A Novel Discourse Parser Based on Support Vector Machine Classification

David A. duVerle
National Institute of Informatics
Tokyo, Japan
Pierre & Marie Curie University
Paris, France
dave@nii.ac.jp

Helmut Prendinger
National Institute of Informatics
Tokyo, Japan
helmut@nii.ac.jp

Abstract

This paper introduces a new algorithm to parse discourse within the framework of Rhetorical Structure Theory (RST). Our method is based on recent advances in the field of statistical machine learning (multivariate capabilities of Support Vector Machines) and a rich feature space. RST offers a formal framework for hierarchical text organization with strong applications in discourse analysis and text generation. We demonstrate automated annotation of a text with RST hierarchically organised relations, with results comparable to those achieved by specially trained human annotators. Using a rich set of shallow lexical, syntactic and structural features from the input text, our parser achieves, in linear time, 73.9% of professional annotators' human agreement F-score. The parser is 5% to 12% more accurate than current state-of-the-art parsers.

1 Introduction

According to Mann and Thompson (1988), all well-written text is supported by a hierarchically structured set of coherence relations which reflect the authors intent. The goal of discourse parsing is to extract this high-level, rhetorical structure.

Dependency parsing and other forms of syntactic analysis provide information on the grammatical structure of text at the sentential level. Discourse parsing, on the other hand, focuses on a higher-level view of text, allowing some flexibility in the choice of formal representation while providing a wide range of applications in both analytical and computational linguistics. Rhetorical Structure Theory (Mann and Thompson, 1988) provides a framework to analyze and study text coherence by defining and applying a set of structural relations to composing units ('spans') of text. Annotation of a text within the RST formalism will produce a tree-like structure that not only reflects text-coherence but also provides input for powerful algorithmic tools for tasks such as text regeneration (Piwek et al., 2007).

RST parsing can be seen as a two-step process:

1. Segmentation of the input text into elementary discourse units ('edus').

2. Generation of the rhetorical structure tree based on 'rhetorical relations' (or 'coherence relations') as labels of the tree, with the edus constituting its terminal nodes.

Mann and Thompson (1988) empirically established 110 distinct rhetorical relations, but pointed out that this set was flexible and open-ended. In addition to rhetorical relations, RST defines the notion of 'nucleus', the relatively more important part of the text, and 'satellite', which is subordinate to the nucleus. In Fig. 1, the left-most edu constitutes the satellite (indicated by out-going arrow), and the right-hand statement constitutes the nucleus. Observe that the nucleus itself is a compound of nucleus and satellite.

Several attempts to automate discourse parsing have been made. Marcu and Soricut focussed on sentence-level parsing and developed two probabilistic models that use syntactic and lexical information (Soricut and Marcu, 2003). Although their algorithm, called 'SPADE', does not produce full-text parse, it demonstrates a correlation between syntactic and discourse information, and their use to identify rhetorical relations even if no signaling cue words are present.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 665–673,
Suntec, Singapore, 2-7 August 2009.

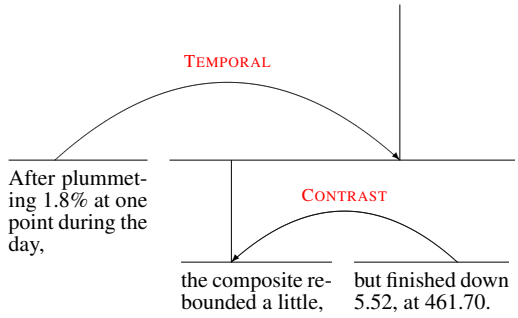

Figure 1: Example of a simple RST tree (Source: RST Discourse Treebank (Carlson et al., 2001), wsj_{0667}).

To the best of our knowledge, Reitter's (2003b) was the only previous research based exclusively on feature-rich supervised learning to produce text-level RST discourse parse trees. However, his full outline for a working parser, using chart-parsing-style techniques, was never implemented.

LeThanh *et al.* (2004) proposed a multi-step algorithm to segment and organize text spans into trees for each successive level of text organization: first at sentence level, then paragraph and finally text. The multi-level approach taken by their algorithm mitigates the combinatorial explosion effect without treating it entirely. At the text-level, and despite the use of beam search to explore the solution space, the algorithm needs to produce and score a large number of trees in order to extract the best candidate, leading, in our experience, to impractical calculation times for large input.

More recently, Baldridge and Lascarides (2005) successfully implemented a probabilistic parser that uses headed trees to label discourse relations. Restricting the scope of their research to texts in dialog form exclusively, they elected to use the more specific framework of Segmented Discourse Representation Theory (Asher and Lascarides, 2003) instead of RST.

In this paper, we advanced the state-of-the-art in general discourse parsing, with an implemented solution that is computationally efficient and sufficiently accurate for use in real-time interactive applications. The rest of this paper is organized as follows: Section 2 describes the general architecture of our system along with the choices we made with regard to supervised learning. Section 3 explains the different characteristics of the input text used to train our system. Section 4 presents our results, and Section 5 concludes the paper.

2 Building a Discourse Parser

2.1 Assumptions and Restrictions

In our work, we focused exclusively on the second step of the discourse parsing problem, i.e., constructing the RST tree from a sequence of edus that have been segmented beforehand. The motivation for leaving aside segmenting were both practical – previous discourse parsing efforts (Soricut and Marcu, 2003; LeThanh et al., 2004) already provide alternatives for standalone segmenting tools – and scientific, namely, the greater need for improvements in labeling. Current state-of-the-art results in automatic segmenting are much closer to human levels than full structure labeling (F-score ratios of automatic performance over gold standard reported in LeThanh *et al.* (2004): 90.2% for segmentation, 70.1% for parsing).

Another restriction is to use the reduced set of 18 rhetorical relations defined in Carlson *et al.* (2001) and previously used by Soricut and Marcu (2003). In this set, the 75 relations originally used in the RST Discourse Treebank (RST-DT) corpus (Carlson et al., 2001) are partitioned into 18 classes according to rhetorical similarity (e.g.: PROBLEM-SOLUTION, QUESTION-ANSWER, STATEMENT-RESPONSE, TOPIC-COMMENT and COMMENT-TOPIC are all grouped under one TOPIC-COMMENT relation). In accord with previous research (Soricut and Marcu, 2003; Reitter, 2003b; LeThanh et al., 2004), we turned all *n*-ary rhetorical relations into nested binary relations (a trivial graph transformation), resulting in more algorithmically manageable binary trees. Finally, we assumed full conformity to the 'Principle of sequentiality' (Marcu, 2000), which guarantees that only adjacent spans of text can be put in relation within an RST tree, and drastically reduces the size of the solution space.

2.2 Support Vector Machines

At the core of our system is a set of classifiers, trained through supervised-learning, which, given two consecutive spans (atomic edus or RST sub-trees) in an input document, will score the likelihood of a direct structural relation as well as probabilities for such a relation's label and nuclearity. Using these classifiers and a straightforward bottom-up tree-building algorithm, we can produce a valid tree close to human cross-

validation levels (our gold standard) in linear time-complexity (see Fig. 2).

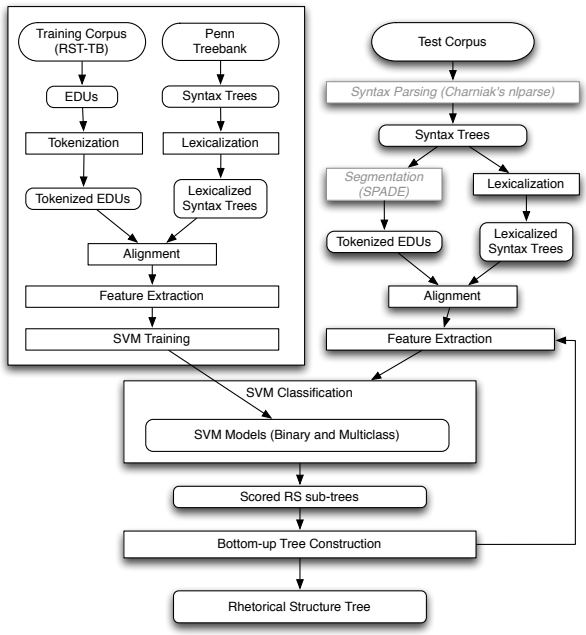

Figure 2: Full system workflow.

In order to improve classification accuracy, it is convenient to train two separate classifiers:

- S: A binary classifier, for structure (existence of a connecting node between the two input sub-trees).
- L: A multi-class classifier, for rhetorical relation and nuclearity labeling.

Using our reduced set of 18 super-relations and considering only valid nuclearity options (e.g., $(\text{ATTRIBUTION}, N, S)$ and $(\text{ATTRIBUTION}, S, N)$, but *not* $(\text{ATTRIBUTION}, N, N)$, as ATTRIBUTION is a purely hypotactic relation group), we come up with a set of 41 classes for our algorithm.

Support Vector Machines (SVM) (Vapnik, 1995) are used to model classifiers S and L. SVM refers to a set of supervised learning algorithms that are based on margin maximization. Given our specific type of classification problem, SVMs offer many properties of particular interest. First, as *maximum margin classifiers*, they sidestep the common issue of overfitting (Scholkopf et al., 1995), and ensure a better control over the generalization error (limiting the impact of using homogeneous newspaper articles that could carry important biases in prose style and lexical content). Second, SVMs offer more resilience to noisy input. Third, depending on the parameters used (see the use of kernel functions below), training time complexity's dependence on feature vector size is low, in some cases linear. This makes SVM well-fitted to treat classification problems involving relatively large feature spaces such as ours ($\approx 10^5$ features). Finally, while most probabilistic classifiers, such as Naive Bayes, strongly assume feature independence, SVMs achieve very good results regardless of input correlations, which is a desirable property for language-related tasks.

SVM algorithms make use of the 'kernel trick' (Aizerman et al., 1964), a method for using linear classifiers to solve non-linear problems. Kernel methods essentially map input data to a higher-dimensional space before attempting to classify them. The choice of a fitting kernel function requires careful analysis of the data and must weigh the effects on both performance and training time. A compromise needs to be found during evaluation between the general efficiency of non-linear kernels (such as polynomial or Radial Basis Function) and low time-complexity of using a linear function (see Sect. 4).

Because the original SVM algorithms build binary classifiers, *multi-label classification* requires some adaptation. A possible approach is to reduce the multi-classification problem through a set of binary classifiers, each trained either on a single class ("one vs. all") or by pair ("one vs. one"). Recent research suggests keeping the classification whole, with a reformulation of the original optimization problem to accommodate multiple labels ("C & S") (Crammer and Singer, 2002).

2.3 Input Data and Feature Extraction

Both S and L classifiers are trained using manually annotated documents taken from the RST-DT corpus. Optimal parameters (when applicable) for each kernel function are obtained through automated grid search with n-fold cross-validation (Staelin, 2003) on the training corpus, while a separate test set is used for performance evaluation. In training mode, classification instances are built by parsing manually annotated trees from the RST-DT corpus paired with lexicalized syntax trees (LS Trees) for each sentence (see Sect. 3). Syntax trees are taken

directly from the Penn Treebank corpus (which covers a superset of the RST-DT corpus), then "lexicalized" (i.e. tagged with lexical "heads" on each internal node of the syntactic tree) using a set of canonical head-projection rules (Magerman, 1995; Collins, 2003). Due to small differences in the way they were tokenized and pre-treated, rhetorical tree and LST are rarely a perfect match: optimal alignment is found by minimizing edit distances between word sequences.

2.4 Tree-building Algorithm

By repeatedly applying the two classifiers and following a naive bottom-up tree-construction method, we are able to obtain a globally satisfying RST tree for the entire text with excellent time-complexity.

The algorithm starts with a list of all atomic discourse sub-trees (made of single edus in their text order) and recursively selects the best match between adjacent sub-trees (using binary classifier S), labels the newly created sub-tree (using multi-label classifier L) and updates scoring for S, until only one sub-tree is left: the complete rhetorical parse tree for the input text.

It can be noted that, thanks to the principle of sequentiality (see Sect. 2.1), each time two sub-trees are merged into a new sub-tree, only connections with adjacent spans on each side are affected, and therefore, only two new scores need to be computed. Since our SVM classifiers work in linear time, the overall time-complexity of our algorithm is $\mathcal{O}(n)$.

3 Features

Instrumental to our system's performance is the choice of a set of salient characteristics ("features") to be used as input to the SVM algorithm for training and classification. Once the features are determined, classification instances can be formally represented as a vector of values in $\mathcal{R}$.

We use n-fold validation on S and L classifiers to assess the impact of some sets of features on general performance and eliminate redundant features. However, we worked under the (verified) assumption that SVMs' capacity to handle high-dimensional data and resilience to input noise limit the negative impact of non-useful features.

In the following list of features, obtained empirically by trial-and-error, features suffixed by 'S[pan]' are sub-tree-specific features, symmetrically extracted from both left and right candidate spans. Features suffixed by 'F[ull]' are a function of the two sub-trees considered as a pair. Multi-label features are turned into sets of binary values and trees use a trivial fixed-length binary encoding that assumes fixed depth.

3.1 Textual Organization

As evidenced by a number of discourse-parsing efforts focusing on intra-sentential parsing (Marcu, 2000; Soricut and Marcu, 2003), there is a strong correlation between different organizational levels of textual units and sub-trees of the RST tree both at the sentence-level and the paragraph level. Although such correspondences are not a rule (sentences and particularly paragraphs, can often be found split across separate sub-trees), they provide valuable high-level clues, particularly in the task of scoring span relation priority (classifier S):

Ex.: *"Belong to same sentence"*$_F$, *"Belong to same paragraph"*$_F$, *"Number of paragraph boundaries"*$_S$, *"Number of sentence boundaries"*$_S$...

As pointed out by Reitter (Reitter, 2003a), we can hypothesize a correlation between span length and some relations (for example, the satellite in a CONTRAST relation will tend to be shorter than the nucleus). Therefore, it seems useful to encode different measures of span size and positioning, using either tokens or edus as a distance unit:

Ex.: *"Length in tokens"*$_S$, *"Length in* edus*"*$_S$, *"Distance to beginning of sentence in tokens"*$_S$, *"Size of span over sentence in* edus*"*$_S$, *"Distance to end of sentence in tokens"*$_S$...

In order to better adjust to length variations between different types of text, some features in the above set are duplicated using relative, rather than absolute, values for positioning and distance.

3.2 Lexical Clues and Punctuation

While not always present, *discourse markers* (connectives, cue-words or cue-phrases, etc) have been shown to give good indications on discourse structure and labeling, particularly at the sentence-level (Marcu, 2000). We use an empirical n-gram dictionary (for $n \in \{1,2,3\}$) built from the training corpus and culled by frequency. As an advantage over explicit cue-words list, this method

also takes into account non-lexical signals such as punctuation and sentence/paragraph boundaries (inserted as artificial tokens in the original text during input formatting) which would otherwise necessitate a separate treatment.

We counted and encoded n-gram occurrences while considering only the first and last n tokens of each span. While raising the encoding size compared to a "bag of words" approach, this gave us significantly better performance (classifier accuracy improved by more than 5%), particularly when combined with main constituent features (see Sect. 3.5 below). This is consistent with the suggestion that most meaningful rhetorical signals are located on the edge of the span (Schilder, 2002).

We validated this approach by comparing it to results obtained with an explicit list of approximately 300 discourse-signaling cue-phrases (Oberlander et al., 1999): performance when using the list of cue-phrases alone was substantially lower than n-grams.

3.3 Simple Syntactic Clues

In order to complement signal detection and to achieve better generalization (smaller dependency on lexical content), we opted to add shallow syntactic clues by encoding part-of-speech (POS) tags for both prefix and suffix in each span. Using prefixes or suffixes of length higher than $n = 3$ did not seem to improve performance significantly.

3.4 Dominance Sets

A promising concept introduced by Soricut and Marcu (2003) in their sentence-level parser is the identification of 'dominance sets' in the syntax parse trees associated to each input sentence. For example, it could be difficult to correctly identify the scope of the ATTRIBUTION relation in the example shown in Fig. 3. By using the associated syntax tree and studying the sub-trees spanned by each edu (see Fig. 4), it is possible to quickly infer a logical nesting order ("dominance") between them: 1A $>$ 1B $>$ 1C. This order allows us to favor the relation between 1B and 1C over a relation between 1A and 1B, and thus helps us to make the right structural decision and pick the right-hand tree on Fig. 3.

In addition to POS tags around the frontier between each dominance set (see colored nodes in Fig. 4), Soricut and Marcu (2003) note that in order to achieve good results on relation labeling,

[Shoney's Inc. said]^{1A} [it will report a write-off of $2.5 million, or seven cents a share, for its fourth quarter]^{1B} [ended yesterday.]^{1C} (wsj$_{0667}$)

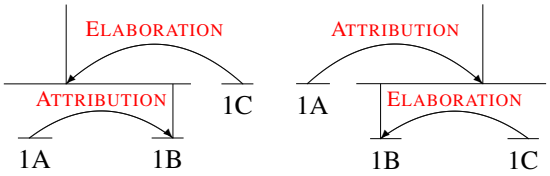

Figure 3: Two possible RST parses for a sentence.

it is necessary to also consider lexical information (obtained through head word projection of terminal nodes to higher internal nodes). Based on this definition of dominance sets, we include a set of syntactic, lexical and tree-structural features that aim at a good approximation of Marcu & Soricut's rule-based analysis of dominance sets while keeping parsing complexity low.

Ex.: *"Distance to root of the syntax tree"*$_S$, *"Distance to common ancestor in the syntax tree"*$_S$, *"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"*$_F$ (diamonds in Figure 4, *"Dominated node's POS tag"*$_F$ (circles in Figure 4), *"Dominated node's sibling's POS tag"*$_F$ (rectangles in Figure 4), *"Relative position of lexical head in sentence"*$_S$...

3.5 Strong Compositionality Criterion

We make use of Marcu's 'Strong Compositionality Criterion' (Marcu, 1996) through a very simple and limited set of features, replicating shallow lexical and syntactic features (previously described in Sections 3.2 and 3.3) on a single representative edu (dubbed *main constituent*) for each span. Main constituents are selected recursively using nuclearity information. We purposely keep the number of features extracted from main constituents comparatively low (therefore limiting the extra dimensionality cost), as we believe our use of rhetorical sub-structures ultimately encodes a variation of Marcu's compositionality criterion (see Sect. 3.6).

3.6 Rhetorical Sub-structure

A large majority of the features considered so far focus exclusively on sentence-level information.

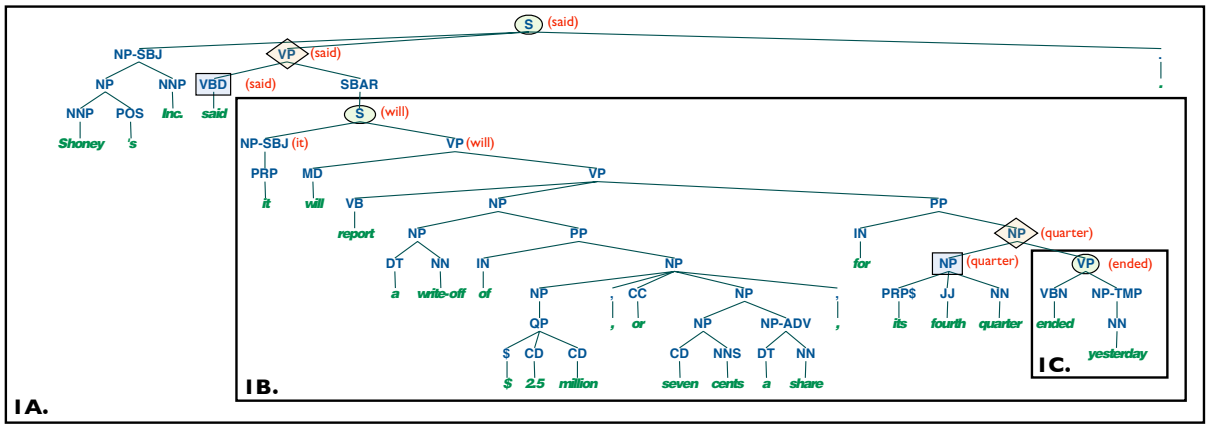

Figure 4: Using dominance sets to prioritize structural relations.
Circled nodes define dominance sets and studying the frontiers between circles and diamonds gives us a dominance order between each of the three sub-trees considered: 1A $>$ 1B $>$ 1C. Head words obtained through partial lexicalization have been added between parenthesis.

In order to efficiently label higher-level relations, we need more structural features that can guide good classification decision on large spans. Hence the idea of encoding each span's rhetorical subtree into the feature vector seems natural.

Beside the role of nuclearity in the sub-structure implied by Marcu's compositionality criterion (see Sect. 3.5), we expect to see certain correlations between the relation being classified and relation patterns in either sub-tree, based on theoretical considerations and practical observations. The original RST theory suggests the use of 'schemas' as higher-order patterns of relations motivated by linguistic theories and verified through empirical analysis of annotated trees (Mann and Thompson, 1988). In addition, some level of correlation between relations at different levels of the tree can be informally observed throughout the corpus. This is trivially the case for n-ary relations such as LIST which have been binarized in our representation, i.e., the presence of several LIST relations in rightmost nodes of a subtree greatly increases the probability that the parent relation might be a LIST itself.

4 Evaluation

4.1 General Considerations

In looking to evaluate the performance of our system, we had to work with a number of constraints and difficulties tied to variations in the methodologies used across past works, as well as a lack of consensus with regard to a common evaluation corpus. In order to accommodate these divergences while providing figures to evaluate both relative and absolute performance of our algorithm, we used three different test sets. Absolute performance is measured on the official test subset of the RST-DT corpus. A similarly available subset of doubly-annotated documents from the RST-DT is used to compare results with human agreement on the same task. Lastly, performance against past algorithms is evaluated with another subset of the RST-DT, such as used by LeThanh *et al.* (2004) in their own evaluation.

4.2 Raw SVM Classification

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 raw performance of SVM classifiers. Binary classifier S is trained on 52,683 instances (split approximately $1/3$, $2/3$ between positive and negative examples), extracted from 350 documents, and tested on 8,558 instances extracted from 50 documents. The feature space dimension is 136,987. Classifier L is trained on 17,742 instances (labeled across 41 classes) and tested on 2,887 instances, of same dimension as for S.

Classifier:	**Binary** (S)			**Multi-label** (L)		**Reitter**
Kernel	Linear	Polyn.	RBF	Linear	RBF	RBF
Software	*liblinear*	svm_{light}	svm_{light}	$svm_{multiclass}$	*libsvm*	svm_{light}
Multi-label	-			C&S	1 vs. 1	1 vs. all
Training time	21.4s	5m53s	12m	15m	23m	216m
Accuracy	82.2	85.0	82.9	65.8	66.8	61.0

Table 1: SVM Classifier performance. Regarding 'Multi-label', see Sect. 2.2.

The noticeably good performance of linear

kernel methods in the results presented in Table 1 compared to more complex polynomial and RBF kernels, would indicate that our data separates fairly well linearly: a commonly observed effect of high-dimensional input (Chen et al., 2007) such as ours ($>$ 100,000 features).

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

Taking into account matters of performance and runtime complexity, we selected a linear kernel for S and an optimally parameterized RBF kernel for L, using modified versions of the *liblinear* and *libsvm* software packages. All further evaluations noted here were conducted with these.

4.3 Full System Performance

A measure of our full system's performance is realized by comparing structure and labeling of the RST tree produced by our algorithm to that obtained through manual annotation (our gold standard). Standard performance indicators for such a task are *precision*, *recall* and *F-score* as measured by the PARSEVAL metrics (Black et al., 1991), with the specific adaptations to the case of RST trees made by Marcu (2000, page 143-144).

Our first evaluation (see Table 2) was conducted using the standard test subset of 41 files provided by the RST-DT corpus. In order to more accurately compare our results to the gold standard (defined as manual agreement between human annotators), we also evaluated performance using the 52 doubly-annotated files present in the RST-DT as test set (see Table 3). In each case, the remaining 340–350 files are used for training.

For each corpus evaluation, the system is run twice: once using perfectly-segmented input (taken from the RST-DT), and once using the output of the SPADE segmenter (Soricut and Marcu, 2003). The first measure gives us a good idea of our system's optimal performance (given optimal input), while the other gives us a more real-world evaluation, apt for comparison with other systems.

In each case, parse trees are evaluated using the four following, increasingly complex, matching criteria: blank tree structure ('S'), tree structure with nuclearity ('N'), tree structure with rhetorical relations ('R') and our final goal: fully labeled structure with both nuclearity and rhetorical relation labels ('F').

Segment.	**Manual**				**SPADE**			
	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
Recall	83.0	68.4	55.3	54.8	69.2	55.8	44.7	44.2
F-Score	83.0	68.4	55.3	54.8	69.3	56.0	44.8	44.3

Table 2: Discourse-parser evaluation depending on segmentation using standard test subset

	System performance								**Human agreement**			
Segment.	**Manual**				**SPADE**				**-**			
	S	N	R	F	S	N	R	F	S	N	R	F
Precision	84.1	70.6	55.6	55.1	70.6	58.1	46.0	45.6	88.0	77.5	66.0	65.2
Recall	84.1	70.6	55.6	55.1	71.2	58.6	46.4	46.0	88.1	77.6	66.1	65.3
F-Score	84.1	70.6	55.6	**55.1**	70.9	58.3	46.2	**45.8**	88.1	77.5	66.0	**65.3**

Table 3: Comparing to human-agreement depending on segmentation using doubly-annotated subset

Note: When using perfect segmentation, precision and recall are identical since both trees have same number of constituents.

4.4 Comparison with other Algorithms

To the best of our knowledge, only two fully functional text-level discourse parsing algorithms for general text have published their results: Marcu's decision-tree-based parser (Marcu, 2000) and the multi-level rule-based system built by LeThanh *et al.* (2004). For each one, evaluation was conducted on a different corpus, using unavailable documents for Marcu's and a selection of 21 documents from the RST-DT (distinct from RST-DT's test set) for LeThanh's. We therefore retrained and evaluated our classifier, using LeThanh's set of 21 documents as testing subset (and the rest for training) and compared performance (see Table 4). In order to achieve the most uniform conditions possible, we use LeThanh's results on 14 classes (Marcu's use 15, ours 18) and select SPADE segmentation figures for both our system and Marcu's (LeThanh's

system uses its own segmenter and does not provide figures for perfectly segmented input).

	Structure			Nuclearity			Relations		
Algorithm	*M*	*lT*	*dV*	*M*	*lT*	*dV*	*M*	*lT*	*dV*
Precision	65.8	54.5	**72.4**	54.0	47.8	**57.8**	34.3	40.5	**47.8**
Recall	34.0	52.9	**73.3**	21.6	46.4	**58.5**	13.0	39.3	**48.4**
F-score	44.8	53.7	**72.8**	30.9	47.1	**58.1**	18.8	39.9	**48.1**

Table 4: Side-by-side text-level algorithms comparison: Marcu (*M*), LeThanh *et al.* (*lT*) and ours (*dV*)

Some discrepancies between reported human agreement F-scores suggest that, despite our best efforts, evaluation metrics used by each author might differ. Another explanation may lie in discrepancies between training/testing subsets used. In order to take into account possibly varying levels of difficulties between corpora, we therefore divided each F-score by the value for human agreement, such as measured by each author (see Table 5). This ratio should give us a fairer measure of success for the algorithm taking into account how well it succeeds in reaching near-human level.

	Structure			Nuclearity			Relations		
Algorithm	*M*	*lT*	*dV*	*M*	*lT*	*dV*	*M*	*lT*	*dV*
$\frac{\text{F-score}_{\text{algo}}}{\text{F-score}_{\text{human}}}$	56.0	73.9	**83.0**	42.9	71.8	**75.6**	25.7	70.1	**73.9**

Table 5: Performance scaled by human agreement scores: Marcu (*M*), LeThahn *et al.* (*lT*) and ours (*dV*)

Table 5 shows 83%, 75.6% and 73.9% of human agreement F-scores in structure, nuclearity and relation parsing, respectively. Qualified by the (practical) problems of establishing comparison conditions with scientific rigor, the scores indicate that our system outperforms the previous state-of-the-art (LeThanh's 73.9%, 71.8% and 70.1%). As suggested by previous research (Soricut and Marcu, 2003), these scores could likely be further improved with the use of better-performing segmenting algorithms. It can however be noted that our system seems considerably less sensitive to imperfect segmenting than previous efforts. For instance, when switching from manual segmentation to automatic, our performance decreases by 12.3% and 12.9% (respectively for structure and relation F-scores) compared to 46% and 67% for Marcu's system (LeThanh's performance on perfect input is unknown).

5 Conclusions and Future Work

In this paper, we have shown that it is possible to build an accurate automatic text-level discourse parser based on supervised machine-learning algorithms, using a feature-driven approach and a manually annotated corpus. Importantly, our system achieves its accuracy in linear complexity of the input size with excellent runtime performance. The entire test subset in the RST-DT corpus could be fully annotated in a matter of minutes. This opens the way to many novel applications in real-time natural language processing and generation, such as the RST-based transformation of monological text into dialogues acted by virtual agents in real-time (Hernault et al., 2008).

Future directions for this work notably include a better tree-building algorithm, with improved exploration of the solution space. Borrowing techniques from generic global optimization meta-algorithms such as simulated annealing (Kirkpatrick et al., 1983) should allow us to better deal with issues of local optimality while retaining acceptable time-complexity.

A complete online discourse parser, incorporating the parsing tool presented above combined with a new segmenting method has since been made freely available at `http://nlp.prendingerlab.net/hilda/`.

Acknowledgements

This project was jointly funded by Prendinger Lab (NII, Tokyo) and the National Institute for Informatics (Tokyo), as part of a MOU (Memorandum of Understanding) program with Pierre & Marie Curie University (Paris).

References

M.A. Aizerman, E.M. Braverman, and L.I. Rozonoer. 1964. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25(6):821–837.

N. Asher and A. Lascarides. 2003. *Logics of conversation*. Cambridge University Press.

J. Baldridge and A. Lascarides. 2005. Probabilistic head-driven parsing for discourse structure. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, volume 96, page 103.

E. Black, S. Abney, S. Flickenger, C. Gdaniec, C. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, et al. 1991. Procedure for quantitatively comparing the syntactic coverage of English grammars. *Proceedings of the workshop on Speech and Natural Language*, pages 306–311.

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

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

M. Collins. 2003. Head-Driven Statistical Models for Natural Language Parsing. *Computational Linguistics*, 29(4):589–637.

K. Crammer and Y. Singer. 2002. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292.

H. Hernault, P. Piwek, H. Prendinger, and M. Ishizuka. 2008. Generating dialogues for virtual agents using nested textual coherence relations. *Proceedings of the 8th International Conference on Intelligent Virtual Agents (IVA'08), LNAI*, 5208:139–145, Sept.

S. Kirkpatrick, CD Gelatt, and MP Vecchi. 1983. Optimization by Simulated Annealing. *Science*, 220(4598):671–680.

H. LeThanh, G. Abeysinghe, and C. Huyck. 2004. Generating discourse structures for written texts. *Proceedings of the 20th international conference on Computational Linguistics*.

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

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

D. Marcu. 1996. Building Up Rhetorical Structure Trees. *Proceedings of the National Conference on Artificial Intelligence*, pages 1069–1074.

D. Marcu. 2000. *The theory and practice of discourse parsing and summarization*. MIT Press.

J. Oberlander, J.D. Moore, J. Oberlander, A. Knott, and J. Moore. 1999. Cue phrases in discourse: further evidence for the core: contributor distinction. *Proceedings of the 1999 Levels of Representation in Discourse Workshop (LORID'99)*, pages 87–93.

P. Piwek, H. Hernault, H. Prendinger, and M. Ishizuka. 2007. Generating dialogues between virtual agents automatically from text. *Proceedings of the 7th International Conference on Intelligent Virtual Agents (IVA '07), LNCS*, 4722:161.

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

D. Reitter. 2003b. Simple Signals for Complex Rhetorics: On Rhetorical Analysis with Rich-Feature Support Vector Models. *Language*, 18(52).

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

B. Scholkopf, C. Burges, and V. Vapnik. 1995. Extracting Support Data for a Given Task. *Knowledge Discovery and Data Mining*, pages 252–257.

R. Soricut and D. Marcu. 2003. 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.

C. Staelin. 2003. Parameter selection for support vector machines. *Hewlett-Packard Company, Tech. Rep. HPL-2002-354R1*.

V.N. Vapnik. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA.