Text-level Discourse Parsing with Rich Linguistic Features

Vanessa Wei Feng
Department of Computer Science
University of Toronto
Toronto, ON, M5S 3G4, Canada
weifeng@cs.toronto.edu

Graeme Hirst
Department of Computer Science
University of Toronto
Toronto, ON, M5S 3G4, Canada
gh@cs.toronto.edu

Abstract

In this paper, we develop an RST-style text-level discourse parser, based on the HILDA discourse parser (Hernault et al., 2010b). We significantly improve its tree-building step by incorporating our own rich linguistic features. We also analyze the difficulty of extending traditional sentence-level discourse parsing to text-level parsing by comparing discourse-parsing performance under different discourse conditions.

1 Introduction

In a well-written text, no unit of the text is completely isolated; interpretation requires understanding the unit's relation with the context. Research in discourse parsing aims to unmask such relations in text, which is helpful for many downstream applications such as summarization, information retrieval, and question answering.

However, most existing discourse parsers operate on individual sentences alone, whereas discourse parsing is more powerful for text-level analysis. Therefore, in this work, we aim to develop a text-level discourse parser. We follow the framework of Rhetorical Structure Theory (Mann and Thompson, 1988) and we take the HILDA discourse parser (Hernault et al., 2010b) as the basis of our work, because it is the first fully implemented text-level discourse parser with state-of-the-art performance. We significantly improve the performance of HILDA's tree-building step (introduced in Section 5.1 below) by incorporating rich linguistic features (Section 5.3). In our experiments (Section 6), we also analyze the difficulty with extending traditional sentence-level discourse parsing to text-level parsing, by comparing discourse parsing performance under different discourse conditions.

2 Discourse-annotated corpora

2.1 The RST Discourse Treebank

Rhetorical Structure Theory (Mann and Thompson, 1988) is one of the most widely accepted frameworks for discourse analysis. In the framework of RST, a coherent text can be represented as a discourse tree whose leaves are non-overlapping text spans called *elementary discourse units* (EDUs); these are the minimal text units of discourse trees. Adjacent nodes can be related through particular discourse relations to form a discourse subtree, which can then be related to other adjacent nodes in the tree structure. According to RST, there are two types of discourse relations, hypotactic ("mononuclear") and paratactic ("multi-nuclear"). In mononuclear relations, one of the text spans, the *nucleus*, is more salient than the other, the *satellite*, while in multi-nuclear relations, all text spans are equally important for interpretation.

The example text fragment shown in Figure 1 consists of four EDUs (e_1-e_4), segmented by square brackets. Its discourse tree representation is shown below in the figure, following the notational convention of RST. The two EDUs e_1 and e_2 are related by a mononuclear relation ATTRIBUTION, where e_1 is the more salient span; the span (e_1-e_2) and the EDU e_3 are related by a multi-nuclear relation SAME-UNIT, where they are equally salient.

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 60–68,
Jeju, Republic of Korea, 8-14 July 2012. ©2012 Association for Computational Linguistics

[Catching up with commercial competitors in retail banking and financial services,]e_1 [they argue,]e_2 [will be difficult,]e_3 [particularly if market conditions turn sour.]e_4

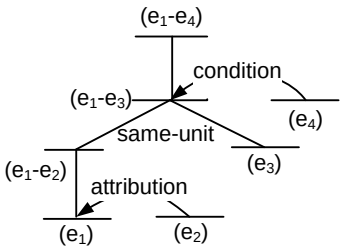

Figure 1: An example text fragment (wsj_0616) composed of four EDUs, and its RST discourse tree representation.

The RST Discourse Treebank (RST-DT) (Carlson et al., 2001), is a corpus annotated in the framework of RST. It consists of 385 documents (347 for training and 38 for testing) from the *Wall Street Journal*. In RST-DT, the original 24 discourse relations defined by Mann and Thompson (1988) are further divided into a set of 18 relation classes with 78 finer-grained rhetorical relations in total, which provides a high level of expressivity.

2.2 The Penn Discourse Treebank

The Penn Discourse Treebank (PDTB) (Prasad et al., 2008) is another annotated discourse corpus. Its text is a superset of that of RST-DT (2159 *Wall Street Journal* articles). Unlike RST-DT, PDTB does not follow the framework of RST; rather, it follows a lexically grounded, predicate-argument approach with a different set of predefined discourse relations, as proposed by Webber (2004). In this framework, a discourse connective (e.g., *because*) is considered to be a predicate that takes two text spans as its arguments. The argument that the discourse connective structurally attaches to is called **Arg2**, and the other argument is called **Arg1** — unlike in RST, the two arguments are not distinguished by their saliency for interpretation. Another important difference between PDTB and RST-DT is that in PDTB, there does not necessarily exist a tree structure covering the full text, i.e., PDTB-styled discourse relations exist only in a very local contextual window. In PDTB, relation types are organized hierarchically: there are 4 *classes*, which can be further divided into 16 *types* and 23 *subtypes*.

3 Related work

Discourse parsing was first brought to prominence by Marcu (1997). Since then, many different algorithms and systems (Soricut and Marcu, 2003; Reitter, 2003; LeThanh et al., 2004; Baldridge and Lascarides, 2005; Subba and Di Eugenio, 2009; Sagae, 2009; Hernault et al., 2010b) have been proposed, which extracted different textual information and adopted various approaches for discourse tree building. Here we briefly review two fully implemented text-level discourse parsers with the state-of-the-art performance.

The HILDA discourse parser of Hernault and his colleagues (duVerle and Prendinger, 2009; Hernault et al., 2010b) is the first fully-implemented feature-based discourse parser that works at the full text level. Hernault et al. extracted a variety of lexical and syntactic features from the input text, and trained their system on RST-DT. While some of their features were inspired by the previous work of others, e.g., lexico-syntactic features borrowed from Soricut and Marcu (2003), Hernault et al. also proposed the novel idea of discourse tree building by using two classifiers in cascade — a binary structure classifier to determine whether two adjacent text units should be merged to form a new subtree, and a multi-class classifier to determine which discourse relation label should be assigned to the new subtree — instead of the more-usual single multi-class classifier with the additional label NO-REL. Hernault et al. obtained 93.8% F-score for EDU segmentation, 85.0% accuracy for structure classification, and 66.8% accuracy for 18-class relation classification.

Lin et al. (2009) attempted to recognize implicit discourse relations (discourse relations which are not signaled by explicit connectives) in PDTB by using four classes of features — contextual features, constituent parse features, dependency parse features, and lexical features — and explored their individual influence on performance. They showed that the production rules extracted from constituent parse trees are the most effective features, while contextual features are the weakest. Subsequently, they fully implemented an end-to-end PDTB-style discourse parser (Lin et al., 2010).

Recently, Hernault et al. (2010a) argued that more effort should be focused on improving performance

on certain infrequent relations presented in the discourse corpora, since due to the imbalanced distribution of different discourse relations in both RST-DT and PDTB, the overall accuracy score can be overwhelmed by good performance on the small subset of frequent relations, even though the algorithms perform poorly on all other relations. However, because of infrequent relations for which we do not have sufficient instances for training, many unseen features occur in the test data, resulting in poor test performance. Therefore, Hernault et al. proposed a semi-supervised method that exploits abundant, freely-available unlabeled data as a basis for feature vector extension to alleviate such issues.

4 Text-level discourse parsing

Not until recently has discourse parsing for full texts been a research focus — previously, discourse parsing was only performed on the sentence level[1]. In this section, we explain why we believe text-level discourse parsing is crucial.

Unlike syntactic parsing, where we are almost never interested in parsing above sentence level, sentence-level parsing is not sufficient for discourse parsing. While a sequence of local (sentence-level) grammaticality can be considered to be global grammaticality, a sequence of local discourse coherence does not necessarily form a globally coherent text. For example, the text shown in Figure 2 contains two sentences, each of which is coherent and sensible itself. However, there is no reasonable content transition between these two sentences, so the combination of the two sentences does not make much sense. If we attempt to represent the text as an RST discourse tree like the one shown in Figure 1, we find that no discourse relation can be assigned to relate the spans $(e_1\text{-}e_2)$ and $(e_3\text{-}e_4)$ and the text cannot be represented by a valid discourse tree structure.

In order to rule out such unreasonable transitions between sentences, we have to expand the text units upon which discourse parsing is performed: from sentences to paragraphs, and finally paragraphs to the full text.

[No wonder he got an A for his English class,]e_1 [he was studying so hard.]e_2 [He avoids eating chocolates,]e_3 [since he is really worried about gaining weight.]e_4

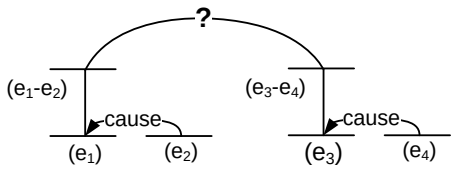

Figure 2: An example of incoherent text fragment composed of two sentences. The two EDUs associated with each sentence are coherent themselves, whereas the combination of the two sentences is not coherent at the sentence boundary. No discourse relation can be associated with the spans $(e_1\text{-}e_2)$ and $(e_3\text{-}e_4)$.

Text-level discourse parsing imposes more constraints on the global coherence than sentence-level discourse parsing. However, if, technically speaking, text-level discourse parsing were no more difficult than sentence-level parsing, any sentence-level discourse parser could be easily upgraded to a text-level discourse parser just by applying it to full texts. In our experiments (Section 6), we show that when applied above the sentence level, the performance of discourse parsing is consistently inferior to that within individual sentences, and we will briefly discuss what the key difficulties with extending sentence-level to text-level discourse parsing are.

5 Method

We use the HILDA discourse parser of Hernault et al. (2010b) as the basis of our work. We refine Hernault et al.'s original feature set by incorporating our own features as well as some adapted from Lin et al. (2009). We choose HILDA because it is a fully implemented text-level discourse parser with the best reported performance up to now. On the other hand, we also follow the work of Lin et al. (2009), because their features can be good supplements to those used by HILDA, even though Lin et al.'s work was based on PDTB. More importantly, Lin et al.'s strategy of performing feature selection prior to classification proves to be effective in reducing the total feature dimensions, which is favorable since we wish to incorporate rich linguistic features into our discourse parser.

[1]Strictly speaking, for PDTB-style discourse parsing (e.g., Lin et al. (2009; 2010)), there is no absolute distinction between sentence-level and text-level parsing, since in PDTB, discourse relations are annotated at a level no higher than that of adjacent sentences. Here we are concerned with RST-style discourse parsing.

5.1 Bottom-up approach and two-stage labeling step

Following the methodology of HILDA, an input text is first segmented into EDUs. Then, from the EDUs, a bottom-up approach is applied to build a discourse tree for the full text. Initially, a binary *Structure* classifier evaluates whether a discourse relation is likely to hold between consecutive EDUs. The two EDUs which are most probably connected by a discourse relation are merged into a discourse subtree of two EDUs. A multi-class *Relation* classifier evaluates which discourse relation label should be assigned to this new subtree. Next, the *Structure* classifier and the *Relation* classifier are employed in cascade to re-evaluate which relations are the most likely to hold between adjacent spans (discourse subtrees of any size, including atomic EDUs). This procedure is repeated until all spans are merged, and a discourse tree covering the full text is therefore produced.

Since EDU boundaries are highly correlated with the syntactic structures embedded in the sentences, EDU segmentation is a relatively trivial step — using machine-generated syntactic parse trees, HILDA achieves an F-score of 93.8% for EDU segmentation. Therefore, our work is focused on the tree-building step, i.e., the *Structure* and the *Relation* classifiers. In our experiments, we improve the overall performance of these two classifiers by incorporating rich linguistic features, together with appropriate feature selection. We also explore how these two classifiers perform differently under different discourse conditions.

5.2 Instance extraction

Because HILDA adopts a bottom-up approach for discourse tree building, errors produced on lower levels will certainly propagate to upper levels, usually causing the final discourse tree to be very dissimilar to the gold standard. While appropriate post-processing may be employed to fix these errors and help global discourse tree recovery, we feel that it might be more effective to directly improve the raw instance performance of the *Structure* and *Relation* classifiers. Therefore, in our experiments, all classifications are conducted and evaluated on the basis of individual instances.

Each instance is of the form (S_L, S_R), which is a pair of adjacent text spans S_L (left span) and S_R (right span), extracted from the discourse tree representation in RST-DT. From each discourse tree, we extract positive instances as those pairs of text spans that are siblings of the same parent node, and negative examples as those pairs of adjacent text spans that are not siblings in the tree structure. In all instances, both S_L and S_R must correspond to a constituent in the discourse tree, which can be either an atomic EDU or a concatenation of multiple consecutive EDUs.

5.3 Feature extraction

Given a pair of text spans (S_L, S_R), we extract the following seven types of features.

HILDA's features: We incorporate the original features used in the HILDA discourse parser with slight modification, which include the following four types of features occurring in S_L, S_R, or both: (1) N-gram prefixes and suffixes; (2) syntactic tag prefixes and suffixes; (3) lexical heads in the constituent parse tree; and (4) POS tag of the dominating nodes.

Lin et al.'s features: Following Lin et al. (2009), we extract the following three types of features: (1) pairs of words, one from S_L and one from S_R, as originally proposed by Marcu and Echihabi (2002); (2) dependency parse features in S_L, S_R, or both; and (3) syntactic production rules in S_L, S_R, or both.

Contextual features: For a globally coherent text, there exist particular sequential patterns in the local usage of different discourse relations. Given (S_L, S_R), the pair of text spans of interest, contextual features attempt to encode the discourse relations assigned to the preceding and the following text span pairs. Lin et al. (2009) also incorporated contextual features in their feature set. However, their work was based on PDTB, which has a very different annotation framework from RST-DT (see Section 2): in PDTB, annotated discourse relations can form a chain-like structure such that contextual features can be more readily extracted. However, in RST-DT, a full text is represented as a discourse tree structure, so the *previous* and the *next* discourse relations are not well-defined.

We resolve this problem as follows. Suppose $S_L = (e_i\text{-}e_j)$ and $S_R = (e_{j+1}\text{-}e_k)$, where $i \leq j < k$. To find the previous discourse relation REL_{prev} that immedi-

ately precedes (S_L, S_R), we look for the largest span $S_{prev} = (e_h\text{-}e_{i-1}), h < i$, such that it ends right before S_L and all its leaves belong to a single subtree which neither S_L nor S_R is a part of. If S_L and S_R belong to the same sentence, S_{prev} must also be a *within-sentence* span, and it must be a *cross-sentence* span if S_L and S_R are a cross-sentence span pair. REL_{prev} is then the discourse relation which covers S_{prev}. The next discourse relation REL_{next} that immediately follows (S_L, S_R) is found in the analogous way.

However, when building a discourse tree using a greedy bottom-up approach, as adopted by the HILDA discourse parser, REL_{prev} and REL_{next} are not always available; therefore these contextual features represent an idealized situation. In our experiments we wish to explore whether incorporating perfect contextual features can help better recognize discourse relations, and if so, set an upper bound of performance in more realistic situations.

Discourse production rules: Inspired by Lin et al. (2009)'s syntactic production rules as features, we develop another set of production rules, namely discourse production rules, derived directly from the tree structure representation in RST-DT.

For example, with respect to the RST discourse tree shown in Figure 1, we extract the following discourse production rules: ATTRIBUTION → NO-REL NO-REL, SAME-UNIT → ATTRIBUTION NO-REL, CONDITION → SAME-UNIT NO-REL, where NO-REL denotes a leaf node in the discourse subtree.

The intuition behind using discourse production rules is that the discourse tree structure is able to reflect the relatedness of different discourse relations — discourse relations on the lower level of the tree can determine the relation of their direct parent to some degree. Hernault et al. (2010b) attempt to capture such relatedness by traversing a discourse subtree and encoding its traversal path as features, but since they used a depth-first traversal order, the information encoded in a node's direct children is too distant; whereas most useful information can be gained from the relations covering these direct children.

Semantic similarities: Semantic similarities are useful for recognizing relations such as COMPARISON, when there are no explicit syntactic structures or lexical features signaling such relations.

We use two subsets of similarity features for verbs and nouns separately. For each verb in either S_L or S_R, we look up its most frequent verb class ID in VerbNet[2], and specify whether that verb class ID appears in S_L, S_R, or both. For nouns, we extract all pairs of nouns from (S_L, S_R), and compute the average similarity among these pairs. In particular, we use *path_similarity*, *lch_similarity*, *wup_similarity*, *res_similarity*, *jcn_similarity*, and *lin_similarity* provided in the *nltk.wordnet.similarity* package (Bird et al., 2009) for computing WordNet-based similarity, and always choose the most frequent sense for each noun.

Cue phrases: We compile a list of cue phrases, the majority of which are connectives collected by Knott and Dale (1994). For each cue phrase in this list, we determine whether it appears in S_L or S_R. If a cue phrase appears in a span, we also determine whether its appearance is in the beginning, the end, or the middle of that span.

5.4 Feature selection

If we consider all possible combinations of the features listed in Section 5.3, the resulting data space can be horribly high dimensional and extremely sparse. Therefore, prior to training, we first conduct feature selection to effectively reduce the dimension of the data space.

We employ the same feature selection method as Lin et al. (2009). Feature selection is done for each feature type separately. Among all features belonging to the feature type to be selected, we first extract all possible features that have been seen in the training data, e.g., when applying feature selection for *word pairs*, we find all word pairs that appear in some text span pair that have a discourse relation between them. Then for each extracted feature, we compute its mutual information with all 18 discourse relation classes defined in RST-DT, and use the highest mutual information to evaluate the effectiveness of that feature. All extracted features are sorted to form a ranked list by effectiveness. After that, we use a threshold to select the top features from that ranked list. The total number of selected features used in our experiments is 21,410.

[2] http://verbs.colorado.edu/~mpalmer/projects/verbnet

6 Experiments

As discussed in Section 5.1, our research focus in this paper is the tree-building step of the HILDA discourse parser, which consists of two classifications: *Structure* and *Relation* classification. The binary *Structure* classifier decides whether a discourse relation is likely to hold between consecutive text spans, and the multi-class *Relation* classifier decides which discourse relation label holds between these two text spans if the *Structure* classifier predicts the existence of such a relation.

Although HILDA's bottom-up approach is aimed at building a discourse tree for the full text, it does not explicitly employ different strategies for *within-sentence* text spans and *cross-sentence* text spans. However, we believe that discourse parsing is significantly more difficult for text spans at higher levels of the discourse tree structure. Therefore, we conduct the following three sub-experiments to explore whether the two classifiers behave differently under different discourse conditions.

Within-sentence: Trained and tested on text span pairs belonging to the same sentence.

Cross-sentence: Trained and tested on text span pairs belonging to different sentences.

Hybrid: Trained and tested on all text span pairs.

In particular, we split the training set and the testing set following the convention of RST-DT, and conduct *Structure* and *Relation* classification by incorporating our rich linguistic features, as listed in Section 5.3 above. To rule out all confounding factors, all classifiers are trained and tested on the basis of individual text span pairs, by assuming the discourse subtree structure (if any) covering each individual text span has been already correctly identified (no error propagation).

6.1 Structure classification

The number of training and testing instances used in this experiment for different discourse conditions is listed in Table 1. Instances are extracted in the manner described in Section 5.2. We observe that the distribution of positive and negative instances is extremely skewed for cross-sentence instances, while for all conditions, the distribution is similar in the training and the testing set.

In this experiment, classifiers are trained using the SVM^{perf} classifier (Joachims, 2005) with a linear kernel.

	Dataset	Pos #	Neg #	Total #
Within	Training	11,087	10,188	21,275
	Testing	1,340	1,181	2,521
Cross	Training	6,646	49,467	56,113
	Testing	882	6,357	7,239
Hybrid	Training	17,733	59,655	77,388
	Testing	2,222	7,539	9,761

Table 1: Number of training and testing instances used in *Structure classification*.

Structure classification performance for all three discourse conditions is shown in Table 2. The columns **Full** and **NC** (No Context) denote the performance of using all features listed in Section 5.3 and all features except for *contextual features* respectively. As discussed in Section 5.3, contextual features represent an ideal situation which is not always available in real applications; therefore, we wish to see how they affect the overall performance by comparing the performance obtained with them and without them as features. The column **HILDA** lists the performance of using Hernault et al. (2010b)'s original features, and **Baseline** denotes the performance obtained by always picking the more frequent class. Performance is measured by four metrics: accuracy, precision, recall, and F_1 score on the test set, shown in the first section in each sub-table.

Under the within-sentence condition, we observe that, surprisingly, incorporating contextual features boosts the overall performance by a large margin, even though it requires only 38 additional features. Under the cross-sentence condition, our features result in lower accuracy and precision than HILDA's features. However, under this discourse condition, the distribution of positive and negative instances in both training and test sets is extremely skewed, which makes it more sensible to compare the recall and F_1 scores for evaluation. In fact, our features achieve much higher recall and F_1 score despite a much lower precision and a slightly lower accuracy.

In the second section of each sub-table, we also list the F_1 score on the training data. This allows

us to compare the model-fitting capacity of different feature sets from another perspective, especially when the training data is not sufficiently well fitted by the model. For example, looking at the training F_1 score under the cross-sentence condition, we can see that classification using full features and classification without contextual features both perform significantly better on the training data than HILDA does. At the same time, such superior performance is not due to possible over-fitting on the training data, because we are using significantly fewer features (21,410 for **Full** and 21,372 for **NC**) than Hernault et al. (2010b)'s 136,987; rather, it suggests that using carefully selected rich linguistic features is able to better model the problem itself.

Comparing the results obtained under the first two conditions, we see that the binary classification problem of whether a discourse relation is likely to hold between two adjacent text spans is much more difficult under the cross-sentence condition. One major reason is that many features that are predictive for within-sentence instances are no longer applicable (e.g., *Dependency parse features*). In addition, given the extremely imbalanced nature of the dataset under this discourse condition, we might need to employ special approaches to deal with this needle-in-a-haystack problem. This difficulty can also be perceived from the training performance. Compared to the within-sentence condition, all features fit the training data much more poorly under the cross-sentence condition. This suggests that sophisticated features or models in addition to our rich linguistic features must be incorporated in order to fit the problem sufficiently well. Unfortunately, this underfitting issue cannot be resolved by exploiting any abundant linguistic resources for feature vector extension (e.g., Hernault et al. (2010a)), because the poor training performance is no longer caused by the unknown features found in test vectors.

Turning to the hybrid condition, the performance of **Full** features is surprisingly good, probably because we have more available training data than the other two conditions. However, with contextual features removed, our features perform quite similarly to those of Hernault et al. (2010b), but still with a marginal, but nonetheless statistically significant, improvement on recall and F_1 score.

	Full	**NC**	**HILDA**	**Baseline**
		Within-sentence		
Accuracy	**91.04***	85.17*	83.74	53.15
Precision	**92.71***	85.36*	84.81	53.15
Recall	**90.22***	87.01*	84.55	100.00
F_1	**91.45***	86.18*	84.68	69.41
Train F_1	**97.87***	96.23*	95.42	68.52
		Cross-sentence		
Accuracy	87.69	86.68	**89.13**	87.82
Precision	49.60	44.73	**61.90**	–
Recall	**63.95***	39.46*	28.00	0.00
F_1	**55.87***	41.93*	38.56	–
Train F_1	**87.25***	71.93*	49.03	–
		Hybrid		
Accuracy	**95.64***	87.03	87.04	77.24
Precision	**94.77***	74.19	79.41	–
Recall	**85.92***	65.98*	58.15	0.00
F_1	**89.51***	69.84*	67.13	–
Train F_1	**93.15***	80.79*	72.09	–

Table 2: *Structure* classification performance (in percentage) on text spans of *within-sentence*, *cross-sentence*, and *all* level. Performance that is significantly superior to that of HILDA ($p < .01$, using the Wilcoxon sign-rank test for significance) is denoted by *.

6.2 Relation classification

The *Relation* classifier has 18 possible output labels, which are the coarse-grained relation classes defined in RST-DT. We do not consider nuclearity when classifying different discourse relations, i.e., ATTRIBUTION[N][S] and ATTRIBUTION[S][N] are treated as the same label. The training and test instances in this experiment are from the positive subset used in *Structure* classification.

In this experiment, classifiers are trained using LibSVM classifier (Chang and Lin, 2011) with a linear kernel and probability estimation.

Relation classification performance under three discourse conditions is shown in Table 3. We list the performance achieved by **Full**, **NC**, and **HILDA** features, as well as the majority baseline, which is obtained by always picking the most frequent class label (ELABORATION in all cases).

	Full	NC	HILDA	Baseline
Within-sentence				
MAFS	**0.490**	0.485	0.446	–
WAFS	**0.763**	0.762	0.740	–
Acc (%)	78.06	**78.13**	76.42	31.42
TAcc (%)	99.90	**99.93**	99.26	33.38
Cross-sentence				
MAFS	**0.194**	0.184	0.127	–
WAFS	**0.334**	0.329	0.316	–
Acc (%)	**46.83**	46.71	45.69	42.52
TAcc (%)	**78.30**	67.30	57.70	47.79
Hybrid				
MAFS	**0.440**	0.428	0.379	–
WAFS	**0.607**	0.604	0.588	–
Acc (%)	**65.30**	65.12	64.18	35.82
TAcc (%)	**99.96**	99.95	90.11	38.78

Table 3: *Relation* classification performance on text spans of *within-sentence*, *cross-sentence*, and *all* levels.

Following Hernault et al. (2010a), we use Macro-averaged F-scores (MAFS) to evaluate the performance of each classifier. Macro-averaged F-score is not influenced by the number of instances that exist in each relation class, by equally weighting the performance of each relation class[3]. Therefore, the evaluation is not biased by the performance on those prevalent classes such as ATTRIBUTION and ELABORATION. For reasons of space, we do not show the class-wise F-scores, but in our results, we find that using our features consistently provides superior performance for most class relations over HILDA's features, and therefore results in higher overall MAFS under all conditions. We also list two other metrics for performance on the test data — Weight-averaged F-score (WAFS), which weights the performance of each relation class by the number of its existing instances, and the testing accuracy (Acc) — but these metrics are relatively more biased evaluation metrics in this task. Similar to *Structure* classification, the accuracy on the training data (TAcc)[4] is listed in the second section of each subtable. It demonstrates that our carefully selected rich linguistic features are able to better fit the classification problem, especially under the *cross-sentence* condition.

Similar to our observation in *Structure* classification, the performance of *Relation* classification for cross-sentence instances is also much poorer than that on within-sentence instances, which again reveals the difficulty of text-level discourse parsing.

7 Conclusions

In this paper, we aimed to develop an RST-style text-level discourse parser. We chose the HILDA discourse parser (Hernault et al., 2010b) as the basis of our work, and significantly improved its tree-building step by incorporating our own rich linguistic features, together with features suggested by Lin et al. (2009). We analyzed the difficulty of extending traditional sentence-level discourse parsing to text-level parsing by showing that using exactly the same set of features, the performance of *Structure* and *Relation* classification on cross-sentence instances is consistently inferior to that on within-sentence instances. We also explored the effect of contextual features on the overall performance. We showed that contextual features are highly effective for both *Structure* and *Relation* classification under all discourse conditions. Although perfect contextual features are available only in idealized situations, when they are correct, together with other features, they can almost correctly predict the tree structure and better predict the relation labels. Therefore, an iterative updating approach, which progressively updates the tree structure and the labeling based on the current estimation, may push the final results toward this idealized end.

Our future work will be to fully implement an end-to-end discourse parser using our rich linguistic features, and focus on improving performance on cross-sentence instances.

[3]No significance test is reported for relation classification, because we are comparing MAFS, which equally weights the performance of each relation. Therefore, traditional significance tests which operate on individual instances rather than individual relation classes are not applicable.

[4]We use accuracy instead of MAFS as the evaluation metric on the training data because it is the metric that the training procedure is optimized toward.

Acknowledgments

This work was financially supported by the Natural Sciences and Engineering Research Council of Canada and by the University of Toronto.

References

Jason Baldridge and Alex Lascarides. 2005. Probabilistic head-driven parsing for discourse structure. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 96–103.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python — Analyzing Text with the Natural Language Toolkit*. O'Reilly.

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

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:1–27.

David A. duVerle and Helmut Prendinger. 2009. 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, Volume 2*, ACL '09, pages 665–673, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hugo Hernault, Danushka Bollegala, and Mitsuru Ishizuka. 2010a. A semi-supervised approach to improve classification of infrequent discourse relations using feature vector extension. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 399–409, Cambridge, MA, October. Association for Computational Linguistics.

Hugo Hernault, Helmut Prendinger, David A. duVerle, and Mitsuru Ishizuka. 2010b. HILDA: A discourse parser using support vector machine classification. *Dialogue and Discourse*, 1(3):1–33.

Thorsten Joachims. 2005. A support vector method for multivariate performance measures. In *International Conference on Machine Learning (ICML)*, pages 377–384.

Alistair Knott and Robert Dale. 1994. Using linguistic phenomena to motivate a set of coherence relations. *Discourse Processes*, 18(1).

Huong LeThanh, Geetha Abeysinghe, and Christian Huyck. 2004. Generating discourse structures for written texts. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 329–335.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the Penn Discourse Treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, Volume 1*, EMNLP '09, pages 343–351.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2010. A PDTB-styled end-to-end discourse parser. Technical report, School of Computing, National University of Singapore.

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

Daniel Marcu and Abdessamad Echihabi. 2002. 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. Association for Computational Linguistics.

Daniel Marcu. 1997. The rhetorical parsing of natural language texts. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 96–103.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse Treebank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*.

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

Kenji Sagae. 2009. Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 81–84.

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

Rajen Subba and Barbara Di Eugenio. 2009. 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.

Bonnie Webber. 2004. D-LTAG: Extending lexicalized TAG to discourse. *Cognitive Science*, 28(5):751–779.