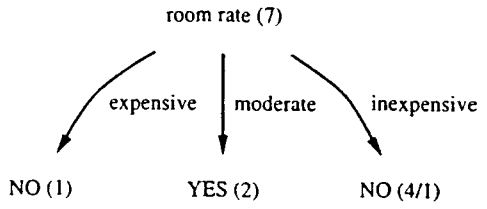


a particular case is placed is determined by the outcome of a 'test' on that case. Let us explain how this works by way of the hotel example above. Suppose that the first test is "bath/shower", which has three outcomes, `bath`, `shower`, and `none`. Then the data set breaks up into three groups, {1,4,5} (`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}.[1] Either one now contains cases from a single class. A decision tree for divisions we made is shown in Figure 2.

Now compare a hand-created decision tree in Fig-

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

Figure 3: 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, eg. (4/1), indicates the number of misclassified cases.



ure 2 with one in Figure 3, 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*.[2] The intuitive idea behind the criteria is to prefer a test with a least entropy, i.e., a test that partitions 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 1, 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 1 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.

2.2 Parsing with Decision Tree

As we mentioned in section 2, we define discourse parsing as a task of finding a best tree T, or a set of dependencies among sentences that maximizes $P(T \mid D)$.

$$T_{best} = \arg\max_T P(T \mid D)$$

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

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

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

$\mathbf{DT}_F$ 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 3. 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

[2] 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:

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

$freq(C, T)$ 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,

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

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.

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

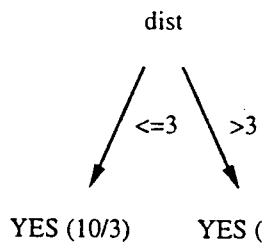
where:

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

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

[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.

Figure 4: A hypothetical decision tree.



$\mathbf{C}_F(D)$ 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 4.

With $\mathbf{C}_F(D)$ and $\mathbf{DT}_F$ at hand, we are now in a position to find $P(A \leftarrow B \mid \mathbf{C}_F(D))$, for each possible dependency $S_j \leftarrow S_i$, and $S_k \leftarrow S_i$.

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

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

Since S_i links with either S_j or S_k, by Equation 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.

2.2.1 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 ocurrs 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, $\#Par(\text{last\_sentence\_in\_text}) - 1$.

<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

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

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

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; 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.

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 $X = \{x_1, \ldots\}$ and $Y = \{y_1, \ldots\}$, where x and y are words, we define the lexical similarity between X and Y by:

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

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

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

Table 2: 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

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.

2.2.2 Notes on Discourse Encoding

Let us make further notes on how to encode a discourse with the set of features we have described. We characterize a sentence in relation to its potential "modifyee" sentence, a sentence in a discourse 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 $C_F(D) = C_F(\mathcal{P}(D))$. Furthermore, we may want to constrain $\mathcal{P}$ by restricting the attention to 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, we assumed a discourse as consisting of backward dependency pairs and encoded each pair with the set of features above. Assumptions we made about the structure of a discourse are the following:

1. Every sentence in a discourse has exactly one *preceding* "modifyee" to link to.
2. A discourse may have crossing dependencies.

3 Evaluation

3.1 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 modifyee 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 and Thompson, 1987). 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 says 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 and Thompson, 1987), 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 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:

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

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.

3.2 Results and Analyses

We list major results of the experiments in Table 3, 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 and Nagao, 1994; Litman and Passonneau, 1995; Grosz and Sidner, 1986; Marcu, 1997), where the notion of clues or cue phrases forms an important part of identifying a structure of discourse.[7]

Table 4 shows how the confidence value (CF) affects the performance of discourse models. The CF

[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 and Nagao, 1994); 15 narratives (1113 clauses) (Litman and Passonneau, 1995); 3 texts (Marcu, 1997). It is not clear how reliable estimates of performance obtained there would be.

Table 3: 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%)

Table 4: 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

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 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 4 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 4 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 5. 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 6. Since a sentence connective is relevant to establishing inter-sentential relationships, it was expected 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 5 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 5). 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 did not establish any correlation, either positive or negative, 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.

4 Conclusion

The paper 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 a decision tree and (b) the way a discourse is encoded. While we have found that distance and

Table 5: 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 (±0.000%)
Sim$^-$	0.643 (+0.160%)	0.640 (+0.790%)	0.632 (±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 (±0.000%)	0.632 (+0.640%)
LenSenB$^-$	0.642 (±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 (±0.000%)
IsAtitle$^-$	0.638 (−0.620%)	0.635 (±0.000%)	0.631 (−0.150%)	0.628 (±0.000%)

Figure 5: The ratio of improvement after removal of feature.

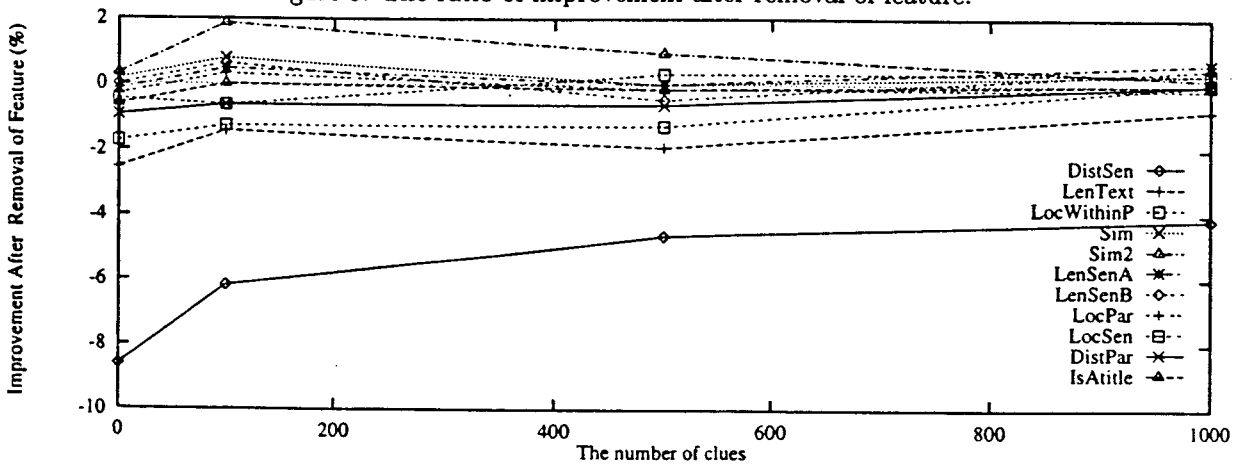


Table 6: Connectives found in the corpus. Underlined items (also marked with an asterisk) are those that the tokenizer program erroneously identified as a connective.

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*

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 works on discourse as well as on sentence parsing (Magerman, 1995; Collins, 1996).

The following are some of the future research issues:

Building a larger corpus Our discourse parser did not perform as well as a statistical sentence parser, which normally performs with over 80% precision. We suspect that the reason may have to do with inconsistencies in tagging and the size of the corpus we used.

Parsing with the rhetorical structure theory Technically it is straightforward to turn the present parsing method into a full-fledged RST parser, which involves modifying the way classes are defined and redefining constraints on a structure of discourse. A problem, however, is that the task of assigning sentences to rhetorical relations with some consistency could turn out to be quite difficult for human coders.

Extending to other Languages The general framework in which our parser is built does not presuppose elements specific to a particular language. It should be possible to carry it over to other languages with no significant modification to it.

References

Jean Carletta, Amy Isard, Stephen Isard, Jacqueline C. Kowtko, Gwyneth Doherty-Sneddon, and Anne H. Anderson. 1997. The Reliability of a Dialogue Structure Coding Scheme. *Computational Linguistics*, 23(1):13–31.

Michael John Collins. 1996. A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191, California, USA., June. Association for Computational Linguistics.

Barbara A. Fox. 1987. *Discourse structure and anaphora.* Cambridge Studies in Linguistics 48. Cambridge University Press, Cambridge, UK.

Barbara Grosz and Candance Sidner. 1986. Attention, Intentions and the Structure of Discourse. *Computational Linguistics*, 12(3):175–204.

M.A.K. Halliday and R. Hasan. 1990. *Cohesion in English.* Longman, New York.

Masahiko Haruno. 1997. Kettei-gi o mochiita nihongo kakariuke kaiseki (A Japanese Dependency Parser Using A Decision Tree). Submitted for publication.

Marti A. Hearst. 1994. Multi-Paragraph Segmentation of Expository Text. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 9–16, New Mexico, USA.

Sadao Kurohashi and Makoto Nagao. 1994. Automatic Detection od Discourse Structure by Checking Surface Information in Sentences. In *Proceedings of The 15th Intewrnational Conference on Computational Linguistics*, pages 1123–1127, August. Kyoto,Japan.

Stephen C. Levinson. 1994. *Pragmatics.* Cambridge Textbooks in Linguistics. Cambridge University Press.

Diane Litman and Rebecca J. Passonneau. 1995. Combining multiple knowledge sources for discourse segmentation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 108–115. The Association for Computational Linguistics, June. Cambridge, MASS. USA.

David M. Magerman. 1995. Statistical Decision-Tree Models for Parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, Camridge, MASS. USA., June. Association for Computational Linguistics.

William C. Mann and Sandra A. Thompson. 1987. Rhetorical Structure Theory : A Theory of Text Organization. Technical Report ISI/RS 87-190, ISI.

Daniel Marcu. 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*, pages 96–102, Madrid, Spain, July.

Mitcell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, June.

Nihon-Keizai-Shimbun-Sha. 1995. Nihon Keizai Shimbun 95 nen CD-ROM ban. CD-ROM. Nihon Keizai Shimbun, Inc., Tokyo.

J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning.* Morgan Kaufmann.

Hirofumi Sakurai and Toru Hisamitsu. 1997. Keitaiso Puroguramu ANIMA no Sekkei to Jissoo. In *Jyoohoo Shori Gakkai Zenki Zenkoku Taikai Kooen Ronbun Shuu*, volume 2, pages 57–56. Information Processing Society of Japan, March 12-14.