Generating Discourse Structures for Written Texts

Huong LeThanh, 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 organization are integrated in a beam search in order to generate best discourse structures. The experiments were done with documents from the RST Discourse Treebank. It shows promising results in a reasonable search space compared to the discourse trees generated by human analysts.

1 Introduction

Many recent studies in Natural Language Processing have paid attention to Rhetorical Structure Theory (RST) (Mann and Thompson 1988; Hovy 1993; Marcu 2000; Forbes et al. 2003), a method of structured description of text. 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 analyzers have been proposed so far. Most research in this field concentrates on specific discourse phenomena (Schiffrin 1987; Litman and Hirschberg 1990). The amount of research available in discourse segmentation is considered small; in discourse parsing it is even smaller.

The difficulties in developing a discourse parser are (i) recognizing discourse relations between text spans and (ii) 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. This system can apply to unrestricted texts, 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.

Forbes et al. (2003) have a different approach of implementing a discourse parser for a Lexicalized Tree Adjoining Grammar (LTAG). They simplify discourse analysis by developing a grammar that uses cue phrases as anchors to connect discourse trees. Despite the potential of this approach for discourse analysis, the case of no cue phrase present in the text has not been fully investigated in their research. Polanyi et al. (2004) propose a far more complicated discourse system than that of Forbes et al. (2003), which uses syntactic, semantic and lexical rules. Polanyi et al. have proved that their approach can provide promising results, especially in text summarization.

In this paper, different factors were investigated to achieve a better discourse parser, including syntactic information, constraints about textual adjacency and textual organization. With a given text and its syntactic information, the search space in which well-structured discourse trees of a text are produced is minimized.

The rest of this paper is organized as follows. The discourse analyzer 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 experiments and discuss the results we have achieved so far. Section 5 concludes the paper and proposes possible future work on this approach.

2 Sentence-level Discourse Analyzing

The sentence-level discourse analyzer constructs discourse trees for each sentence. In doing so,

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 a text into elementary discourse units (edus)[1]. This task is done using syntactic information and cue phrases, as discussed in Section 2.1.1 and Section 2.1.2 below.

2.1.1 Segmentation by Syntax – Step 1

Since an edu can be a clause or a simple sentence, syntactic information is useful for the segmentation process. One may argue that using syntactic information is complicated since a syntactic parser is needed to generate this information. Since there are many advanced syntactic parsers currently available, the above problem can be solved. Some studies in this area were based on regular expressions of cue phrases to identify edus (e.g., Marcu 2000). However, Redeker (1990) found that only 50% of clauses contain cue phrases. Segmentation based on cue phrases alone is, therefore, insufficient by itself.

In this study, the segmenter's input is a sentence and its syntactic structure; documents from the Penn Treebank were used to get the syntactic information. A syntactic parser is going to be integrated into our system (see future work).

Based on the sentential syntactic structure, the discourse segmenter checks segmentation rules to split sentences into edus. These rules were created based on previous research in discourse segmentation (Carlson et al. 2002). The segmentation 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 rule

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

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

The system searches for the 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 derives an edu "*he owns*" from the noun phrase "*some land 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 sentence is now split into two edus "*Mr. Silas Cathcart built a shopping mall on some land*" and "*he owns.*" A discourse relation between these two edus is then initiated. Its relation's name and the nuclearity roles of its text spans are determined later on in a relation recognition-process (see Section 2.2).

2.1.2 Segmentation by Cue Phrase–Step 2

Several NPs are considered as edus when they are accompanied by a strong cue phrase. These cases cannot be recognized by syntactic information; another segmentation process is, therefore, integrated into the system. This process seeks strong cue phrases from the output of Step 1. When a strong cue phrase is found, this process detects the end boundary of the NP. This end boundary can be punctuation such as a semicolon, or a full stop. Normally, a new edu is created from the begin position of the cue phrase to the end boundary of the NP. However, this procedure may create incorrect results as shown in the example below:

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

The correct segmentation boundary for the sentence given in Example (1) should be the position between the comma (',') and the adverb "*mainly*". Such a situation happens when an adverb stands before a strong cue phrase. The post process deals with this case by first detecting the position of the NP. After that, it searches for the appearance of adverbs before the position of the strong cue phrase. If an adverb is found, the new edu is segmented from the start position of the adverb to the end boundary of the NP. Otherwise, the new edu is split from the start position of the cue phrase to the end boundary of the NP. This is shown in the following example:

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

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

Similar to Step 1, Step 2 also initiates discourse relations between edus that it derives. The relation name and the nuclearity role of edus are posited later in a relation recognition-process.

2.2 Sentence-level Discourse Parsing

This module takes edus from the segmenter as the input and generates discourse trees for each sentence. As mentioned in Section 2.1, many edus have already been connected in an initial relation. The sentence-level discourse parser finds a relation name for the existing relations, and then connects all sub-discourse-trees within one sentence into one tree. All leaves that correspond to another sub-tree are replaced by the corresponding sub-trees, as shown in Example (3) below:

(3) [She knows$_{3.1}$] [what time you will come$_{3.2}$][because I told her yesterday.$_{3.3}$]

The discourse segmenter in Step 1 outputs two sub-trees, 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 tree. This process is illustrated in Figure 1.

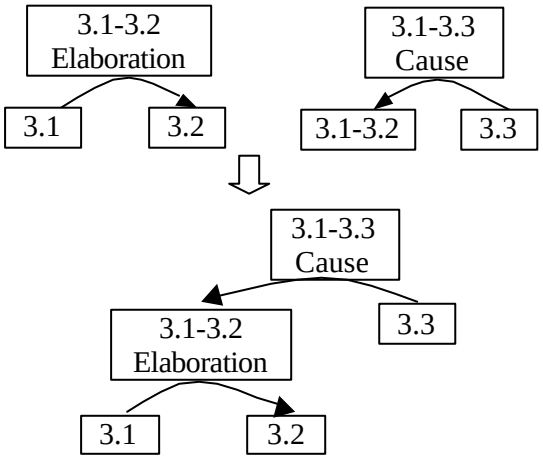

Figure 1. The discourse structure of text (3)

Syntactic information is used to figure out which discourse relation holds between text spans 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 this relation.

Cue phrases are also used to detect the connection between edus, as shown in (4):

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

The cue phrase "*because of*" signals a Cause 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.

To posit relation names, we combine several factors, including syntactic information, cue phrases, NP-cues, VP-cues[2], and cohesive devices (e.g., synonyms and hyponyms derived from WordNet) (Le and Abeysinghe 2003). With the presented method of constructing sentential discourse trees based on syntactic information and cue phrases, combinatorial explosions can be prevented and still get accurate analyses.

3 Text-level Discourse Analyzing

3.1 Search Space

The original search space of a discourse parser is enormous (Marcu 2000). Therefore, a crucial problem in discourse parsing is search-space reduction. In this study, this problem was solved by using constraints about textual organization and textual adjacency.

Normally, each text has an organizational framework, which consists of sections, paragraphs, etc., to express a communicative goal. Each textual unit completes an argument or a topic that the writer intends to convey. Thus, a text span should have semantic links to text spans in the same textual unit before connecting with text spans in a different one. Marcu (2000) applied this constraint by generating discourse structures at each level of granularity (e.g., paragraph, section). The discourse trees at one level are used to build the discourse trees at the higher level, until the discourse tree for the entire text is generated. Although this approach is good for deriving all valid discourse structures that represent the text, it is not optimal when only some discourse trees are required. This is because the parser cannot determine how many discourse trees should be generated for each paragraph or section. In this research, we apply a different approach to control the levels of granularity. Instead of processing one textual unit at a time, we use a *block-level-score* to connect the text spans

[2] An NP-cue (VP-cue) is a special noun (verb) in the NP (VP) that signals discourse relations.

that are in the same textual unit. A detailed description of the *block-level-score* is presented in Section 3.2. The parser completes its task when the required number of discourse trees that cover the entire text is achieved.

The second factor that is used to reduce the search space is the textual adjacency constraint. This is one of the four main constraints in constructing a valid discourse structure (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-trees in the search space are not adjacent. This search space is much smaller than the one in Marcu's (2000) because Marcu's system generates all possible trees, and then uses this constraint to filter the inappropriate ones.

3.2 Algorithm

To generate discourse structures at the text-level, the constraints of textual organization and textual adjacency are used to initiate all possible connections among text spans. Then, all possible discourse relations between text spans are posited based on cue phrases, NP-cues, VP-cues and other cohesive devices (Le and Abeysinghe 2003). Based on this relation set, the system should generate the best discourse trees, each of which covers the entire text. This problem can be considered as searching for the best solution of combining discourse relations. An algorithm that minimizes the search space and maximizes the tree's quality needs to be found. We apply a beam search, which is the optimization of the best-first search where only a predetermined number of paths are kept as candidates. This algorithm is described in detail below.

A set called *Subtrees* is used to store sub-trees that have been created during the constructing process. This set starts with sentential discourse trees. As sub-trees corresponding to contiguous text spans are grouped together to form bigger trees, *Subtrees* contains fewer and fewer members. When *Subtrees* contains only one tree, this tree will represent the discourse structure of the input text. All possible relations that can be used to construct bigger trees at a time *t* form a hypothesis set *PotentialH*. Each relation in this set, which is called a hypothesis, is assigned a score called a *heuristic-score*, which is equal to the total score of all discourse cues contributing to this relation. A cue's score is between 0 and 100, depending on its certainty in signaling a specific relation. This score can be optimized by a training process, which evaluates the correctness of the parser's output with the discourse trees from an existing discourse corpus. At present, these scores are assigned by our empirical research.

In order to control the textual block level, each sub-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 priority than that in a different block.

- If two sub-trees are 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, the *block-level-score* of their parent tree is equal to -1000 * (Li-L0), in which L0 is the paragraph level, Li is the lowest block level that two sub-trees are in the same unit. For example, if two sub-trees are in the same section but in different paragraphs; and there is no subsection in this section; then Li-L0 is equal to 1. The negative value (-1000) means the higher distance between two text spans, the lower combinatorial priority they get.

When selecting a discourse relation, the relation corresponding to the node with a higher *block-level-score* has a higher priority than the node with a lower one. If relations have the same *block-level-score*, the one with higher *heuristic-score* is chosen.

To simplify the searching process, an *accumulated-score* is used to store the value of the search path. The *accumulated-score* of a path at one step is the highest *predicted-score* of this path at the previous step. The *predicted-score* of one step is equal to the sum of the *accumulated-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 *predicted-score*.

At each step of the beam search, we select the most promising nodes from *PotentialH* that have

been generated so far. If a hypothesis involving two text spans <Ti> and <Tj> is used, the new sub-tree created by joining the two sub-trees corresponding to these text spans is added to *Subtrees*. *Subtrees* is now updated so that it does not contain overlapping sub-trees. *PotentialH* is also updated according to the change in *Subtrees*. The relations between the new sub-tree and its adjacent sub-trees in *Subtrees* are created and added to *PotentialH*.

All hypotheses computed by the discourse parser 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 *StoredH* to check whether it has already been created or not. If it is not found, it will be generated by a discourse relation recognizer.

The most promising node from *PotentialH* is again 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, the system will start looking for a less promising node in one of the higher-level branches that had been ignored. The last node of the old branch is stored in the system. The searching process returns to this node when 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. This number is chosen to be 10, as in experiments we found that it is large enough to derive good discourse trees. If *Subtrees* contains only one tree, this tree will be added to the tree's set.[3] The searching algorithm finishes when the number of discourse trees is equal to the number of trees required by the user. Since the parser searches for combinations of discourse relations that maximize the *accumulated-score*, which represents the tree's quality, the trees being generated are often the best descriptions of the text.

4 Evaluation

The experiments were done by testing 20 documents from the RST Discourse Treebank (RST-DT 2002), including ten short documents and ten long ones. The length of the documents varies from 30 words to 1284 words. The syntactic information of these documents was taken from Penn Treebank, which was used as the input of the discourse segmenter. In order to evaluate the system, 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) was used.[4] The difference among *cause*, *result* and *cause-result* is the nuclearity role of text spans. We also carried out another evaluation with the set of 14 relations, which was created by grouping similar relations in the set of 22 relations. The RST corpus, which was created by humans, was used as the standard discourse trees for our evaluation. We computed the output's accuracy on seven levels shown below:

- Level 1 - The accuracy of discourse segments. It was calculated by comparing the segment boundaries assigned by the discourse segmenter with the boundaries assigned in the corpus.
- Level 2 - The accuracy of text spans' combination at the sentence-level. The system generates a correct combination if it connects the same text spans as the corpus.
- Level 3 - The accuracy of the nuclearity role of text spans at the sentence-level.
- Level 4 - The accuracy of discourse relations at the sentence-level, using the set of 22 relations (level 4a), and the set of 14 relations (level 4b).
- Level 5 - The accuracy of text spans' combination for the entire text.
- Level 6 - The accuracy of the nuclearity role of text spans for the entire text.
- Level 7 - The accuracy of discourse relations for the entire text, using the set of 22 relations (level 7a), and the set of 14 relations (level 7b).

The system performance when the output of a syntactic parser is used as the input of our discourse segmenter will be evaluated in the future, when a syntactic parser is integrated with our system. It is also interesting to evaluate the per-

[3] If no relation is found between two discourse sub-trees, a Joint relation is assigned. Thus, a discourse tree that covers the entire text can always be found.

[4] See (Le and Abeysinghe 2003) for a detailed description of this discourse relation set.

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 performance vs. human performance

formance of the discourse parser when the correct discourse segments generated by an analyst are used as the input, so that we can calculate the accuracy of our system in determining discourse relations. This evaluation will be done in our future work.

In our experiment, the output of the previous process was used as the input of the process following it. Therefore, the accuracy of one level is affected by the accuracies of the previous levels. The human performance was considered as the upper bound for our discourse parser's performance. This value was obtained by evaluating the agreement between human annotators using 53 double-annotated documents from the RST corpus. The performance of our system and human agreement are represented by precision, recall, and F-score[5], which are shown in Table 1.

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 edus and build sentence-level discourse parse trees. The RST corpus was also used in Soricut and Marcu (S&M)'s experiment, in which 347 articles were used as the training set and 38 ones were used as the test set. S&M 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. The performance given by the human annotation agreement reported by S&M is, therefore, different than the one used in this paper. To compare the performance between our system and SPADE at the sentence-level, we calculated the difference of F-score between the system and the analyst. Table 2 presents the performance of SPADE when syntactic trees from the Penn Treebank were used as the input.

	Discourse segments	Unlabelled	110 labels	18 labels
SPADE	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. SPADE performance vs. human performance

Table 1 and Table 2 show that the discourse segmenter in our study has a better performance than SPADE. We considered the evaluation of the *"Unlabelled"* case in S&M's experiment as the evaluation of Level 2 in our experiment. The values shown in Table 1 and Table 2 imply that the error generated by our system is considered similar to the one in SPADE.

To our knowledge, there is only one report about a discourse parser at the text-level that measures accuracy (Marcu 2000). When using WSJ documents from the Penn Treebank, Marcu's decision-tree-based discourse parser received 21.6% recall and 54.0% precision for the

[5] The F-score is a measure combining into a single figure. We use the F-score version in which precision (P) and recall (R) are weighted equally, defined as 2*P*R/(P+R).

span nuclearity; 13.0% recall and 34.3% precision for discourse relations. The recall is more important than the precision since we want discourse relations that are as correct as possible. Therefore, the discourse parser presented in this paper shows a better performance. However, more work needs to be done to improve the system's reliability.

As shown in 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. This is because discourse is too complex and ill defined to easily 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 stand-alone system.

5 Conclusions

We have presented a discourse parser and evaluated it using the RST corpus. The presented discourse parser is divided into two levels: sentence-level and text-level. The experiment showed 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 were generated by combining the hypothesized discourse relations among non-overlapped text spans. We concentrated on solving the combinatorial explosion in searching for discourse trees. The constraints of textual adjacency and textual organization, and a beam search were applied to find the best-quality trees in a search space that is much smaller than the one given by Marcu (2000). The experiment on documents from the RST corpus showed that the proposed approach could produce reasonable results compared to human annotator agreements. To improve the system performance, future work includes refining the segmentation rules and improving criteria to select optimal paths in the beam search. We would also like to integrate a syntactic parser to this system. We hope this research will aid the development of text processing such as text summarization and text generation.

References

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski 2002. Building a Discourse-Tagged Corpus in the Framework of Rhetorical Structure Theory. In *Current Directions in Discourse and Dialogue*, Kluwer Academic Publishers.

Katherine Forbes, Eleni Miltsakaki, Rashmi Prasad, Anoop Sarkar, Aravind Joshi and Bonnie Webber 2003. D-LTAG System: Discourse Parsing with a Lexicalized Tree-Adjoining Grammar. *Journal of Logic, Language and Information*, 12(3), 261-279.

Edward Hovy 1993. Automated Discourse Generation Using Discourse Structure Relations. *Artificial Intelligence*, 63: 341-386.

Huong T. Le and Geetha Abeysinghe 2003. *A Study to Improve the Efficiency of a Discourse Parsing System.* In Proc of CICLing-03, 104-117.

Diane Litman and Julia Hirschberg 1990. *Disambiguating cue phrases in text and speech.* In Proc of COLING-90. Vol 2: 251-256.

William Mann and Sandra Thompson 1988. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text*, 8(3): 243-281.

Daniel Marcu 2000. *The theory and practice of discourse parsing and summarization.* MIT Press, Cambridge, Massachusetts, London, England.

Livia Polanyi, Chris Culy, Gian Lorenzo Thione and David Ahn 2004. *A Rule Based Approach to Discourse Parsing.* In Proc of SigDial2004.

Gisela 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.

Lloyd Rutledge, Brian Bailey, Jacco van Ossenbruggen, Lynda Hardman, and Joost Geurts 2000. *Generating Presentation Constraints from Rhetorical Structure.* In Proc of HYPERTEXT 2000.

Deborah Schiffrin 1987. *Discourse markers.* Cambridge: Cambridge University Press.

Radu Soricut and Daniel Marcu 2003. *Sentence Level Discourse Parsing using Syntactic and Lexical Information.* In Proc of HLT-NAACL 2003.

Mark Torrance, and Nadjet Bouayad-Agha 2001. Rhetorical structure analysis as a method for understanding writing processes. In Proc of MAD 2001.