Dialogue and Discourse 1(3) (2010) 1-33 doi: 10.5087/dad.2010.003

HILDA: A Discourse Parser Using Support Vector Machine Classification*

Hugo Hernault HUGO@MI.CI.I.U-TOKYO.AC.JP
Graduate School of Information Science & Technology
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

Helmut Prendinger HELMUT@NII.AC.JP
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

David A. duVerle DAVE@KUICR.KYOTO-U.AC.JP
Bioinformatics Center, Institute for Chemical Research
Kyoto University
Gokasho, Uji, Kyoto 611-0011, Japan

Mitsuru Ishizuka ISHIZUKA@I.U-TOKYO.AC.JP
Graduate School of Information Science & Technology
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

Editor: Tim Paek

Abstract

Discourse structures have a central role in several computational tasks, such as question–answering or dialogue generation. In particular, the framework of the Rhetorical Structure Theory (RST) offers a sound formalism for hierarchical text organization. In this article, we present HILDA, an implemented discourse parser based on RST and Support Vector Machine (SVM) classification. SVM classifiers are trained and applied to discourse segmentation and relation labeling. By combining labeling with a greedy bottom-up tree building approach, we are able to create accurate discourse trees in linear time complexity with respect to the length of the input text. Importantly, our parser can parse entire texts, whereas the publicly available parser SPADE (Soricut and Marcu, 2003) is limited to sentence level analysis. HILDA outperforms other discourse parsers for tree structure construction and discourse relation labeling. For the discourse parsing task, our system reaches 78.3% of the performance level of human annotators. Compared to a state-of-the-art rule-based discourse parser, our system achieves an performance increase of 11.6%.

Keywords: Discourse Parser, Rhetorical Structure Theory, Support Vector Machines

1. Introduction

In the last twenty years, the study of discourse has received continuous attention from the Natural Language Processing community (Moore and Wiemer-Hastings, 2003). Discourse structure is fundamental to many text-based applications, such as question–answering (Chai and Jin, 2004) or

∗. This article is a significantly improved and extended version of duVerle and Prendinger (2009).

dialogue generation (Prendinger et al., 2007). Those applications require the availability of robust and efficient discourse parsers.

Several attempts have been made to create discourse parsers in the framework of the Rhetorical Structure Theory (Mann and Thompson, 1988), which is one of the most widely used theories of text organization. In RST, a text is first divided into non-overlapping text chunks, called *elementary discourse units* (abbreviated: EDUs). For instance, the following sentence, taken from the RST Discourse Treebank corpus (Carlson et al., 2001):

> *Farm lending was enacted to correct this problem by providing a reliable flow of lendable funds.*

can be segmented into EDUs as shown in Figure 1.

[Farm lending was enacted]^{1A} [to correct this problem]^{1B} [by providing a reliable flow of lendable funds.]^{1C} (wsj_{1131})

Figure 1: Segmentation of a sentence into EDUs

Next, consecutive EDUs are put in relation with each other, using a pre-defined set of rhetorical, or discourse, relations. The final goal of the discourse parser is to produce a tree structure as a representation of how all units of the text relate to each other.

Figure 2 shows two types of discourse relations in RST: hypotactic ('mononuclear') and paratactic ('multi-nuclear'). The left-hand tree is an example of a mononuclear relation. Here the two discourse units connected by the relation have different status, which is indicated by the direction of the arrow. The endpoint of the arrow denotes the 'nucleus' of the relation, whereas the unit at the other end is referred to as the 'satellite'. The nucleus is considered as more prominent than the satellite. For example, CONDITION is defined in Carlson et al. (2001) as: "In a CONDITION relation, the truth of the proposition associated with the nucleus is a consequence of the fulfillment of the condition in the satellite. The satellite presents a situation that is not realized." Other mononuclear discourse relations described in (Mann and Thompson, 1988; Carlson et al., 2001) include: BACKGROUND, CIRCUMSTANCE, ELABORATION and PURPOSE.

Paratactic ('multi-nuclear') relations, on the other hand, have no distinguished nucleus and the relation can connect an arbitrary number of discourse units. The relation to the right in Figure 2 is multi-nuclear. For example, LIST is defined as "[...] a multinuclear relation whose elements can be listed, but which are not in a comparison [...]" (Carlson et al., 2001). The following discourse relations are also multi-nuclear (see Mann and Thompson (1988); Carlson et al. (2001)): CONTRAST, DISJUNCTION, SEQUENCE, and TOPIC-COMMENT.

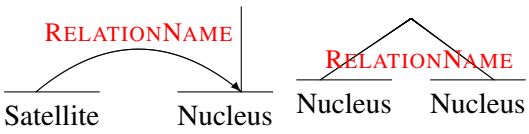


Figure 2: The two relation types in RST. Left: mononuclear; Right: multi-nuclear

Figure 3 shows the tree representation of the preceding example (see Figure 1).

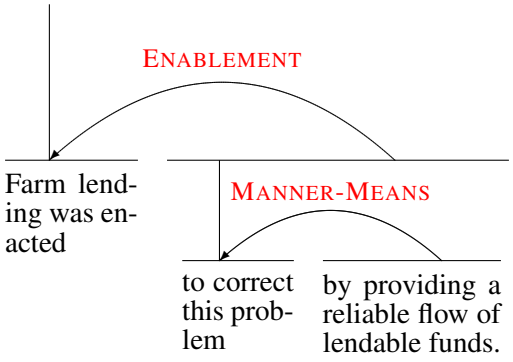


Figure 3: A simple discourse tree (wsj_{1131})

The tasks of discourse segmentation and relation labeling have been previously modeled using rule-based and statistical methods. In rule-based systems, rules manipulating syntactic and lexical information are defined manually. When applied to a text, these rules enable to detect the presence of EDU boundaries or select the discourse relations holding between EDUS. However, given the heterogeneity of all possible texts, a rule-based model requires the creation of a large number of rules.

Supervised learning techniques offer an interesting alternative. With those techniques, labeled instances are extracted from a large body of text. Classifiers are then trained to differentiate between EDU boundary and non-boundary words, or assign discourse relation labels to EDU pairs. The machine learning based approach for discourse parsing was greatly facilitated by the release of the RST Discourse Treebank (RST-DT) (Carlson et al., 2001), a corpus of RST-annotated snippets from the Wall Street Journal.

Previous supervised approaches were aimed at producing sentence level analysis (Soricut and Marcu, 2003) or at describing partially-implemented systems (Reitter, 2003b). By contrast, our work targets discourse structure at text level. Specifically, we created a fully-implemented, extensively-evaluated system.

In this article, we will present HILDA (HIgh-Level Discourse Analyzer), a text level discourse parser based on Support Vector Machine classification. In Section 2, we will report on available approaches using rule-based and statistical discourse segmentation and parsing. In Section 3, we will explain the two core tasks for an automated discourse parser (segmentation and relation labeling) and describe our working assumptions. Section 3.4 is dedicated to the set of lexical, syntactic and structural features used in our model. In Section 4, we will provide (1) a detailed evaluation of the discourse segmentation and relation labeling modules, (2) a full system evaluation, and (3) a comparison to other discourse parsers. Finally, in Section 5, we will conclude the article and discuss future work.

2. Related work

Since Marcu's first attempt at developing a rule-based discourse parser (Marcu, 2000), several algorithms for discourse parsing have been proposed, both statistical and rule-based. Each of them extracts a different set of features from the input, and demonstrates different run-time complexity. In this section we present the most notable ones, which also proved instrumental in identifying

useful features for our own algorithm. We start with discussing statistical approaches to discourse segmenting and parsing.

SPADE (Soricut and Marcu, 2003) is a sentence level discourse parser. Two probabilistic models are employed that use syntactic and lexical information to segment and to parse text. Issues of algorithmic complexity that made their original algorithm impractical on large instances (Marcu, 2000) are overcome by a dynamic programming approach. Although SPADE does not support full text discourse parsing, it demonstrates the correlation between syntactic and discourse information and their capability to identifying structure and relations empirically, particularly when no signaling cue-word is present (between 60% and 70% of the total (Taboada, 2006)). The authors' exploration of 'dominance sets' in lexicalized syntax trees has provided the basis for a set of lexico-syntactic features in our own algorithm. A key difference to our approach is we are able to process full text input, rather than individual sentences.

To the best of our knowledge, Reitter (2003b) presents the only method based exclusively on feature-rich supervised learning to produce text level discourse parse trees. His algorithm relies on training a set of Support Vector Machine classifiers to score and to label relations between spans. Although the author's suggestion of a bottom-up tree building algorithm using chart parsing style techniques has not been implemented yet, the results for raw instance classification provides a good intermediate benchmark for the evaluation of our own instance classifier.

More recently, Baldridge and Lascarides (2005) successfully implemented a probabilistic parser that uses headed trees to label discourse relations. The authors employed the more specific framework of Segmented Discourse Representation Theory (Asher and Lascarides, 2003) rather than RST, which they applied to texts in dialogue form.

In Sagae (2009), a discourse parser based on shift-reduced parsing is presented. The parser is trained on the RST-DT, employs transition algorithms for dependency and constituent trees, and uses its own syntax and dependency tagger. It brings noticeable improvements in accuracy and speed against Marcu's original chart parsing approach (Marcu, 2000). The parser also includes a discourse segmenter based on a binary classifier trained on lexico-syntactic features. The discourse segmenter performs better than the one presented in Reitter (2003b) and is significantly faster. Importantly, compared to Soricut and Marcu (2003), the proposed system is able to create discourse structures for full texts, not only sentences. For discourse segmentation, the author reports an F-score of 86.7% and an F-score of 44.5% for creating text level discourse trees.

Subba and Di Eugenio (2009) propose a discourse parser based on Inductive Logic Programming, a first-order logic approach for learning discourse relations. In their system, besides traditional syntactic and discourse cue information, rules are learnt based on semantic information from WordNet (Fellbaum, 1998), similarity measures, structural properties between text segments, as well as compositional semantics. Then, a shift-reduce parsing model is employed to produce discourse structures. The authors employ a custom corpus of instructional texts, manually segmented into discourse units, and annotated with a custom set of 26 discourse relations. For sentence-level relation labeling, their system reaches an F-score of 63%, while the F-score for text level discourse trees is 35.44%.

A statistical discourse segmenter based on artificial neural networks is presented in Subba and Di Eugenio (2007). The system employs a multi-layer perceptron model and is trained on the RST-DT using syntactic and lexical information, in particular discourse cues. Bagging (Breiman, 1996) is applied to reduce over-fitting. The performance of this segmenter is comparable to the one

employed in Soricut and Marcu (2003), with an F-score of 84.4% (86% when using perfect parse trees).

Next we turn to rule-based approaches to segmentation and discourse parsing. Le Thanh et al. (2004b) propose a multi-step algorithm to segment text and organize its spans into trees for each successive level of text organization: sentence level, paragraph level, and text level. Within each level, the authors explore a search space consisting of all valid trees fitting the 'sequentiality principle' (i.e., two spans connected in the tree are adjacent in the original text, see Section 3.2). The multi-level nature of their algorithm mitigates the combinatorial explosion effect. However, at the text level, the algorithm has to score a large number of trees to extract the best candidate—despite the use of beam search as a method to explore the solution space. Hence, this approach is impractical for large input text. Furthermore, the assumption of strong tree-consistency (the adjacency hypothesis) at each organizational level is not supported in the RST corpus, where a majority of documents contain organizational blocks that do not map to valid subtrees.

Tofiloski et al. (2009) argue that using rules has certain advantages over automatic learning methods and present a rule-based discourse segmenter. Their proposed model does not depend on a specific training corpus and supports high-precision segmentation by inserting fewer but 'quality' boundaries. However, segmentation is done in a manner different from the segmentation guidelines used in the RST-DT corpus (Carlson et al., 2001). First, the authors only create EDUs that contain verbs. They try to capture specific relations (e.g., CONDITION, PURPOSE) and avoid less informative relations, such as ELABORATION. Further, complement clauses are not placed in independent units in order not to break NP constituency, and avoid SAME-UNIT relations. For instance, *"He said that"* is not considered an independent EDU. In this article, we will not enter the discussion of what constitutes the best segmentation guidelines, and simply take the RST-DT corpus as the basis for segmentation by supervised methods.

3. Approach

3.1 The RST Discourse Treebank

As previously mentioned, RST-DT is a large corpus of documents annotated with EDU segmentation and full text level rhetorical structure (Carlson et al., 2001). The corpus was released in 2002 and contains 385 articles transcribed from the *Wall Street Journal*. These articles are a subset of the Penn Treebank corpus of English (Marcus et al., 1993). The corpus allows us to both train and evaluate the performance of our algorithm on a large number of documents of varied lengths.

Although the original RST set is composed of 24 discourse relations (Mann and Thompson, 1988), this list has evolved to fit the type of application and expressive power required by each researcher. In the RST-DT, relation annotation is done using a set of 78 rhetorical relations, which enables a high level of expressivity. These relations are divided in two categories: 53 mononuclear relations and 25 multi-nuclear relations.

3.2 Working assumptions

An essential characteristic of RST is that "a left-to-right reading of the terminal frontier of a discourse tree associated with a text corresponds to the span of text analyzed, in the same linear order." (Marcu, 2000). This 'sequentiality principle' guarantees that only consecutive spans of a text can be put into relation, which dramatically reduces the size of the solution space (see Section 3.3).

When building discourse parsers, researchers have opted to use small relation sets, which makes the problem of assigning relations easier. For instance, Marcu (2000) used 15 relations, while Le Thanh et al. (2004b) used 14 relations. Although the rich 78-relations set of the RST-DT enables a high level of expressivity, which is useful in the case of linguistic studies, it has unnecessary precision for most text analysis applications. Furthermore, from a machine learning perspective, working with a smaller set of relations improves the computational properties of the problem. A problem with fewer classes guarantees a better separability between the different classes, as it abstracts from the ambiguities inherent to fine-grained relations. We adopt a similar strategy and work with the 18 relations defined in Carlson et al. (2001), and previously used by Soricut and Marcu (2003).

In this reduced set, the original RST-DT relations are partitioned into 16 categories, which correspond to (merged) relations (called "classes" in Carlson et al. (2001)), depending on their rhetorical similarity.[1] For instance, the semantically similar PROBLEM-SOLUTION, QUESTION-ANSWER, STATEMENT-RESPONSE, TOPIC-COMMENT and COMMENT-TOPIC relations, are merged to TOPIC-COMMENT. Two extra relations are used for helping the structuring of the text, TEXTUAL-ORGANIZATION and SAME-UNIT.

In RST, multi-nuclear relations can take an arbitrary number of arguments. For instance, the LIST relation can connect any number of spans, leading to n-ary tree representations. In order to maintain compatibility with our SVM classification approach, we have to convert n-ary trees to binary trees. This conversion can be done trivially, by consecutively nesting the arguments of multi-nuclear relations, in the fashion shown in Figure 4.

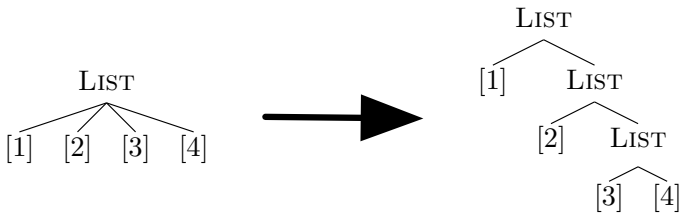


Figure 4: Binarization of multi-nuclear relations

It is interesting to note that the reverse transformation is possible most of the time. However, a loss of reversibility occurs when two similar multi-nuclear relations are nested consecutively (for instance, two LISTs). However, these cases occur rarely in practice—only 15 times in the 380 documents of our corpus.

3.3 Outline of discourse segmentation and discourse relation labeling

Figure 5 shows the basic workflow of the HILDA parser (details of the workflow are provided in Appendix A).

1. For a complete list of the relations used, see Appendix B.

- A text is first segmented into EDUs.
- Then the relation labeling step evaluates which relations are likely to hold between consecutive EDUs. The two EDUs which are most probably connected by a rhetorical relation are merged into a rhetorical structure tree of two EDUs.
- Next, we go back to the labeling step to re-evaluate which relations are the most likely to hold between spans (rhetorical structure trees of any size, including atomical EDUs).
- This procedure is repeated until all spans are merged.

The outlined method enables us to find a good approximation of 'perfect' discourse trees (i.e., the ones created by human annotators) in linear time complexity with respect to the length of the input text.

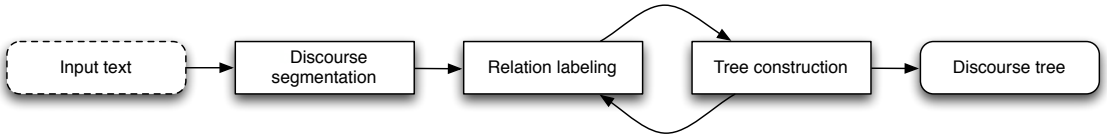


Figure 5: Basic parser workflow

The tasks of discourse segmentation and relation labeling are modeled as supervised classification tasks. We chose to use Support Vector Machines (SVM) (Vapnik, 1995). SVMs are a set of maximum-margin classifiers, i.e., they minimize the classification error and maximize the geometric margin. They have been used extensively in domains ranging from bio-informatics to natural language processing, and are considered as a state-of-the-art classifier for many practical tasks.

In discourse segmentation, the problem is to assign each word of the input text an observation category $c \in \{0, 1\}$, where 0 indicates that a word is *not a boundary*, and 1 indicates that it *is a boundary*. For example, in the snippet from Figure 1, the words *said*, *quarter* and *yesterday* are assigned 1, whereas all other words belong to category 0. Hence, we train a binary classifier $Seg : \mathcal{W} \rightarrow \{0, 1\}$, where $\mathcal{W}$ is the set of all words occurring in the input text, denoted by $InputText$. After the segmentation step, we obtain a list of EDUs, $E = \langle e_1, e_2, \ldots \rangle$. The concatenation of these EDUs, from left to right, gives back $InputText$.

For relation labeling and tree construction, we apply the following method. Given two consecutive spans (EDUs or subtrees), we determine (1) the likelihood of a direct structural relation, (2) the probabilities for the label of the relation with nuclearity of the spans. A full rhetorical structure tree is produced by applying this mechanism repeatedly in a straightforward bottom-up fashion.

Definition 1 (discourse relation set) *A discourse relation set $\mathcal{R}$ is defined as a set $\mathcal{R} = \{R_1, R_2, \ldots R_i, \ldots, R_n\}$, such that $\forall i$, $R_i = \langle RR_i, Left_i, Right_i \rangle$, whereby:*

- $RR_i \in \{$ATTRIBUTION, CAUSE, ELABORATION, LIST, $\ldots\}$ *(see Appendix B for a full list of relations considered in HILDA);*
- $Left_i,\ Right_i \in \{Nucleus,\ Satellite\}$*;*

- *all R_i triplets are valid, i.e., given a rhetorical relation RR_i, the left and right nuclearities $Left_i$ and $Right_i$ are allowed nuclearity options. Our working rhetorical relation set and allowed nuclearity options are defined in Appendix B. We employ 18 rhetorical relations which, when nuclearized, give a total of 41 possible R_i triplets (the cardinal number of $\mathcal{R}$).*

Definition 2 (valid RS-tree) *A RS-tree T is valid if it satisfies the following properties:*

1. *All leaf nodes of T are* EDUs;[2]
2. *All non-leaf nodes of T are tagged with a discourse relation $R_i \in \mathcal{R}$.*

$\mathcal{T} = \{T_1, T_2, \ldots T_i, \ldots\}$ denotes the set of all valid RS-trees. In our algorithm, RS-trees are implemented using binary tree structures.
To obtain a good classification accuracy, we train two separate classifiers:

- A binary classifier for structure labeling, i.e., existence of a rhetorical relation between two valid RS-trees: $Struct : \mathcal{T} \times \mathcal{T} \rightarrow \{0, 1\}$.
- A multi-class classifier for rhetorical relation and nuclearity labeling: $Label : \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{R}$. $Label$ returns predicted relation label and nuclearities.

The detailed algorithm for tree construction is provided in Figure 6. We first create a list containing all EDUs of the input text, in left-to-right reading order. The binary classifier $Struct$ is then applied to all pairs of consecutive elements from this list, in order to determine the probability of a structural relation between consecutive EDUs. Next we select the pair with the highest probability and apply $Label$ to it, in order to determine the relation's label and nuclearity. We can now remove the two selected EDUs from the list, and replace them by a newly created RS-tree labeled by the estimated relation label and nuclearity, whose left and right children are the selected EDUs. We update the probabilities of structural relations holding between the newly created tree and adjacent elements in the list, using $Struct$. Subsequently, we repeat the process until the list contains only one element, which is our final discourse tree, built bottom-to-top. In doing so, we have adopted a 'greedy' approach, i.e., at each iteration the two spans most likely connected by a relation have been merged. Hence, our algorithm, which makes locally-optimal choices, performs in time complexity of $\mathcal{O}(n)$.

3.4 Features

3.4.1 DISCOURSE SEGMENTATION

The first step of the parser is to segment the text into units. For this task, we use a combination of syntactic and lexical features: words, POS tags, and lexical heads. In particular, we use the lexico-syntactic features of Soricut and Marcu (2003), which were found good indicators for the presence of EDU boundaries.

Figure 9 shows the parse tree of the sentence "*Farm lending was enacted to correct this problem by providing a reliable flow of lendable funds.*" (see also Figure 1). Here, lexical heads are generated using projection rules from Magerman (1995) and indicated between brackets. For a word w, we look at its highest ancestor in the parse tree with a lexical head equal to w, and with a right-sibling.

2. Note that a single EDU also constitutes a valid RS-tree.

Require: $E = \langle e_1, e_2 \dots \rangle$, list of all the text's EDU
Ensure: $FinalTree$ is a valid RS-tree for $InputText$
  $L \leftarrow E$
  for all (l_i, l_{i+1}) in L **do**
    $Scores[i] \leftarrow Struct(l_i, l_{i+1})^a$
  end for
  while $|L| > 1$ **do**
    $i \leftarrow argmax(Scores)$
    $NewLabel \leftarrow Label(l_i, l_{i+1})$
    $NewSubTree \leftarrow CreateTree(l_i, l_{i+1}, NewLabel)^b$
    $Scores[i-1] \leftarrow Struct(l_{i-1}, NewSubTree)$
    $Scores[i+2] \leftarrow Struct(NewSubTree, l_{i+2})$
    $delete(Scores[i])$
    $delete(Scores[i+1])$
    $L \leftarrow [l_0, \dots, l_{i-1}, NewSubTree, l_{i+2}, \dots]$
  end while
  $FinalTree \leftarrow l_0$
  return $FinalTree$

a. l_i denotes the i-th element of list L.
b. $CreateTree$ is a function that takes two RS-trees T and T', a relation $R \in \mathcal{R}$, and returns the RS-tree whose left child is T, whose right child is T', and which is tagged with relation R.

Figure 6: Bottom-up tree construction

This highest-ancestor node is called N_w. Then, we call its parent N_p, and its right-sibling N_r. For instance, when applying this process to the word "enacted", we obtain $N_w =$ VBN(enacted), $N_p =$ VP(enacted), $N_r =$ S(correct).

We can now define the *contextual features of the word at position* i in the text. It is the set composed of the word w_i, its POS, as well as the POS and lexical heads of N_{wi}, N_{p_i}, and N_{ri}. Next, the features for position i in the text are created by concatenating the contextual features at positions $i-2$, $i-1$, and i. Those positions were determined empirically to give the maximum contextual coverage.

3.4.2 RELATION LABELING

For relation labeling, several shallow lexical and syntactic features are taken into account, including features of textual organization, lexical features, 'dominance sets' (Soricut and Marcu, 2003), and structural features.

The first type of features we incorporate is related to textual organization. A number of previous efforts (Marcu, 1996; Soricut and Marcu, 2003) have shown that there is a strong correlation between different organizational levels of textual units and subtrees of the RS-tree, both at the sentence level and at the paragraph level. Hence, the following features provide valuable high-level cues for the task of scoring span relation priority (classifier $Struct$). As hypothesized by Reitter (2003b), there is a correlation between span length and type of rhetorical relations. For instance, the satellite of a CONTRAST relation tends to be shorter than its nucleus.

The organizational features taken into account in our system are presented in Table 1. In this table, all subtree-specific features, which are symmetrically extracted from both left and right candidate spans, are suffixed by "S[pan]". The other features, calculated as a function of the two subtrees considered as a pair, are indicated by "F[ull]".

Table 1: Features encoding textual organization

Feature name	**Scope**
Belong to same sentence	F
Belong to same paragraph	F
Number of paragraph boundaries	S
Number of sentence boundaries	S
Length in tokens	S
Length in EDUs	S
Distance to beginning of sentence in tokens	S
Size of span over sentence in EDUs	S
Size of span over sentence in tokens	S
Size of both spans over sentence in tokens	F
Distance to beginning of sentence in EDUs	S
Distance to beginning of text in tokens	S
Distance to end of sentence in tokens	S

Discourse cues, such as *because*, *however*, etc are another type of feature that frequently indicate the presence of discourse relations. Instead of working with a pre-defined dictionary of discourse cues, we chose to build an empirical n-gram dictionary from the training corpus, ranked by frequency, in order to keep it to a reasonable size. This method has the advantage of capturing non-lexical signals, such as punctuation and paragraph boundaries. We encode the prefix and suffix of each span as ordered 3-grams, which were found to give the best signal-to-noise ratio (see Manning and Schütze, 1999). Encoding the beginning and end of a span is motivated by the hypothesis that the most meaningful rhetorical signals are found at the edges of the span (Schilder, 2002). For instance, for the span in Figure 3 consisting of units $1B$ and $1C$ connected by a MANNER-MEANS relation, the beginning and end-of-span 3-grams we encode in our features $(to, correct, this)$ and $(of, lendable, funds)$, respectively. In total, our 3-gram dictionary is of size 12,000, and encoding prefix and suffix raise the encoding size to $2^2 \times |\text{3-gram dictionary}| = 48,000$.

This approach was validated by comparing it to results obtained from using a fixed dictionary of 300 cue-phrases instead (Oberlander and Moore, 1999). Performance was found to be lower when using only the cue-phrase dictionary, and marginally lower when using both together. To improve discourse cue detection and to make it less dependent on lexical information, we complement it with shallow syntactic information, by encoding POS tags for the prefix and suffix of each span. The Penn Treebank contains $N_{\text{POS}} = 384$ tags, hence it requires $2 \times 3 \times N_{\text{POS}} = 2304$ additional dimensions.

To reflect the way relations attach in RS-trees, we use the notion of *dominance sets* as defined in Soricut and Marcu (2003). A dominance set is formed by consecutive EDUs of a sentence that are connected in the lexicalized syntax tree by a head node (the dominating node) and an attachment node (the dominated node). For instance, the sentence of Figure 1 contains three EDUs, which

are connected by two rhetorical relations, ENABLEMENT and MANNER-MEANS. This leads to two possible tree constructions (see Figure 8), in which EDU $1B$ is connected either to $1A$ or to $1C$. In the left-hand construction, $1A$ and $1B$ are connected first, then the subtree they span is connected to $1C$. In the right-hand construction, $1B$ and $1C$ are connected first, then the subtree they span is connected to $1A$. It is possible to infer the correct logical nesting order by studying the associated syntax tree, and observing the sub-trees spanned by each EDU. In Figure 9, looking at the frontiers between dominating nodes (diamond-shaped) and dominated nodes (oval-shaped) indicates the correct dominance order, $1A > 1B > 1C$. Thus, $1B$ should be attached to $1C$, and the correct tree in Figure 8 is the right-hand one. Soricut and Marcu (2003) also note that only taking the POS tags of the dominance nodes into account is too general. Then, in order to augment information about the lexico-syntactic context, lexical heads of those nodes are also taken into account.

For approximating Soricut and Marcu's definition of dominance sets, we incorporate the lexico-syntactic and structural features presented in Table 2 into our model. It is worth noting that most of these features apply only when both spans belong to the same sentence. Otherwise, the parse trees of spans are disjunct, and the notion of dominance does not apply. In this case, the features receive default values during feature extraction.

Table 2: Features encoding dominance sets

Feature name	**Scope**
Distance to root of the syntax tree	S
Distance to common ancestor in the syntax tree	S
Delta of distances to common ancestor	F
Dominating node's lexical head in span	S
Common ancestor's POS tag	F
Common ancestor's lexical head	F
Dominating node's POS tag	F
Dominating node's lexical head	F
Dominated node's POS tag	F
Dominated node's lexical head	F
Dominated node's sibling's POS tag	F
Dominated node's sibling's lexical head	F
Relative position of lexical head in sentence	S

The *strong compositionality criterion* was found by Marcu through empirical analysis of large RST trees: "[...] whenever two large text spans are connected through a rhetorical relation, that rhetorical relation holds also between the most important parts *(i.e. the nuclei)* of the constituent spans." (see Marcu (1996, 2)). This criterion, he argues, provides a good indicator of validity in the case of text level trees where relations can connect large spans of text beyond the sentence level. The idea of strong compositionality is implemented by replicating the set of lexical and syntactic features of Table 2, which are extracted from the 'most important' EDU, called *main constituent*, of each span. The main constituent of the left span is found by recursively selecting the right-most nucleus, starting from the top relation. For the right span, the left-most nucleus is selected instead. We chose the leftmost and rightmost nuclei, respectively, so that the EDUs considered are as close

as possible to each other. Marcu (1996) recommends selecting the union of all nuclei in each span. However, the restriction we impose is necessary to accommodate the fixed length nature of our feature vectors.

Finally, to guide decisions regarding high-level relation classification on large spans, we encode the rhetorical structure of a span into the feature vector. We (informally) observed some level of correlation between relations at different levels of the tree throughout the corpus. This is trivially the case for n-ary relations such LIST which have been binarized in our representation; the presence of several LIST relations in right-most nodes of a subtree greatly increases the chance that the parent relation might be a LIST itself. We encode this structure as a breadth-first, flat list representation of the binary tree, whereby each element of the list is split over one of the 41 potential class labels. As the feature vectors have a fixed length, we set an arbitrary limitation on the depth encoded. This increases the encoding size by $2 \times |\mathcal{R}| \times 2^d = 41 \times 2^{d+1}$, where d is the maximal depth considered. Experiments have shown optimal performances for $2 \leq d \leq 3$ (we selected $d = 3$) with sharp decreases for values of $d > 4$. This result can be explained by the excessively noisy data brought about by an exponentially growing set of features.

4. Experiments

4.1 Discourse segmentation

We first measure the performance of the Seg classifier. After feature extraction from the 341 training documents and 38 test documents of the RST-DT, we obtain 177,633 training vectors and 21,667 test vectors for this task. Because a sentence typically contains few EDU boundaries, our dataset is skewed, with approximately 89% negative training examples and 11% positive examples. Our segmenter is evaluated with respect to two types of competing systems. First, we measure the segmentation result when using parse trees from the Penn Treebank (Marcus et al., 1993) as our gold standard. Second, as a practical evaluation, we compare the performance when using parse trees generated respectively by the Stanford parser[3] (Klein and Manning, 2003) and by the Charniak parser[4] (Charniak, 2000).

Evaluation is done on the test subset of the RST-DT. In our experiments, we use the same measure as Soricut and Marcu (2003) and Tofiloski et al. (2009), i.e., we only measure the score on boundaries inside sentences. This avoids artificially boosting the performance by including obvious end-of-sentence or end-of-paragraph boundaries. The radial basis function (RBF) (Vapnik, 1995) kernel is selected, and parameter estimation is done using grid search with 5-fold cross validation (Staelin, 2003). In practice, we did not observe any problem related to the class imbalance in the training set with these parameters. The performance of our and other available segmenters is shown in Table 3.

NNDS refers to the Neural-Networks Discourse Segmenter (Subba and Di Eugenio, 2007), SPADE-Seg is the segmentation module of SPADE, described in Soricut and Marcu (2003), and SAG refers to the segmenter included in the discourse parser of Sagae (2009). Table 3 (top row) shows that HILDA-Seg significantly outperforms other discourse segmenters. With gold standard trees, SPADE-Seg and NNDS yield F-scores of 84.7% and 86%, respectively, versus 95% for HILDA-Seg. The measure of the human annotators' agreement for the segmentation task has been

3. http://nlp.stanford.edu/software/lex-parser.shtml
4. ftp://ftp.cs.brown.edu/pub/nlparser/

Table 3: Performance comparison with other segmenters

System	Trees	Precision	Recall	F-score
SPADE-Seg	Penn	84.1	85.4	84.7
NNDS	Penn	85.5	86.6	86.0
HILDA-Seg	Penn	**95.5**	**94.5**	**95.0**
SPADE-Seg	Charniak	83.5	82.7	83.1
NNDS	Charniak	83.9	84.8	84.4
HILDA-Seg	Charniak	**94.7**	**93.4**	**94.0**
SAG	SAG	87.4	86.0	86.7
HILDA-Seg	Stanford	94.5	93.1	93.8
Human agreement	–	98.5	98.2	98.3

calculated in Soricut and Marcu (2003), with a F-score of 98.3%. Using perfect parse trees, HILDA-Seg reaches 96.6% of the annotators' level. When the Stanford parse trees are selected, 95.4% of this level is reached. The current state-of-the-art discourse segmenter, SAG, in which a custom syntax parser is employed, has an F-score of 86.7% for this task. When using Stanford parse trees, our segmenter thus provides a 8.2% performance increase compared to SAG (9.6% when using perfect trees).

In Table 3, we decided not to include the results of the rule-based segmenters of Le Thanh et al. (2004a) and Tofiloski et al. (2009), for several reasons. First, Le Thanh et al. (2004a) report their results using a 'softer' metric, in which end-of-sentence boundaries are taken into account. The authors used Penn Treebank parse trees, and after evaluation on 8 texts of the RST-DT, obtain a precision value of 81.4% and a recall value of 79.2%. Finally, the results of Tofiloski et al. (2009) cannot be compared directly to ours, as different segmentation guidelines were used. The authors report a precision value of 89% and a recall value of 86% when using Charniak parse trees, and a precision value of 82% and a recall value of 86% when using Stanford trees. Moreover, these scores were measured on three texts of the RST-DT only, which makes comparison even more difficult.

To investigate which features contributed to the performance of the discourse segmenter, we run several experiments that measure the segmentation performance when training with different combinations of features, taken at different relative positions. For each experiment, we use parse trees from the Stanford parser. The results are indicated in Table 4. Position (0) and $(-1, 0)$ indicate that, when extracting features for the word at position i in the text, we only used contextual features from position i, and the concatenation of contextual features from positions $i-1$ and i, respectively.

We observe that, for position (0), using as features the current word w and its part of speech POS(w) only, we obtain a precision of 82.7%, but a comparatively lower recall of 70.8%. However, when using as features the three nodes N_w, N_p, N_r of the parse tree only (see 3.4.1), the precision is slightly lower (82%) but the recall higher than the previous case, at 78.4%. When using the combination of all these features, we obtain a precision of 81.9% and a recall of 79.4%. With this set of relevant features defined, we now increase the coverage by also including features extracted from the previous word in our feature set. With these features, now taken from positions

$(-1, 0)$, we notice a sharp F-score increase of 14.5%. In particular, precision reaches 93.3% and recall 91.5%. Finally, we add the contextual features for the word at position $i - 2$ to the feature set. The F-score increase in this case is much smaller (1.5%). We now attain a precision of 94.5% and a recall of 93.1%. From this point forward, adding features from anterior words did not improve performance, although it significantly increased training time. Finally, we did not notice any significant performance increase when including contextual features for words located ahead of the current position.

Table 4: Segmentation performance with different sets of features

Positions	**Features**	**P**	**R**	**F**
(0)	w	80.7	69.4	74.6
(0)	w, POS(w)	82.7	70.8	76.3
(0)	N_w, N_p, N_r	82.0	78.4	80.2
(0)	w, POS(w), N_w, N_p, N_r	81.9	79.4	80.7
$(-1, 0)$	w, POS(w), N_w, N_p, N_r	93.3	91.5	92.4
$(-2, -1, 0)$	w, POS(w), N_w, N_p, N_r	94.5	93.1	93.8

4.2 Relation labeling

Although our final goal is to achieve good performance on the entire tree-building task, a useful intermediate evaluation of our system can be conducted by measuring the raw performance of our SVM classifiers. The binary classifier $Struct$ is trained on 52,683 instances (split approximately between $1/3$ positive and $2/3$ negative examples), extracted from 341 documents, and tested on 8,558 instances extracted from 38 documents. The feature space has a dimension of 136,987.

The other classifier, $Label$, is trained on 17,742 instances (labeled across 41 classes) and tested on 2,887 instances, the same feature vector dimension as $Struct$. The optimal training parameters for each kernel function are obtained through automated grid search with 5-fold cross-validation (Staelin, 2003). Table 5 shows the training time and accuracy for these experiments, using various software and selected different kernels, for both $Struct$ and $Label$, as well as a comparison to the results of Reitter (2003a). For $Struct$ we used the regression mode of the SVM software to obtain probabilistic output. Note that in Table 5, the results for $Struct$ and $Label$ are independent, as they indicate performance for cross-validation.

Table 5: SVM Classifiers performances – LL: Liblinear[a], SVML: SVM Light[b], SVMM: SVM Multiclass[c], LS: libsvm[d]

Classifier	*Struct*			*Label*		Reitter
Kernel	Linear	Polyn.	RBF	Linear	RBF	RBF
Software	LL	SVML	SVML	SVMM	LS	SVML
Training time	21.4s	5m53s	12m	15m	23m	216m
Accuracy	82.2	**85.0**	82.9	65.8	**66.8**	61.0

[a] http://www.csie.ntu.edu.tw/~cjlin/liblinear/
[b] http://svmlight.joachims.org/
[c] http://svmlight.joachims.org/svm_multiclass.html
[d] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Despite their simplicity, the noticeably good performances of linear kernel methods in the results presented in Table 5, as compared to the more complex polynomial and RBF kernels, indicate that our data separates fairly well linearly. This is a commonly observed effect of high-dimensional input such as ours (over 100,000 features) (Chen et al., 2007).

Reitter (2003a) provides a baseline for absolute comparison on the multi-label classification task for a similar classifier that assumes perfect segmentation of the input, as ours does. Reitter's accuracy results of 61% for a smaller set of training instances (7976 instances from 240 documents compared to 17,742 instances in our case) with considerably less classes (16 rhetorical relation labels with no nuclearity, as opposed to our 41 nuclearized relation classes) seems to indicate that this sub-component of our system with an accuracy of 66.8% performs well.

In HILDA, we select a linear kernel for $Struct$ and an optimally parameterized RBF kernel for Lab, considering performance and runtime complexity. We use modified versions of the $liblinear$ (optimized for large-instances) and *libsvm* packages. All further evaluation results reported in this article were conducted using these kernels.

To give a more precise idea of the classifiers' performance, we look into the results for each class, when trained on RST-DT's standard training set and evaluated on the standard test set. Table 6 indicates the results for $Struct$, while Table 7 indicates the results for $Label$. In Table 6, we notice in particular that the performance for the class '$+1$', corresponding to the presence of a structural relation between to RS-trees is lower (F-score of 73.3%) than for the absence of a relation (F-score of 86.7%).

Because RST-DT's test set is relatively small (38 documents), and as certain discourse relations occur rarely in natural language texts, several of $Label$'s classes are not present in the test set and thus not presented in Table 7. We observe that performance varies widely across classes. For instance, ATTRIBUTION is very well detected, with F-scores close to 95%, whereas CAUSE[N][S] has a very low F-score of 3.9%. Here, the average F-score is 47.7%.

As we employed a linear kernel in $Struct$, it is possible to identify which features were useful, by looking at the weight assigned by the classifier to each feature in the SVM model file. Table 8 contains the 25 features with the highest weights in absolute value. These features are thus useful for detecting the presence of a discourse relation between two RS trees. In particular, we observe

Table 6: Class-specific performance for $Struct$, evaluated on RST-DT's test set

SVM Class	Precision	Recall	F-score
$+1$	74.0	72.7	73.3
-1	86.3	87.1	86.7

Table 7: Class-specific performance for $Label$, evaluated on RST-DT's test set

SVM Class	Precision	Recall	F-score
ATTRIBUTION[N][S]	93.6	96.2	94.9
ATTRIBUTION[S][N]	95.7	93.7	94.7
BACKGROUND[N][S]	47.8	41.5	44.4
BACKGROUND[S][N]	38.7	20.7	27.0
CAUSE[N][S]	33.3	2.1	3.9
COMPARISON[N][S]	50.0	5.9	10.5
CONDITION[N][S]	100.0	47.8	64.7
CONDITION[S][N]	85.7	72.0	78.3
CONTRAST[N][N]	31.1	21.9	25.7
CONTRAST[N][S]	50.0	20.8	29.4
CONTRAST[S][N]	51.1	39.7	44.7
ELABORATION[N][S]	58.1	94.5	72.0
ENABLEMENT[N][S]	61.9	59.1	60.5
ENABLEMENT[S][N]	50.0	50.0	50.0
EXPLANATION[N][S]	66.7	16.8	26.9
JOINT[N][N]	62.2	55.2	58.5
MANNER-MEANS[N][S]	53.8	29.2	37.8
SAME-UNIT[N][N]	78.8	96.9	86.9
SUMMARY[N][S]	100.0	31.2	47.6
TEMPORAL[N][N]	83.3	11.9	20.8
TEMPORAL[N][S]	85.7	24.0	37.5
TEXTUAL-ORGANIZATION[N][N]	33.3	22.2	26.7
TOPIC-CHANGE[N][N]	83.3	38.5	52.6

that the most heavily-weighted feature is the binary feature indicating if the two input spans belong to the same sentence. Among these features, we notice several measures of distance, such as the distance of the left span to the sentence beginning, the relative size of the left and right span's EDUs over the sentence length, or the number of paragraphs in the right span. We also find several features related to dominance sets, such as features encoding the dominated and dominating node's POS tags and lexical heads.

Table 8: Top 25 SVM weights of the linear kernel for *Struct*.

Feature	**Weight**
Both spans belong to the same sentence	4.118836
Size of span over sentence in EDUs	3.582545
Distance of the left span to beginning of sentence in EDUs	-3.437157
Common ancestor's POS tag is 'PRN'	-2.911269
Dominating node's lexical head is 'which'	-2.668148
POS tag of the right span's last token is '.'	2.636921
Size of left span over sentence in tokens	-2.341654
Size of both spans over sentence in tokens	-2.222655
Left and right span belong to the same sentence	-2.217709
POS tag of the left span's last token is '.'	2.170483
Number of paragraphs in the right span	2.123776
Relative position of lexical head in sentence	2.084602
POS tag of the left span's last token is '"'	2.051767
Dominated node's sibling's POS tag is 'VBG'	-1.875610
POS tag of the right span's first token is 'TO'	-1.869705
Distance of the right span from the sentence beginning	-1.839178
POS tag of the right span's last token is ','	-1.814842
Top lexical head of the left span is 'which'	1.777975
Top lexical head of the sentence belongs to the right span	-1.774898
Top lexical head of the left span is 'of'	-1.706626
Dominating node's lexical head is 'set'	-1.670822
Dominated node's lexical head is 'ideas'	-1.669270
Dominating node's POS tag is 'NN'	-1.650378
Dominated node's sibling's POS tag is 'VBN'	1.628170
Dominating node's POS tag is 'VBN'	-1.617607

4.3 Full system performance

A measure of the performance of our full system is realized by comparing the structure and labeling of the RS-tree produced by our algorithm to that obtained through manual annotation (our gold standard). Here, we use the PARSEVAL metrics (Black et al., 1991) that defines the performance metrics (precision, recall, F-score) of a parsing system. For comparability, the tree constituents used in the method are enumerated using Marcu's formalism (see Marcu, 2000, 143–144), which assigns relation labels to the child nodes of the relation instead of the parent node, thus making it easier to represent *n*-ary relations. In practice, this formalism yields lower scores than our own, likely because of the greater emphasis placed on tree structure accuracy over relation labeling.

HILDA is evaluated based on three experiments.

1. Perfectly segmented input taken from the RST-DT (indicated as 'Manual' in the following tables). In this case, we use parse trees from the Penn Treebank in all steps of the parsing.

2. The system is run using the output of SPADE's segmenter (SPADE-Seg), in which case Charniak's parser is used for generating parse trees.

3. The system is run with our own segmenter (HILDA-Seg), presented in Section 4.1. In this case, we use parse trees from the Stanford parser.

The first measure gives us a good estimate of our system's optimal performance (given optimal input), while the others provide a concrete performance evaluation under practical conditions. Our measurements are taken based on the standard test subset of 41 files provided by the RST-DT corpus. For each experiment, parse trees are evaluated on four successive, increasingly complex criteria. First, on the blank tree structure ('S'), then on the tree structure with nuclearity indication ('N'), then the tree structure with rhetorical relation indication but no nuclearity indication ('R'), and finally on the fully-labeled tree structure with both nuclearity and rhetorical relation labels ('F'). For each criteria, precision is calculated by taking the ratio of the number of identical tree constituents found in both the generated RS-tree and the gold-standard tree, against the total number of constituents in the generated discourse tree. Recall is calculated by taking the ratio of the number of identical tree constituents found in both the generated RS-tree and the gold-standard tree, against the total number of constituents in the gold-standard discourse tree. The results are summarized in Figure 9. Note that when using perfect segmentation, precision and recall are identical since both trees have same number of constituents.

Table 9: Full parser evaluation, using the standard test subset

Seg.	**Manual**				**SPADE-Seg**				**HILDA-Seg**			
	S	N	R	F	S	N	R	F	S	N	R	F
Precision	83.0	68.4	55.3	54.8	69.5	56.1	44.9	44.4	73.0	59.7	48.2	47.7
Recall	83.0	68.4	55.3	54.8	69.2	55.8	44.7	44.2	71.7	58.6	47.4	46.9
F-Score	83.0	68.4	55.3	54.8	69.3	56.0	44.8	44.3	72.3	59.1	47.8	47.3

As expected, we observe the highest performance when using manual segmentation, with an F-score of 83% for structure annotation, and 54.8% for the fully-annotated tree. Using SPADE yields lower results, with an F-score of 69.3% for structure annotation, and 44.3% for the full discourse tree. On the other hand, when using the discourse segmenter presented in Section 4.1, structure annotation produces F-scores of 72.3% for structure annotation, and 47.3% for the full tree. These differences indicate that, although these segmenters score high on the segmentation task (above 80%), errors occurring at the beginning of the pipeline are amplified and weight heavily on the performance of the output. Furthermore, the parse trees of SPADE-Seg and HILDA-Seg are generated automatically, which also explains the lower results.

In order to compare our results to the gold standard (defined as manual agreement between human annotators) more accurately, we also evaluated performances using another test set (see Table 10) of 52 files annotated by two labelers of the RST-DT. In each case, the remaining 340-350 files are used for training.

The F-score for human agreement is 65.3%. When using manual segmentation, we obtain an F-score of 55.1% on this test set, which is 84% of human performance level. When using our own segmenter, we obtain an F-score of 48.8%, which is 74.7% of human performance level.

Table 10: Evaluation on a doubly-annotated subset and comparison to human agreement

	System performance								Human agreement			
Seg.	**Manual**				**HILDA-Seg**				**–**			
	S	N	R	F	S	N	R	F	S	N	R	F
Precision	84.1	70.6	55.6	55.1	74.0	61.7	49.4	48.9	88.0	77.5	66.0	65.2
Recall	84.1	70.6	55.6	55.1	73.7	61.5	49.1	48.7	88.1	77.6	66.1	65.3
F-Score	84.1	70.6	55.6	**55.1**	73.8	61.6	49.2	**48.8**	88.1	77.5	66.0	**65.3**

4.4 Comparison to other systems

In this section, we attempt to compare our system to other text level discourse parsers, namely Marcu's decision-tree based parser (Marcu, 2000), the multi-level rule-based system built by Le Thanh et al. (2004b), and the shift-reduce parser of Sagae (2009).

For comparison with Marcu and LeThanh's systems, we face several difficulties: Marcu (2000) reports his results on unspecified documents and we do not have access to the parser software. Therefore we could not evaluate his parser on our test set. Then, in the case of LeThanh's system, we do have access to the documents on which the system was evaluated, but we do not have access to the parser software either. Another issue comes from the number of discourse relations employed: In Marcu's case, 15 relations were employed, while LeThanh used 14, and we used 18.

For these two systems, we have no choice but to reproduce the results provided by each author, in separate tables for fairness. We define a measure, F_{algo}/F_{human}, computed for each system, as an informal indicator of how each parser is estimated to perform compared to the human agreement. As each author found a different value for human agreement F-score, we scale the score of each system by the human agreement F-score found by its respective author. For instance, Le Thanh et al. (2004b) report a surprisingly low 72.7% F-score for structure, when Marcu (2000) notes 81%, and we found 87%. This suggests that, despite our best efforts, evaluation metrics used by each author might differ. Again, this measure is an informal indicator of performance. Table 11 indicates the results given by Marcu (2000), Table 12 the results given by Le Thanh et al. (2004b), and Table 13 the results for our system. Note that in Table 13, in order to make our experimental setting as close as possible to LeThanh's, we used LeThanh's set of 21 documents as testing subset, and the rest of the corpus for training.

Using our calculated human-agreement F-scores, we estimate that our system reaches 86.4% of the human performance level for structure labeling, while for nuclearity and relation labeling, we obtain 79.8% and 78.3%, respectively, of the human F-score. Marcu (2000) calculated reaching 25.7% of the human performance level for relation labeling, while Le Thanh et al. (2004b) calculated reaching 39.9% of this level.

Direct comparison to our system is only possible in the case of Sagae (2009), as Sagae employed the same 18 standard RST-DT relations, and evaluation was done on the RST-DT's test subset. A performance comparison of both systems is presented in Table 14. The scores are for full discourse tree creation. We observe that the performance of the two parsers is relatively similar. In particular, recall is 46.2% in SAG, while we obtained 46.9% in HILDA. Precision for HILDA is however 11% higher than SAG, at 47.7%, against 42.9% for Sagae's system.

Table 11: Performance for the parser of Marcu (2000), using 15 discourse relations, evaluated on unspecified documents

	Structure	**Nuclearity**	**Relations**
Precision	65.8	54.0	34.3
Recall	34.0	21.6	13.0
F-score	44.8	30.9	18.8
$\mathbf{F}_{algo}$/$\mathbf{F}_{human}$	56.0	42.9	25.7

Table 12: Performance for the parser of Le Thanh et al. (2004b), using 14 discourse relations, evaluated on 21 documents from the RST-DT

	Structure	**Nuclearity**	**Relations**
Precision	54.5	47.8	40.5
Recall	52.9	46.4	39.3
F-score	53.7	47.1	39.9
$\mathbf{F}_{algo}$/$\mathbf{F}_{human}$	73.9	71.8	70.1

4.5 Time performance

To complete the discussion of the parser's actual performance, we measure the time taken to parse standard texts from the test section of the RST-DT. The computer used in the tests has a 2.53 GHz processor and 4 GB of RAM. On average, loading the Stanford syntax parser and SVM models altogether takes four seconds. This is done only once. We use modified command-line versions of $libsvm$ and $liblinear$ to accept data from `stdin`, so that loading of models is performed only once. After this, we can perform fast classification decisions, taking a few tenths of a second each. Table 15 shows the performance of the parser on various texts. t_{seg} is the time taken to segment the text, and t_{build} the time taken to build the tree.

As expected, we empirically observe that t_{seg} and t_{build} follow a $\mathcal{O}(n)$ time complexity. However, in the system, the only process not under our control is the syntax parser. It is important to ensure that the time taken to generate parse trees is reasonable. On average, on our test computer, the Stanford parser takes 10 seconds to parse 300 words. Of the three processes (syntax parsing, discourse segmentation, tree building), syntax parsing is always the slowest task, and thus the bottleneck of the system. Still, this level of performance makes the parser usable as part of an interactive system.

Table 13: Performance for HILDA, using 18 discourse relations, evaluated on 21 documents from the RST-DT (LeThanh's)

	Structure	**Nuclearity**	**Relations**
Precision	76.0	61.4	51.2
Recall	75.6	61.2	50.6
F-score	75.8	61.3	50.9
$\mathbf{F}_{algo}$/$\mathbf{F}_{human}$	86.4	79.8	78.3

Table 14: Performance comparison between the proposed parser (HILDA) and the parser of Sagae (2009) (SAG), for full discourse tree creation, using 18 discourse relations, evaluated on the RST-DT's test set

	SAG	**HILDA**
Precision	42.9	**47.7**
Recall	46.2	**46.9**
F-Score	44.5	**47.3**

4.6 Limitations

In this section, we present some limitations of the parser and discuss possible improvements and directions for future work.

When evaluating the proposed discourse segmenter on the RST-DT's test set, we observe that all segmentation errors are due to over-segmentation, i.e. words which are not EDU boundaries are mistaken for boundaries. Table 16 shows the ten most frequent words on which segmentation errors occur, as well as how prevalent these errors are among all segmentation errors. Several of these mistakes are linked to punctuation, particularly to quotes and dash marks. Also, several mistakes seem related to the segmenter creating excessively small clause-like units. Typical improper seg-

Table 15: Time taken to parse various texts (in seconds)

Text	**Sentences**	**Words**	t_{seg} **(s)**	t_{build} **(s)**
wsj_{0616}	36	838	5.02	11.30
wsj_{1113}	5	102	0.40	0.51
wsj_{1129}	2	45	0.26	0.40
wsj_{2336}	10	262	1.71	3.72
wsj_{2385}	50	516	3.05	7.17

mentation decisions (marked ↑) related to these words are seen in the following sentences, where the true segmentation decisions (when present) are marked ⇑:

```
`` [...]  the Banco exterior group has a lot ↑ to offer a
potential suitor.''
`` [...]  when the market does recover, ↑ the damage is
done [...]''
`` [...]  cut costs, increase capital ↑ and build new areas
of business [...]''
`` [...]  and perhaps even destroy ↑ -- ⇑ a $2.38
settlement fund [...]''
`` In August, ↑ Mr.  Lewis pleaded guilty to three felony
counts.''
`` When it misses one month ↑ it tends to miss the next
month.''
```

Table 16: Words on which erroneous segmentation decisions are most frequent, and their proportion among all segmentation errors on the RST-DT's test set

Word	**POS**	**Prevalence among errors (%)**
to	TO	7.7
the	DT	6.6
and	CC	4.6
“	“	3.4
–	:	2.3
Mr.	NNP	2.3
it	PRP	2.1
for	IN	2.0
if	IN	1.8
said	VBD	1.6

Although a quantitative evaluation of the discourse trees produced by our parser is made possible by the usage of metrics traditionally employed for evaluating the performance of syntax parsers, we found that a qualitative evaluation of the generated discourse trees was difficult. In particular, we hypothesize that finding patterns of errors commonly present in the generated discourse trees is made difficult by the greedy approach adopted. Because of the proposed algorithm and the way the classifiers are cascaded, classification errors occurring anywhere when building the discourse tree will affect both sibling nodes and higher level relations of the tree, making the real cause of errors difficult to identify. Indeed, the presented greedy algorithm is time-efficient, but provides only locally optimal solutions. Using generic global probabilistic optimization meta-algorithms such as simulated annealing (Kirkpatrick et al., 1983), we hope to address the problem of local optimality while maintaining reasonable time complexity. Secondly, we plan to investigate the use of sequential labeling methods (Lafferty et al., 2001) for finding optimal sequences of discourse relations.

Another related shortcoming of the current algorithm, as presented in Figure 6, occurs when two pairs of EDUs have an equal probability of being connected by a discourse relation. In the present system, the default behavior in case of conflict is to select the first pair of units in text order. However, ideally, all candidates should be considered, with their respective trees built independently, somehow scored, and the best candidate being finally returned. We hope to address this point by employing the aforementioned optimization methods.

In Section 4.2, we noticed important discrepancies in F-scores (see Table 7) for the different classes of $Label$. We hypothesize that this issue is mainly related to the nature of discourse relations. Indeed, a notoriously challenging task for discourse parsers is to detect 'implicit relations', i.e. relations not signaled by discourse cues, such as 'thus', 'however', 'but', etc. Certain relations, such as ATTRIBUTION, are invariably realized with a discourse cue, indicated by the verb 'say', such as 'X said that Y'. This relation is easy to detect, and we empirically observe high performance on ATTRIBUTION's classes (see Table 7). On the other hand, an implicit CONTRAST relation holds between the two following discourse units:

[Mr. Roman comes across as a low-key executive;]^{2A} [Mr. Phillips has a flashier personality.]^{2B}

In this example, the contrast is captured by the antonymy between the adjectives *low-key* and *flashy*. As there is no cue, it is harder to detect, and we observe in practice a lower performance for CONTRAST relations. Taboada (2006) estimates that 60% to 70% of naturally-occurring discourse relations are implicit. In general, while explicit discourse relations are well detected (Pitler et al., 2008), implicit relations are challenging to detect. For instance, we fed the proposed parser ten short sentences containing an implicit CONTRAST relation. All relations were wrongly classified, either as ELABORATION in six cases or as JOINT in four cases. Similar errors were seen when trying to classify implicit COMPARISON and TEMPORAL relations. This indicates that our set of features is not fully-adapted for implicit relations. Recent evidence (Webber, 2009) even suggests that the features present when an explicit discourse relation is realized differ significantly from those present in the case of an implicit relation. A possible solution is to train a separate discourse relation classifier for implicit discourse relations. A different set of features will be required. For this task, several researchers (Marcu and Echihabi, 2002; Pitler et al., 2009; Lin et al., 2009) have noted relatively good performance of word pair features taken between the relation's two argument

spans. For instance, using the same example, we would encode lemmatized word pairs such as (Mr., Mr.), (Mr., Phillips), ..., (low-key, flashy), ..., (executive, personality), etc. However, word pair features alone are not sufficient to obtain high performance rates. We plan to investigate implicit discourse relation classification by employing features based on measures of semantic similarity between EDUs, using for instance WordNet (Fellbaum, 1998).

Finally, we plan to apply feature selection in order to effectively reduce the size of the feature set. This method has the promise of shorter training and testing times. Another interesting aspect of feature selection is the potential reduction of noise, which may lead to an improvement in accuracy.

5. Conclusions

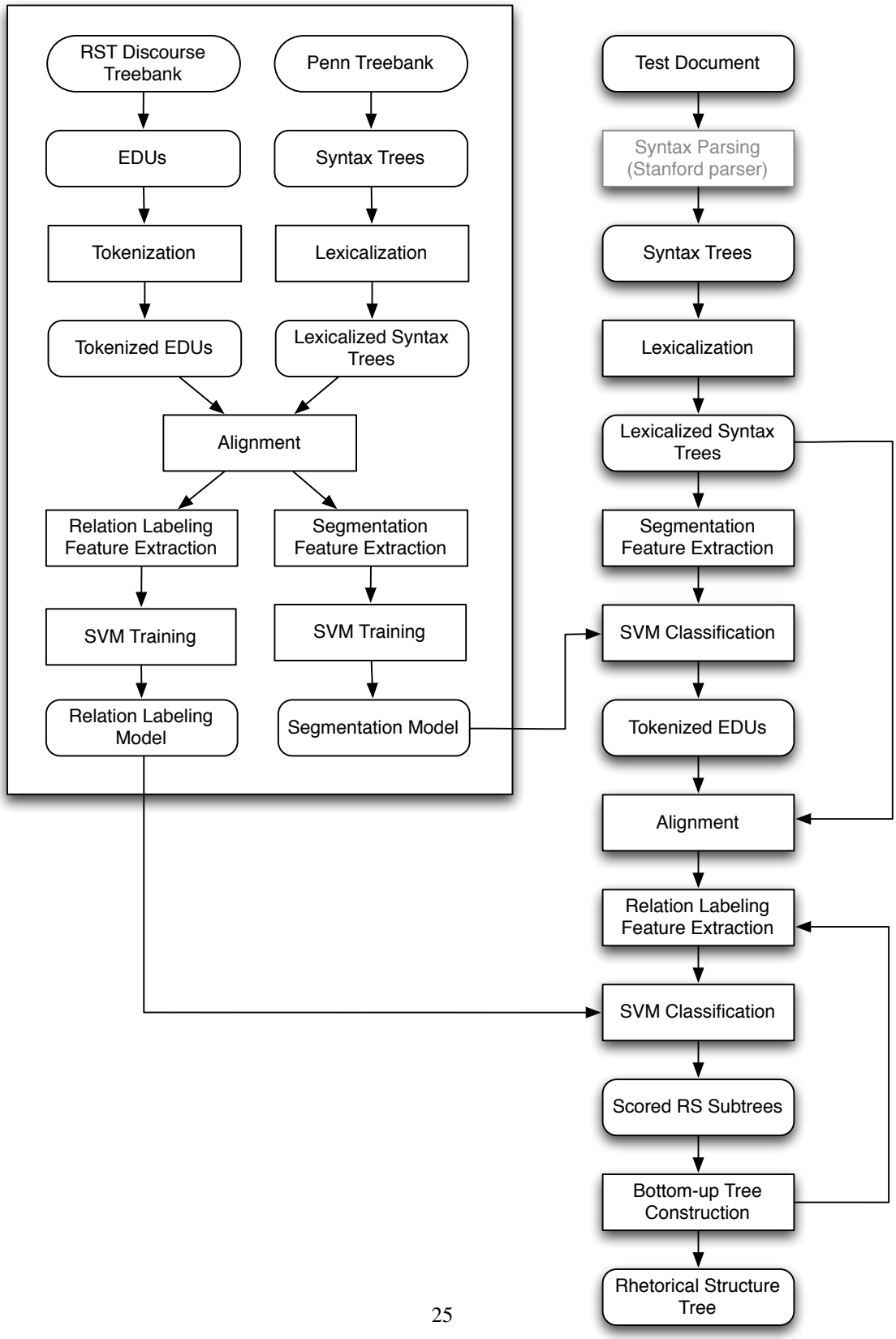
We presented HILDA, an automated discourse parser that analyzes text in the framework of the Rhetorical Structure Theory. The system is composed of two modules, (1) a discourse segmenter, and (2) a relation classifier and tree builder. Both tasks are based on Support Vector Machine classification. The discourse segmenter performs at around 96% of the human level, while the relation classification and tree-building module reaches 84% of human level. Our system combines both modules for creating a fully automatic text level parser, reaching 78.3% of the human performance level.

Importantly, the tree-building process performs in linear time, which allows us to use the system in the context of a (near) real-time or interactive system, such as dialogue generation or question-answering. For instance, the proposed parser can be integrated in a dialogue generation system such as Text-to-Dialogue (Prendinger et al., 2007; Hernault et al., 2008). In this system, RST structures are used as the backbone for generating dialogues between two agents representing a layperson and an expert. Question-answer pairs between the two agents are created by applying relation-specific mapping rules manipulating the leaves of the RST tree. If the RST tree utilized contains errors, such as improperly-segmented EDUs or erroneous discourse relations, the generated dialogue will be semantically incorrect. Hence, a sound discourse parser is important.

More recently, the CODA corpus (Stoyanchev and Piwek, 2010) was released. This corpus contains 700 expository dialogues labeled with dialogue acts, paired with human-authored, equivalent monologues annotated with RST discourse relations. Using this corpus, one can learn how to generate an expository dialogue from an RST-annotated monologue. An ambitious next step is to automatize the generation process, i.e. creating a meaningful expository dialogue given any plain, non-annotated monologue. This is a promising perspective for creating systems that are able to present information automatically, such as tutoring applications. A prerequisite is the accurate detection of the discourse relations in the input monologues, and hence, the role of the discourse parser is once again central.

With these applications in mind, we plan to make the proposed parser more efficient, by addressing the issues presented in Section 4.6. In particular, we intend to improve the tree-building method, in order to create globally optimal discourse structures. Finally, we plan to train a separate classifier for detecting implicit discourse relations as well.

Appendix A. System workflow



Appendix B. Relation list

Relation	Nuclearity options	
ATTRIBUTION	Nucleus	Satellite
	Satellite	Nucleus
BACKGROUND	Nucleus	Satellite
	Satellite	Nucleus
CAUSE	Nucleus	Nucleus
	Nucleus	Satellite
	Satellite	Nucleus
COMPARISON	Nucleus	Nucleus
	Nucleus	Satellite
	Satellite	Nucleus
CONDITION	Nucleus	Nucleus
	Nucleus	Satellite
	Satellite	Nucleus
CONTRAST	Nucleus	Nucleus
	Nucleus	Satellite
	Satellite	Nucleus
ELABORATION	Nucleus	Satellite
	Satellite	Nucleus
ENABLEMENT	Nucleus	Satellite
	Satellite	Nucleus
EVALUATION	Nucleus	Nucleus
	Nucleus	Satellite
	Satellite	Nucleus
EXPLANATION	Nucleus	Nucleus
	Nucleus	Satellite
	Satellite	Nucleus
JOINT	Nucleus	Nucleus
MANNER-MEANS	Nucleus	Satellite
	Satellite	Nucleus
SUMMARY	Nucleus	Satellite
	Satellite	Nucleus
TEMPORAL	Nucleus	Nucleus
	Nucleus	Satellite
	Satellite	Nucleus
TOPIC-CHANGE	Nucleus	Nucleus
	Nucleus	Satellite
TOPIC-COMMENT	Nucleus	Nucleus
	Nucleus	Satellite
	Satellite	Nucleus
SAME-UNIT	Nucleus	Nucleus
TEXTUAL-ORGANIZATION	Nucleus	Nucleus

References

Nicholas Asher and Alex Lascarides. *Logics of conversation*. Cambridge University Press, 2003.

Jason Baldridge and Alex Lascarides. Probabilistic head-driven parsing for discourse structure. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 96–103, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

Ezra Black, Steven P. Abney, D. Flickenger, Claudia Gdaniec, Ralph Grishman, P. Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith L. Klavans, Mark Liberman, Mitchell P. Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. A procedure for quantitatively comparing the syntactic coverage of english grammars. In *Proceedings of Workshop on Speech and Natural Language*, pages 306–311. Association for Computational Linguistics Morristown, NJ, USA, 1991.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. *Proceedings of Second SIGdial Workshop on Discourse and Dialogue-Volume 16*, pages 1–10, 2001.

Joyce Y. Chai and Rong Jin. Discourse structure for context question answering. In Sanda Harabagiu and Finley Lacatusu, editors, *HLT-NAACL 2004: Workshop on Pragmatics of Question Answering*, pages 23–30, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.

Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

Degang Chen, Qiang He, and Xizhao Wang. On linear separability of data sets in feature space. *Neurocomputing*, 70(13-15):2441–2448, 2007.

David duVerle and Helmut Prendinger. A novel discourse parser based on support vector machine classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 665–673, Suntec, Singapore, August 2009. Association for Computational Linguistics.

Christiane Fellbaum, editor. *WordNet: An electronic lexical database*. MIT Press, 1998.

Hugo Hernault, Paul Piwek, Helmut Prendinger, and Mitsuru Ishizuka. Generating dialogues for virtual agents using nested textual coherence relations. In *IVA '08: Proceedings of the 8th international conference on Intelligent Virtual Agents*, pages 139–145, Berlin, Heidelberg, 2008. Springer-Verlag.

Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.

Dan Klein and Christopher D. Manning. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems*, volume 15, pages 3–10. MIT Press, 2003.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML'01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

Huong Le Thanh, Geetha Abeysinghe, and Christian Huyck. Automated discourse segmentation by syntactic information and cue phrases. In *Proceedings of AIA'04*, Innsbruck, Austria, February 16–18 2004a.

Huong Le Thanh, Geetha Abeysinghe, and Christian Huyck. Generating discourse structures for written texts. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 329–335, Geneva, Switzerland, Aug 23–Aug 27 2004b. COLING.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. Recognizing implicit discourse relations in the Penn Discourse Treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 343–351, Singapore, August 2009. Association for Computational Linguistics.

David M. Magerman. Statistical decision-tree models for parsing. *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 276–283, 1995.

William C. Mann and Sandra A. Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281, 1988.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language processing*. MIT Press, 1999.

Daniel Marcu. Building up rhetorical structure trees. *Proceedings of the National Conference on Artificial Intelligence*, pages 1069–1074, 1996.

Daniel Marcu. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press, 2000.

Daniel Marcu and Abdessamad Echihabi. An unsupervised approach to recognizing discourse relations. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 368–375, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073145.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

Johanna D. Moore and Peter Wiemer-Hastings. Discourse in computational linguistics and artificial intelligence. In A. Graesser, M. Gernsbacher, and S. Goldman, editors, *Handbook of Discourse Processes*, pages 439–486. Erlbaum, Mahwah, NJ, 2003.

Jon Oberlander and Johanna D. Moore. Cue phrases in discourse: Further evidence for the core:contributor distinction. In *Workshop on Levels of Representation in Discourse*, pages 87–93, Edinburgh, UK, 1999.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind Joshi. Easily identifiable discourse relations. In *Coling 2008: Companion volume: Posters*, pages 87–90, Manchester, UK, August 2008. Coling 2008 Organizing Committee.

Emily Pitler, Annie Louis, and Ani Nenkova. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 683–691, Suntec, Singapore, August 2009. Association for Computational Linguistics.

Helmut Prendinger, Paul Piwek, and Mitsuru Ishizuka. A novel method for automatically generating multi-modal dialogue from text. *International Journal of Semantic Computing*, 1(3):319–334, 2007.

David Reitter. Rhetorical Analysis with Rich-Feature Support Vector Models. *Unpublished Master's thesis, University of Potsdam, Potsdam, Germany*, 2003a.

David Reitter. Simple signals for complex rhetorics: On rhetorical analysis with rich-feature support vector models. *LDV Forum*, 18(1/2):38–52, 2003b.

Kenji Sagae. Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 81–84, Paris, France, October 2009. Association for Computational Linguistics.

Frank Schilder. Robust discourse parsing via discourse markers, topicality and position. *Natural Language Engineering*, 8(2-3):235–255, 2002.

Radu Soricut and Daniel Marcu. Sentence level discourse parsing using syntactic and lexical information. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, 1:149–156, 2003.

Carl Staelin. Parameter selection for Support Vector Machines. *Hewlett-Packard Company, Tech. Rep. HPL-2002-354R1*, 2003.

Svetlana Stoyanchev and Paul Piwek. Constructing the coda corpus: A parallel corpus of monologues and expository dialogues. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 2010. European Language Resources Association.

Rajen Subba and Barbara Di Eugenio. Automatic discourse segmentation using neural networks. In *Proceedings of 11th Workshop on the Semantics and Pragmatics of Dialogue*, pages 189–190, Trento, Italy, 2007.

Rajen Subba and Barbara Di Eugenio. An effective discourse parser that uses rich linguistic information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of*

the North American Chapter of the Association for Computational Linguistics, pages 566–574, Boulder, Colorado, June 2009. Association for Computational Linguistics.

Maite Taboada. Discourse markers as signals (or not) of rhetorical relations. *Journal of Pragmatics*, 38(4):567–592, 2006.

Milan Tofiloski, Julian Brooke, and Maite Taboada. A syntactic and lexical-based discourse segmenter. In *ACL '09*, pages 77–80, Suntec, Singapore, August 2009. Association for Computational Linguistics.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

Bonnie Webber. Genre distinctions for discourse in the penn treebank. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 674–682, Suntec, Singapore, August 2009. Association for Computational Linguistics.

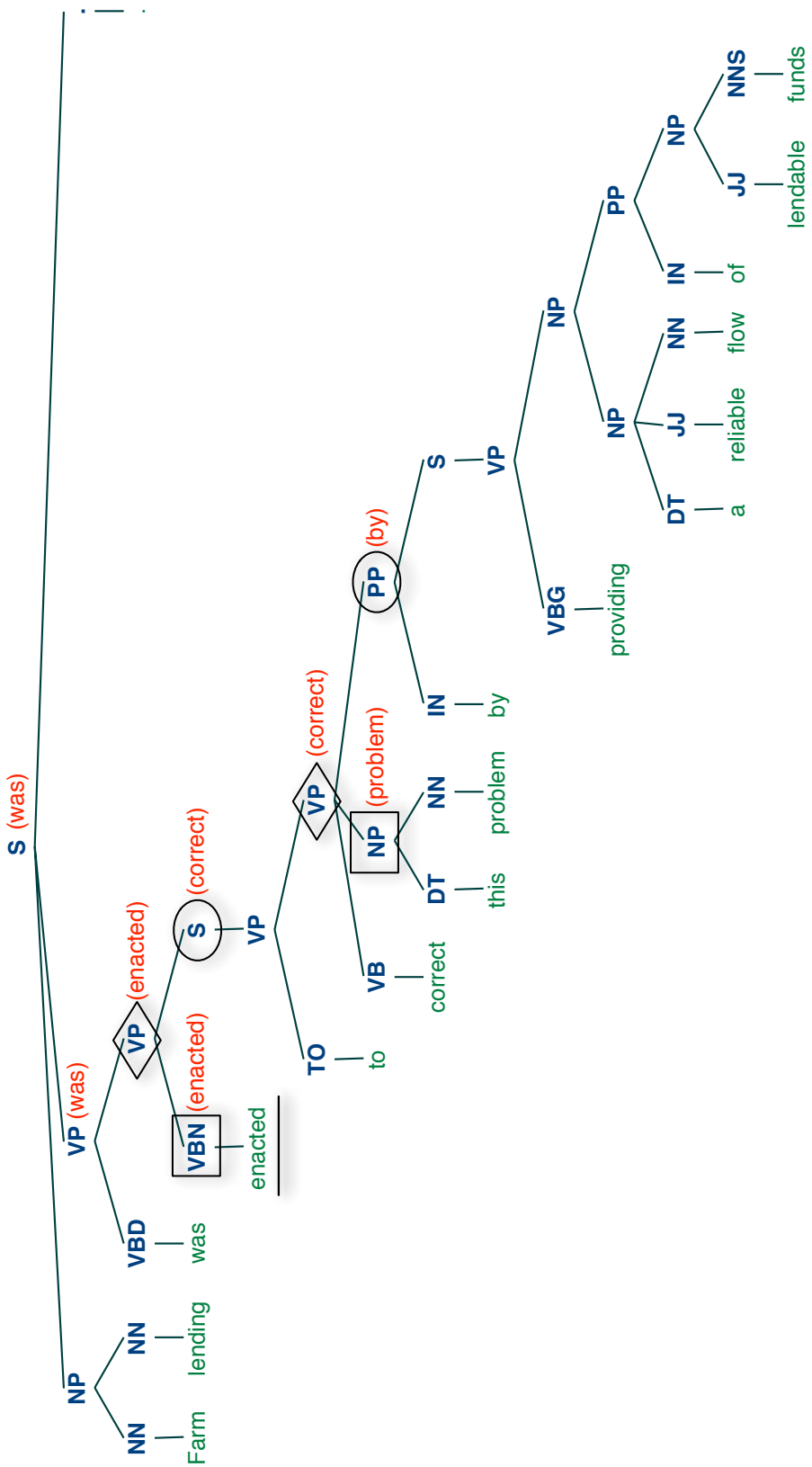


Figure 7: Partially-lexicalized syntax tree (lexical heads are indicated between parentheses)

Figure 8: Two possible attachments in the RS-tree of the text in Figure 1.

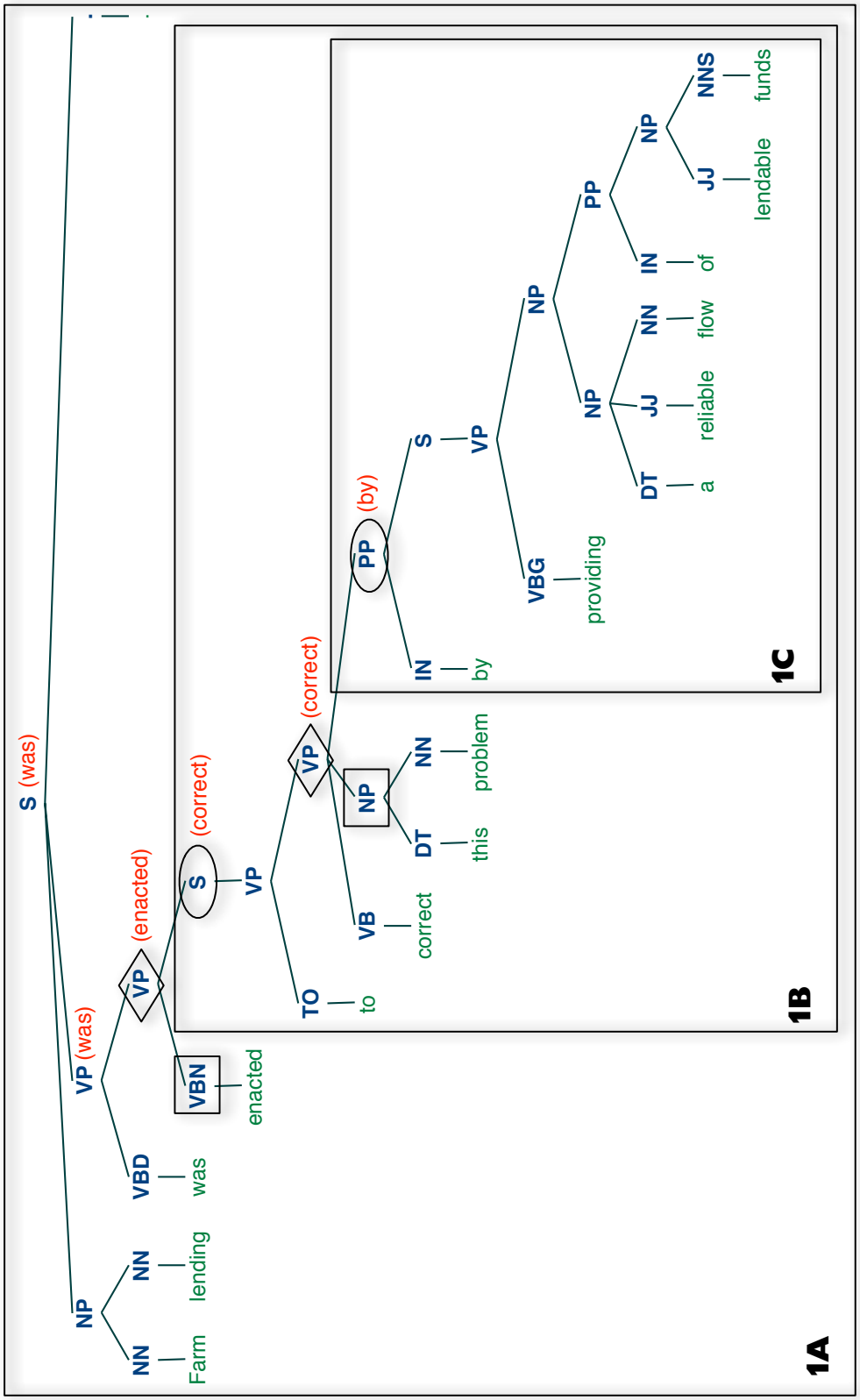


Figure 9: Partially-lexicalized syntax tree, showing dominance sets