

Generating Discourse Structures for Written Text

Huong Le-Thanh, Geetha Abeysinghe, and Christian Huyck

School of Computing Science, Middlesex University

The Burroughs, London, NW4 4BT, United Kingdom

{H.Le, G.Abeysinghe, C.Huyck}@mdx.ac.uk

Abstract

This paper presents a system for automatically generating discourse structures from written text. The system is divided into two levels: sentence-level and text-level. The sentence-level discourse parser uses syntactic information and cue phrases to segment sentences into elementary discourse units and to generate discourse structures of sentences. At the text-level, constraints about textual adjacency and textual organisation are integrated in a beam search in order to generate best discourse structures. Our experiment with documents from the RST Discourse Treebank shows that this system can provide promising results in a reasonable search space when compared with the discourse trees generated by human analysts.

1 Introduction

Much recent research in Natural Language Processing has paid attention to Rhetorical Structure Theory (RST) (Mann and Thompson, 1988; Hovy, E., 1993; Marcu, 2000; Webber et al., 2001). It is a method of structured description of text (Mann and Thompson, 1988). Although rhetorical structure has been found to be useful in many fields of text processing (Rutledge et al., 2000; Torrance and Bouayad-Agha, 2001), only a few algorithms for implementing discourse analysers have been proposed so far. Most of research in this field concentrates on specific phenomena of discourse (Litman and Hirschberg, 1990; Umbach, 2001). The amount of research available in discourse segmentation is considered small, and the amount of research in discourse parsing is even smaller.

The difficulties for a discourse system are recognising discourse relations between text spans, and deriving discourse structures from these relations. Marcu (2000)'s parser is based on cue phrases, and therefore faces problems when cue phrases are not present in the text. Marcu's system can apply to unrestricted text, but faces combinatorial explosion. The disadvantage of

Marcu's approach is that it produces a great number of trees during its process, which is the essential redundancy in computation. As the number of relations increases, the number of possible discourse trees increases exponentially.

Webber (2001) had the different approach of implementing a discourse parser for a Lexicalized Tree Adjoining Grammar (LTAG). Webber developed a grammar that uses discourse cue as an anchor to connect textual trees. Like Marcu's system, Webber's parser cannot recognise discourse structures when there is no cue phrase present in the text.

In this paper, other factors are exploited in order to provide better results. These factors include syntactic information, lexical cohesion, textual adjacent constraint and textual organisation. Given a written text and its syntactic information, we aim to minimise the search space when producing well-structured discourse trees of text.

The rest of this paper is organised as follows. The discourse analyser at the sentence-level is presented in Section 2. A detailed description of our text-level discourse parser is given in Section 3. In Section 4, we describe our experiment and discuss the results we have achieved so far. Section 5 concludes the paper and proposes the possible future work of this approach.

2 Sentence-level Discourse Analysing

The sentence-level discourse analyser constructs discourse trees for each sentence. In doing this, two main tasks need to be accomplished: discourse segmentation and discourse parsing, which will be presented in Section 2.1 and Section 2.2.

2.1 Discourse Segmentation

The purpose of discourse segmentation is to split text into elementary discourse units (edus)¹. This task is done using syntactic information and cue phrases. The two processes of discourse segmentation will be discussed in Section 2.1.1 and Section 2.1.2 below.

¹ For further information on "edus", see (Marcu, 2000).

2.1.1 Discourse Segmentation by Syntax – Step 1

Since an elementary discourse unit can be a clause in a sentence or a simple sentence, syntactic information is very useful for the segmentation process. One may argue that using syntactic information is complicated since a syntactic parser is needed to generate this kind of information. Since there are many good syntactic parsers available nowadays, the above problem can be solved. Some research on this area has been based on regular expressions of cue phrases to identify elementary discourse units (e.g., Marcu, 2000). However, Redeker (1990) found that only 50% of clauses contain cue phrases. Therefore, segmentation based on cue phrases alone is not sufficient to rely upon.

The input of our discourse segmenter is a sentence and its syntactic information. Based on the syntactic structure of the sentence, the discourse segmenter detects all clauses in a sentence, and then checks segmentation's rules to split sentences into edus. The segmentation's rules are created based on previous research in discourse segmentation (Carlson et al., 2002). This process also provides initial information about the discourse relation between edus. For example, the sentence "*Mr. Silas Cathcart built a shopping mall on some land he owns*" maps with the segmentation's rule

(NP|NP-SBJ <text1> (SBAR|RRC <text2>))

NP, SBJ, SBAR, and RRC stand for noun phrase, subject, subordinate clause and relative clause, and reduce relative clause respectively. This rule can be stated as, "*The clause attached to a noun phrase can be recognised as an embedded unit.*"

The system first finds the segmentation's rule that maps with the syntactic structure of the sentence, and then generates edus. After that, a post process is called to check the correctness of discourse boundaries. In the above example, the system first creates two edus "*some land*" and "*he owns*". The post process detects that "*Mr. Silas Cathcart built a shopping mall on*" is not a complete clause without the noun phrase "*some land*". Therefore, these two text spans are combined into one. The two edus now become "*Mr. Silas Cathcart built a shopping mall on some land*" and "*he owns.*" A discourse relation between these two edus is then initiated. The nuclearity role and the name of this relation are decided later in another process.

2.1.2 Discourse Segmentation by Cue Phrases – Step 2

Several noun phrases are considered as edus when they are accompanied by a strong cue phrase. These cases cannot be recognised by syntactic information. Therefore, another segmentation process is integrated into the system to deal with such cases. This process finds strong cue phrases from the output of Step 1. When a strong cue phrase is found, the algorithm seeks the end boundary of the noun phrase. This end boundary can be punctuation such as a comma, a semicolon, or a full stop. Normally, a new edu is created from the beginning position of the cue phrase to the end boundary of the noun phrase. However, this procedure may create incorrect results as in example (1) shown below:

(1) [In 1988, Kidder eked out a \$46 million profit, *mainly*][because of severe cost cutting.]

The correct segmentation for the sentence given in example (1) is generated by a post process, and is given in example (2):

(2) [In 1988, Kidder eked out a \$46 million Profit,][*mainly* because of severe cost cutting.]

Such a situation happens when an adverb stands before the cue phrase. The post process deals with such cases by first detecting the position of the noun phrase, which will be an edu, and then checking for the appearance of adverbs before the strong cue phrase that makes the noun phrase become an edu. If an adverb is found, the new edu is segmented from the beginning position of the adverb to the end boundary of the noun phrase. Otherwise, the new edu is split from the beginning position of the cue phrase to the end boundary of the noun phrase. For example:

(3) [According to a Kidder World story about Mr. Megargel,] [all the firm has to do is "position ourselves more in the deal flow."]

When segmenting the output of the discourse segmenter by cue phrases, the system also initiates the discourse relation between new generated edus.

2.2 Sentence-level Discourse Parsing

This module takes elementary discourse units from the discourse segmenter as the input and generates discourse trees for each sentence. As mentioned in Section 2.1, many edus have already been connected together in an initial relation. The sentence-level discourse parser has to find the relation name for the existing relations, and then connect all sub-discourse-trees within one sentence into one tree. All tree leaves that correspond to another sub-tree are replaced by the corresponding sub-trees, as shown in example (4) below:

(4) [[She knows] [what time you will come]]
 [[because I told her yesterday.]]

The discourse segmenter in Step 1 outputs two subtrees, one with two leaves “*She knows*” and “*what time you will come*”; another with two leaves “*She knows what time you will come*” and “*because I told her yesterday*”. The system combines these two sub-trees into one discourse tree. The connecting process of example (4) is illustrated in Figure 1.

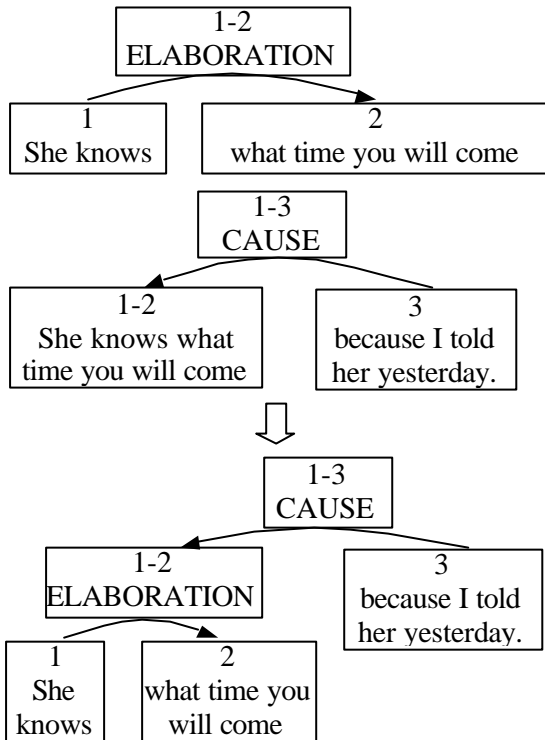


Figure 1. The discourse structure of text (4)

Syntactic information is used to figure out which discourse relation holds between clauses as well as their nuclearity roles. For example, the discourse relation between a reporting clause and a reported clause in a sentence is an ELABORATION relation. The reporting clause is the nucleus; the reported clause is the satellite in that relation.

Cue phrases are also used to detect the connection among clauses in a sentence, as in example (5) shown below:

(5) [He came late] [*because of* the traffic.]

The cue phrase “*because of*” recognises a relation between the clause containing this cue phrase and its adjacent clause. The clause containing “*because of*” is the satellite in a relation between this clause and its adjacent clause.

The method of constructing the discourse tree for each sentence based on syntactic information and cue phrase, which is represented in this section, helps the analysing task to be done accurately and prevents a combinatorial explosion.

The relation names of discourse relations in a sentence are recognised based on syntactic information, cue phrases and cohesive devices

(synonyms, hyponyms, coreferences, etc.). Due to space constraints, a detailed description of the method to recognise discourse relations within a sentence is not presented in this paper.

3 Text-level Discourse Analysing

3.1 The Search Space

As stated in (Marcu, 2000), the search space of the discourse parser is enormous. Therefore, the crucial problem in discourse parsing is search-space reduction. We solve it using constraints about textual organisation and textual adjacency, which will be discussed in the rest of this section.

Normally, a text is divided into sections, paragraphs, sentences, clauses, etc., depending on the intention of the writer. Each of these textual blocks completes an idea, an expression, or a topic. Therefore, each text span should connect with text spans in the same textual block before connecting with text spans in a different block. To generate the discourse structures of a text, our algorithm firstly constructs the discourse structure for each sentence, then continues with higher levels (paragraph, section, etc.) until reaching a total solution (i.e., the discourse tree for the entire text is generated). This divide-and-conquer strategy is especially important in reducing the search space of long texts.

Adjacency is one of the four main constraints in constructing a valid structural analysis of a text (Mann and Thompson, 1988). Based on this constraint, we only consider adjacent text spans in generating new discourse relations. This approach reduces the search space remarkably since most of the text spans corresponding to sub-discourse-trees in the searching space are unadjacent. This search space is much smaller than that in (Marcu, 2000). The reason is Marcu’s system generates all possible trees, and then uses this constraint to filter the inappropriate ones.

3.2 The Algorithm

The discourse tree of a sentence in our system can be derived with high accuracy based on the syntactic structure of the sentence and cue phrases of this sentence. To generate discourse structures at the text-level, other information is taken into account, including information about textual organisation (e.g., sections, paragraphs, etc.), the adjacent property of text spans, and all possible discourse relations between text spans. The hypothetic discourse relations between text spans are recognised based on cue phrases, NP-cues²,

² A NP-cue is a special noun in the noun phrase, which can be used to recognise discourse relations.

VP-cues³ and cohesive devices. The task of constructing discourse trees at the text-level can be stated as follows:

“Given a set of sentence’s discourse trees $\{S_1, S_2, \dots, S_N\}$, and all possible discourse relations between them, generate a discourse tree that best describes the discourse structure of the text.”

The problem of generating the best discourse tree can be considered as searching for the best solution of combining discourse relations. We need to find an algorithm that minimises the search space, and maximises the tree’s quality. For this, we apply a beam search, which is the optimisation of the best-first search where only a predetermined number of paths are kept as candidates. The best-first combines the advantages of both depth-first and breadth-first search. The best-first search follows a single path at a time, but switches paths whenever some competing path looks more promising than the current one. The rest of this Section will describe this algorithm in detail.

A set called *Subtrees* is used to store temporal sub-discourse-trees created during the searching process. All possible relations that can be used to construct bigger trees at a time t form a hypothesis set *PotentialH*. This set contains hypotheses about relations between pairs of adjacent text spans. Each hypothesis is assigned a score called a *heuristic-score*, which is equal to the sum of the heuristic scores of each cue contributing to that relation. The score of a cue is between 0 and 100.

In order to control the block level (e.g., paragraph, section, etc.), each sub-discourse-tree is assigned a *block-level-score*, depending on the block levels of their children. This block-level-score is added to the heuristic-score, aiming at choosing the best combination of sub-trees to be applied in the next round. The value of a block-level-score is set in a different value-scale, so that the combination of sub-trees in the same textual block always has a higher score than that in a different textual block.

If two sub-trees are in different sentences, but in the same paragraph, the tree that connects these sub-trees will have the block-level-score = 0.

If two sub-trees are in different paragraphs, but in the same section, and assuming that there is no sub-section in that section, then the tree that connects these sub-trees will have the block-level-score = -1000. This negative value means the higher distance between two text spans, the lower combinatorial priority they get.

In general, if two sub-trees are in different units (e.g., sentence, paragraph, etc.) of a block level L_i ,

but in the same block level L_j , the tree that connects these sub-trees will have the block-level-score = $-1000 * (L_j - L_i)$. L_j is the higher text level than L_i ; L_i is the highest block level that these sub-trees are in different unit; L_j is the lowest block level that they are in the same unit.

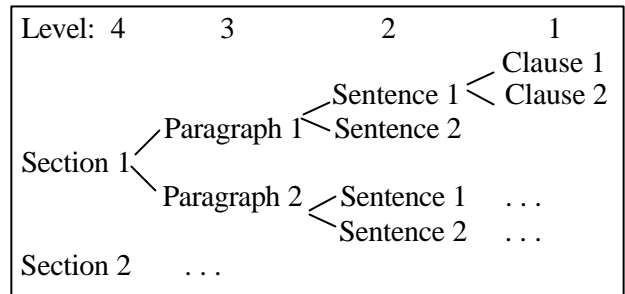


Figure 2. Example of calculating the block-level-score

When selecting the hypothesized discourse relations to be applied, the discourse relation corresponding to the node with a higher block-level-score will have a higher priority than the node with a lower one. If hypotheses have the same block-level-score, the one with higher heuristic-score will be chosen.

To simplify the searching algorithm, an *accumulation-score* is used to store the value of the search path. The accumulation-score of a path at one step is the highest *predict-score* of that path at the previous step. The predict-score of one step is equal to the sum of the accumulation-score, the heuristic-score and the block-level-score of this step. The searching process now becomes the process of searching for the hypothesis with highest predict-score.

At each step of the best-first search process, we select the most promising nodes from the set *PotentialH* we have generated so far. If a hypothesis involving two text spans $\langle T_i, T_j \rangle$ is used, the new sub-tree created by joining the two sub-trees corresponding to text spans $\langle T_i \rangle$ and $\langle T_j \rangle$, is added to the set *Subtrees*. The set *Subtrees* are now updated so that it does not contain overlapping sub-discourse-trees. The set *PotentialH* is also changed according to the change in *Subtrees*. The relations between the new sub-tree and its adjacent sub-trees in the set *Subtrees* are created and added to *PotentialH*.

All hypotheses computed by the discourse parsing system are stored in a hypothesis set called *StoredH*. This set is used to guarantee that a discourse sub-tree will not be created twice. When detecting a relation between two text spans, the parser first looks for this relation in the set *StoredH* to check whether it has already been created or not. If it is not found, it will be generated by a discourse recogniser.

³ A VP-cue is a special verb in the verb phrase, which can be used to recognise discourse relations.

Again the most promising node from *PotentialH* is selected and the process continues. A bit of depth-first searching occurs as the most promising branch is explored. If a solution is not found, that branch will start looking for a less promising node in one of the higher-level branches that had been ignored. At that point, the now more promising, previously ignored branch will be explored. The last node of the old branch is stored in the set of generated and unexpanded nodes. The searching process can return to it whenever all the others get bad enough that it is again the most promising path. In our algorithm, we limit the branches that the search algorithm can switch to by a number *M*, which is set to be 10. If *Subtrees* contains only one tree, this tree will be added to the tree's set.⁴ The searching algorithm terminates when the number of discourse trees is equal to the number of trees required by the user.

4 Evaluation

The experiment was done by testing twenty documents from the RST Discourse Treebank (RST-DT, 2002), including ten short documents and ten long ones. The lengths of the documents vary from 30 words to 1284 words. The syntactic information of these RST documents was taken from Penn Treebank, which was used as the input of our discourse segmenter. Figure 3 displays the discourse tree of the document 0600 from the RST Discourse Treebank, generated by our system.

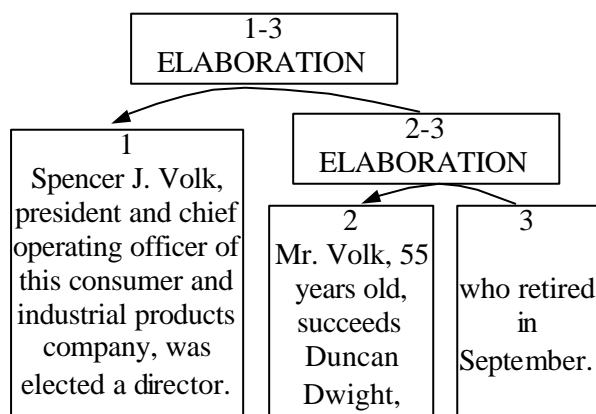


Figure 3. The discourse structure of the document 0600 from Penn Tree Bank

In order to evaluate the system, we used a set of 22 discourse relations (list, sequence, condition, otherwise, hypothetical, antithesis, contrast, concession, cause, result, cause-result, purpose, solutionhood, circumstance, manner, means,

interpretation, evaluation, summary, elaboration, explanation, and joint). The different between “cause”, “result” and “cause-result” is the nuclearity role of text spans in the relation. We carried out another evaluation with the set of 14 relations, in which similar relations in the set of 22 relations are grouped into one relation.

The discourse trees from RST discourse corpus, which created by humans, are used as the standard discourse trees for our evaluation. We compute the output's accuracy on seven levels shown below:

- Level 1 - The accuracy of discourse segment boundaries. It is calculated by comparing the boundary's positions assigned by the discourse segmenter with the boundary's positions assigned by human.
- Level 2 - The accuracy of the combination of text spans in the sentence-level. The system generates a correct combination if it connects the same text spans as human.
- Level 3 - The accuracy of the nuclearity of text spans in the sentence-level.
- Level 4a - The accuracy of discourse relations in the sentence-level (with the set of 22 relations).
- Level 4b - The accuracy of discourse relations in the sentence-level (with the set of 14 relations).
- Level 5 - The total accuracy of combination of text spans for the entire text.
- Level 6 - The total accuracy of the nuclearity of text spans for the entire text.
- Level 7a - The total accuracy of discourse relations for the entire text (with the set of 22 relations).
- Level 7b - The total accuracy of discourse relations for the entire text (with the set of 14 relations).

In our experiment, the output of the previous process is used as the input of the process following it. Therefore, the accuracy of one level is affected by the accuracies of levels before it. The human performance is considered as the upper bound for our discourse parser's performance. This value is obtained by evaluating the agreement between human annotators using 53 double-annotated documents from the RST Discourse Treebank. The performance of our system and human agreement are represented by precision, recall, and F-score⁵, which are shown in Table 1.

⁴ If no relation can be found between two discourse sub-trees, a JOINT relation is assigned instead. Therefore, there is always a discourse tree available that can cover the entire text.

⁵ The F-score is a measure combining precision and recall into a single figure. We use the version in which they are weighted equally, defined as $2 * \text{precision} * \text{recall} / (\text{precision} + \text{recall})$.

Level		1	2	3	4a	4b	5	6	7a	7b
System	Precision	88.2	68.4	61.9	53.9	54.6	54.5	47.8	39.6	40.5
	Recall	85.6	64.4	58.3	50.7	51.4	52.9	46.4	38.5	39.3
	F-score	86.9	66.3	60.0	52.2	53.0	53.7	47.1	39.1	39.9
Human	Precision	98.7	88.4	82.6	69.2	74.7	73.0	65.9	53.0	57.1
	Recall	98.8	88.1	82.3	68.9	74.4	72.4	65.3	52.5	56.6
	F-score	98.7	88.3	82.4	69.0	74.5	72.7	65.6	52.7	56.9
F-score(Human) – F-score(System)		11.8	22	22.4	16.8	21.5	19.0	18.5	13.7	17.0

Table 1: Our system’s performance versus human performance

The F-score of our discourse segmenter is 86.9%, while the F-score of human agreement is 98.7%. The level 2’s F-score of our system is 66.3%, which means the error in this case is 28.7%. This error is the accumulation of errors made by the discourse segmenter and errors in discourse combination, given correct discourse segments. With the set of 14 discourse relations, the F-score of discourse relations at the sentence-level using 14 relations (53.0%) is higher than the case of using 22 relations (52.2%).

The most recent sentence-level discourse parser providing good results is SPADE, which is reported in (Soricut and Marcu, 2003). SPADE includes two probabilistic models that can be used to identify elementary discourse units and build sentence-level discourse parse trees. RST Discourse Treebank was also used in their experiment, in which 347 were used as the training set and 38 articles were used as the test set. However, documents from RST Discourse Treebank need to be modified in an artificial way before they can be used as the SPADE’s input. For example, `<s>` and `</s>` are inserted before and after each sentence; and “15%” is changed to “15 %”. This manual process is not required by our system.

Soricut and Marcu evaluated their system using slightly different criteria than those used in this research. They computed the accuracy of the discourse segments, and the accuracy of the sentence-level discourse trees without labels, with 18 labels and with 110 labels. It is not clear how the sentence-level discourse trees are considered as correct. Due to this reason, the performance given by the human annotation agreement reported by them is different than the calculation used in this paper. To compare the performance between our system and Soricut and Marcu’s system at the sentence-level, we calculate the difference of F-score between the system and the analyst. Table 2 presents the performance of Soricut and Marcu’s system when syntactic trees from the Penn Treebank are used as the input.

	Discourse segments	Un-labeled	110 labels	18 labels
Soricut and Marcu	84.7	73.0	52.6	56.4
Human	98.3	92.8	71.9	77.0
F-score(H) – F-score(S)	13.6	19.8	19.3	20.6

Table 2: Soricut and Marcu’s system (2003) performance vs. human performance

Table 1 and Table 2 show that our discourse segmenter has a better performance than Soricut and Marcu’s system. We consider the evaluation of the “Unlabeled” case in Soricut and Marcu’s experiment as the evaluation of Level 2 in our experiment, which is the accuracy of the combination of text spans in the sentence-level. We cannot have an exact comparison on the accuracy of relation’s names between these two systems since their criteria to evaluate are not the same. However, the values shown in Table 1 and Table 2 imply that the error made by our system, in comparison with human agreement, can be considered as similar to that in Soricut and Marcu (2003)’s system, in respect of discourse relation’s names.

To our knowledge, there is only one report about the accuracy of discourse parser at the text-level, written by (Marcu, 2000). When using WSJ documents from the Penn Tree Bank, Marcu’s decision-tree-based discourse parser received 21.6% recall and 54.0% precision for the span nuclearity; 13.0% recall and 34.3% precision for discourse relations. The recall feature is more important than the precision feature since we want to have as correct discourse relations as possible. Although our discourse parser’s performance is higher than that in (Marcu, 2000), more work need to be done to improve the system’s reliability.

As we can see from Table 1, the accuracy of the discourse trees given by human agreement is not high, 52.7% in case of 22 relations and 56.9% in case of 14 relations. It is because discourse is too complex and too ill defined to generate rules that can automatically derive discourse structures. Different people may create different discourse

trees for the same text (Mann and Thompson, 1988). Because of the multiplicity of RST analyses, the discourse parser should be used as an assistant rather than a completed reliable system. For that purpose, the output of our system is designed to be editable by normal users (e.g., secondary students) through a friendly human interface. The RST Tool created by M. O'Donnell (2002) is used for this purpose.

5 Conclusion and Future Work

In this paper, we have presented a discourse parser and evaluated it using the RST discourse corpus. Our discourse parser is divided into two levels: sentence-level and text-level. The experiment shows that syntactic information and cue phrases are quite effective in constructing discourse structures at the sentence-level, especially in discourse segmentation (86.9% F-score). The discourse trees at the text-level are generated by combining hypothesized discourse relations among non-overlapped text spans. We concentrated on solving the combinatorial explosion in searching for discourse trees. We apply a divide-and-conquer strategy, the adjacent constraint, and a beam search to find the best-quality trees in a search space that is much smaller than the space given by Marcu (2000) and Corston-Oliver (1998). Our experiment on documents from the RST Discourse Treebank shows that this approach can produce reasonable results in constructing discourse trees from text, in comparison with human annotator agreement. However, more work needs to be done to improve the system's performance, such as improving criteria to select the optimal path in the beam search. We hope this research will aid in the future development of text processing such as text summarisation and text generation.

References

- S. Corston-Oliver. 1998. *Computing Representations of the Structure of Written Discourse*. PhD Thesis. University of California, Santa Barbara, CA, U.S.A.
- GATE. 2004. <http://www.gate.ac.uk/>
- E. Hovy. 1993. *Automated Discourse Generation Using Discourse Structure Relations*. *Artificial Intelligence*, 63: 341-386.
- D. Litman & J. Hirschberg. 1990. *Disambiguating cue phrases in text and speech*. In Proc of COLING-90. Vol 2, pp 251-256.
- W. C. Mann, and S. A. Thompson. 1988. *Rhetorical Structure Theory: Toward a Functional Theory of Text Organization*. *Text*, vol. 8(3), 243-281.

- D. Marcu. 2000. *The theory and practice of discourse parsing and summarization*. MIT Press, Cambridge, Massachusetts, London, England.
- M. P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1995. *Penn Treebank II*, LDC.
- C. Mellish, A. Knott, J. Oberlander, and M. O'Donnell. 1998. *Experiments Using Stochastic Search for Text Planning*. In Proceedings of IWNLG-98, Niagara-on-the-Lake, Canada. Association for Computational Linguistics.
- M. O'Donnell. Last visited 2004. RSTTool -- an RST Markup Tool. <http://www.wagsoft.com/RSTTool/index.html>.
- G. Redeker. 1990. Ideational and pragmatic markers of discourse structure. *Journal of Pragmatics*, 367-381.
- RST-DT. 2002. *RST Discourse Treebank*. Linguistic Data Consortium. <http://www ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002T07>.
- L. Rutledge, B. Bailey, J. v. Ossenbruggen, L. Hardman, and J. Geurts. 2000. *Generating Presentation Constraints from Rhetorical Structure*. In Proceedings of the 11th ACM conference on Hypertext and Hypermedia. San Antonio, Texas, USA pp. 19-28.
- R. Soricut and D. Marcu. 2003. *Sentence Level Discourse Parsing using Syntactic and Lexical Information*. In Proceedings of HLT-NAACL 2003.
- M. Torrance, and N. Bouayad-Agha. 2001. *Rhetorical structure analysis as a method for understanding writing processes*. Proceedings of the International Workshop on Multi-disciplinary Approaches of discourse (MAD 2001), pp. 51-59, Amsterdam & Nodus Publications.
- C. Umbach. 2001. *Contrast and Contrastive Topic*. In the ESSLLI 2001 Workshop on Information Structure, Discourse Structure and Discourse Semantics.
- B. Webber et al. 2001. *D-LTAG System - Discourse Parsing with a Lexicalised Tree Adjoining Grammar*. In: ESSLLI Workshop on Information structure, Discourse structure and Discourse Semantics.