The utility of parse-derived features for automatic discourse segmentation

Seeger Fisher and Brian Roark
Center for Spoken Language Understanding, OGI School of Science & Engineering
Oregon Health & Science University, Beaverton, Oregon, 97006 USA
{fishers,roark}@cslu.ogi.edu

Abstract

We investigate different feature sets for performing automatic sentence-level discourse segmentation within a general machine learning approach, including features derived from either finite-state or context-free annotations. We achieve the best reported performance on this task, and demonstrate that our SPADE-inspired context-free features are critical to achieving this level of accuracy. This counters recent results suggesting that purely finite-state approaches can perform competitively.

1 Introduction

Discourse structure annotations have been demonstrated to be of high utility for a number of NLP applications, including automatic text summarization (Marcu, 1998; Marcu, 1999; Cristea et al., 2005), sentence compression (Sporleder and Lapata, 2005), natural language generation (Prasad et al., 2005) and question answering (Verberne et al., 2006). These annotations include sentence segmentation into discourse units along with the linking of discourse units, both within and across sentence boundaries, into a labeled hierarchical structure. For example, the tree in Figure 1 shows a sentence-level discourse tree for the string "Prices have dropped but remain quite high, according to CEO Smith," which has three discourse segments, each labeled with either "Nucleus" or "Satellite" depending on how central the segment is to the coherence of the text.

There are a number of corpora annotated with discourse structure, including the well-known RST Treebank (Carlson et al., 2002); the Discourse GraphBank (Wolf and Gibson, 2005); and the Penn Discourse Treebank (Miltsakaki et al., 2004). While the annotation approaches differ across these corpora, the requirement of sentence segmentation into

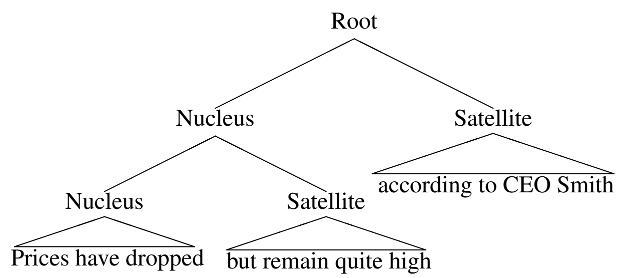

Figure 1: Example Nucleus/Satellite labeled sentence-level discourse tree.

sub-sentential discourse units is shared across all approaches. These resources have facilitated research into stochastic models and algorithms for automatic discourse structure annotation in recent years.

Using the RST Treebank as training and evaluation data, Soricut and Marcu (2003) demonstrated that their automatic sentence-level discourse parsing system could achieve near-human levels of accuracy, if it was provided with manual segmentations and manual parse trees. Manual segmentation was primarily responsible for this performance boost over their fully automatic system, thus making the case that automatic discourse segmentation is the primary impediment to high accuracy automatic sentence-level discourse structure annotation. Their models and algorithm – subsequently packaged together into the publicly available SPADE discourse parser[1] – make use of the output of the Charniak (2000) parser to derive syntactic indicator features for segmentation and discourse parsing.

Sporleder and Lapata (2005) also used the RST Treebank as training data for data-driven discourse parsing algorithms, though their focus, in contrast to Soricut and Marcu (2003), was to avoid context-free parsing and rely exclusively on features in their model that could be derived via finite-state chunkers and taggers. The annotations that they derive are dis-

[1] http://www.isi.edu/publications/licensed-sw/spade/

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 488–495,
Prague, Czech Republic, June 2007.

course "chunks", i.e., sentence-level segmentation and non-hierarchical nucleus/span labeling of segments. They demonstrate that their models achieve comparable results to SPADE without the use of any context-free features. Once again, segmentation is the part of the process where the automatic algorithms most seriously underperform.

In this paper we take up the question posed by the results of Sporleder and Lapata (2005): how much, if any, accuracy reduction should we expect if we choose to use only finite-state derived features, rather than those derived from full context-free parses? If little accuracy is lost, as their results suggest, then it would make sense to avoid relatively expensive context-free parsing, particularly if the amount of text to be processed is large or if there are real-time processing constraints on the system. If, however, the accuracy loss is substantial, one might choose to avoid context-free parsing only in the most time-constrained scenarios.

While Sporleder and Lapata (2005) demonstrated that their finite-state system could perform as well as the SPADE system, which uses context-free parse trees, this does not directly answer the question of the utility of context-free derived features for this task. SPADE makes use of a particular kind of feature from the parse trees, and does not train a general classifier making use of other features beyond the parse-derived indicator features. As we shall show, its performance is not the highest that can be achieved via context-free parser derived features.

In this paper, we train a classifier using a general machine learning approach and a range of finite-state and context-free derived features. We investigate the impact on discourse segmentation performance when one feature set is used versus another, in such a way establishing the utility of features derived from context-free parses. In the course of so doing, we achieve the best reported performance on this task, an absolute F-score improvement of 5.0% over SPADE, which represents a more than 34% relative error rate reduction.

By focusing on segmentation, we provide an approach that is generally applicable to all of the various annotation approaches, given the similarities between the various sentence-level segmentation guidelines. Given that segmentation has been shown to be a primary impediment to high accuracy sentence-level discourse structure annotation, this represents a large step forward in our ability to automatically parse the discourse structure of text, whatever annotation approach we choose.

2 Methods

2.1 Data

For our experiments we use the Rhetorical Structure Theory Discourse Treebank (Carlson et al., 2002), which we will denote RST-DT, a corpus annotated with discourse segmentation and relations according to Rhetorical Structure Theory (Mann and Thompson, 1988). The RST-DT consists of 385 documents from the Wall Street Journal, about 176,000 words, which overlaps with the Penn Wall St. Journal (WSJ) Treebank (Marcus et al., 1993).

The segmentation of sentences in the RST-DT is into clause-like units, known as elementary discourse units, or *edus*. We will use the two terms '*edu*' and 'segment' interchangeably throughout the rest of the paper. Human agreement for this segmentation task is quite high, with agreement between two annotators at an F-score of 98.3 for unlabeled segmentation (Soricut and Marcu, 2003).

The RST-DT corpus annotates *edu* breaks, which typically include sentence boundaries, but sentence boundaries are not explicitly annotated in the corpus. To perform sentence-level processing and evaluation, we aligned the RST-DT documents to the same documents in the Penn WSJ Treebank, and used the sentence boundaries from that corpus.[2] An additional benefit of this alignment is that the Penn WSJ Treebank tokenization is then available for parsing purposes. Simple minimum edit distance alignment effectively allowed for differences in punctuation representation (e.g., double quotes) and tokenization when deriving the optimal alignment.

The RST-DT corpus is partitioned into a training set of 347 documents and a test set of 38 documents. This test set consists of 991 sentences with 2,346 segments. For training purposes, we created a held-out development set by selecting every tenth sentence of the training set. This development set was used for feature development and for selecting the number of iterations used when training models.

2.2 Evaluation

Previous research into RST-DT segmentation and parsing has focused on subsets of the 991 sentence test set during evaluation. Soricut and Marcu (2003)

[2] A small number of document final parentheticals are in the RST-DT and not in the Penn WSJ Treebank, which our alignment approach takes into account.

omitted sentences that were not exactly spanned by a subtree of the treebank, so that they could focus on sentence-level discourse parsing. By our count, this eliminates 40 of the 991 sentences in the test set from consideration. Sporleder and Lapata (2005) went further and established a smaller subset of 608 sentences, which omitted sentences with only one segment, i.e., sentences which themselves are atomic *edus*.

Since the primary focus of this paper is on segmentation, there is no strong reason to omit any sentences from the test set, hence our results will evaluate on all 991 test sentences, with two exceptions. First, in Section 2.3, we compare SPADE results under our configuration with results from Sporleder and Lapata (2005) in order to establish comparability, and this is done on their 608 sentence subset. Second, in Section 3.2, we investigate feeding our segmentation into the SPADE system, in order to evaluate the impact of segmentation improvements on their sentence-level discourse parsing performance. For those trials, the 951 sentence subset from Soricut and Marcu (2003) is used. All other trials use the full 991 sentence test set.

Segmentation evaluation is done with precision, recall and F1-score of segmentation boundaries. Given a word string $w_1 \ldots w_k$, we can index word boundaries from 0 to k, so that each word w_i falls between boundaries $i-1$ and i. For sentence-based segmentation, indices 0 and k, representing the beginning and end of the string, are known to be segment boundaries. Hence Soricut and Marcu (2003) evaluate with respect to sentence internal segmentation boundaries, i.e., with indices j such that $0<j<k$ for a sentence of length k. Let g be the number of sentence-internal segmentation boundaries in the gold standard, t the number of sentence-internal segmentation boundaries in the system output, and m the number of correct sentence-internal segmentation boundaries in the system output. Then

$$P = \frac{m}{t} \quad R = \frac{m}{g} \quad \text{and} \quad F1 = \frac{2PR}{P+R} = \frac{2m}{g+t}$$

In Sporleder and Lapata (2005), they were primarily interested in labeled segmentation, where the segment initial boundary was labeled with the segment type. In such a scenario, the boundary at index 0 is no longer known, hence their evaluation included those boundaries, even when reporting unlabeled results. Thus, in section 2.3, for comparison with reported results in Sporleder and Lapata (2005), our F1-score is defined accordingly, i.e., segmentation boundaries j such that $0 \leq j < k$.

Segmentation system	F1
Sporleder and Lapata best (reported)	88.40
SPADE	
Sporleder and Lapata configuration (reported):	87.06
current configuration:	91.04

Table 1: Segmentation results on the Sporleder and Lapata (2005) data set, with accuracy defined to include sentence initial segmentation boundaries.

In addition, we will report unlabeled bracketing precision, recall and F1-score, as defined in the PARSEVAL metrics (Black et al., 1991) and evaluated via the widely used *evalb* package. We also use *evalb* when reporting labeled and unlabeled discourse parsing results in Section 3.2.

2.3 Baseline SPADE setup

The publicly available SPADE package, which encodes the approach in Soricut and Marcu (2003), is taken as the baseline for this paper. We made several modifications to the script from the default, which account for better baseline performance than is achieved with the default configuration. First, we modified the script to take given parse trees as input, rather than running the Charniak parser itself. This allowed us to make two modifications that improved performance: turning off tokenization in the Charniak parser, and reranking. The default script that comes with SPADE does not turn off tokenization inside of the parser, which leads to degraded performance when the input has already been tokenized in the Penn Treebank style. Secondly, Charniak and Johnson (2005) showed how reranking of the 50-best output of the Charniak (2000) parser gives substantial improvements in parsing accuracy. These two modifications to the Charniak parsing output used by the SPADE system lead to improvements in its performance compared to previously reported results.

Table 1 compares segmentation results of three systems on the Sporleder and Lapata (2005) 608 sentence subset of the evaluation data: (1) their best reported system; (2) the SPADE system results reported in that paper; and (3) the SPADE system results with our current configuration. The evaluation uses the unlabeled F1 measure as defined in that paper, which counts sentence initial boundaries in the scoring, as discussed in the previous section. As can be seen from these results, our improved configuration of SPADE gives us large improvements over the previously reported SPADE performance on this subset. As a result, we feel that we can use SPADE as a very strong baseline for evaluation on the entire test set.

Additionally, we modified the SPADE script to allow us to provide our segmentations to the full discourse parsing that it performs, in order to evaluate the improvements to discourse parsing yielded by any improvements to segmentation.

2.4 Segmentation classifier

For this paper, we trained a binary classifier, which was applied independently at each word w_i in the string $w_1 \ldots w_k$, to decide whether that word is the last in a segment. Note that w_k is the last word in the string, and is hence ignored. We used a log-linear model with no Markov dependency between adjacent tags,[3] and trained the parameters of the model with the perceptron algorithm, with averaging to control for over-training (Collins, 2002).

Let $C{=}\{E, I\}$ be the set of classes: segmentation boundary (E) or non-boundary (I). Let $f(c, i, w_1 \ldots w_k)$ be a function that takes as input a class value c, a word index i and the word string $w_1 \ldots w_k$ and returns a d-dimensional vector of feature values for that word index in that string with that class. For example, one feature might be $(c = E, w_i = \text{the})$, which returns the value 1 when $c = E$ and the current word is 'the', and returns 0 otherwise. Given a d-dimensional parameter vector ϕ, the output of the classifier is that class which maximizes the dot product between the feature and parameter vectors:

$$\hat{c}(i, w_1 \ldots w_k) \;=\; \operatorname*{argmax}_{c \in C} \; \phi \cdot f(c, i, w_1 \ldots w_k) \quad (1)$$

In training, the weights in ϕ are initialized to 0. For m epochs (passes over the training data), for each word in the training data (except sentence final words), the model is updated. Let i be the current word position in string $w_1 \ldots w_k$ and suppose that there have been $j{-}1$ previous updates to the model parameters. Let $\bar{c}_i$ be the true class label, and let $\hat{c}_i$ be shorthand for $\hat{c}(i, w_1 \ldots w_k)$ in equation 1. Then the parameter vector ϕ_j at step j is updated as follows:

$$\phi_j \;=\; \phi_{j-1} - f(\hat{c}, i, w_1 \ldots w_k) + f(\bar{c}, i, w_1 \ldots w_k) \quad (2)$$

As stated in Section 2.1, we reserved every tenth sentence as held-out data. After each pass over the training data, we evaluated the system performance on this held-out data, and chose the model that optimized accuracy on that set. The averaged perceptron was used on held-out and evaluation sets. See Collins (2002) for more details on this approach.

2.5 Features

To tease apart the utility of finite-state derived features and context-free derived features, we consider three feature sets: (1) basic finite-state features; (2) context-free features; and (3) finite-state approximation to context-free features. Note that every feature must include exactly one class label c in order to discriminate between classes in equation 1. Hence when presenting features, it can be assumed that the class label is part of the feature, even if it is not explicitly mentioned.

The three feature sets are not completely disjoint. We include simple position-based features in every system, defined as follows. Because *edus* are typically multi-word strings, it is less likely for a word near the beginning or end of a sentence to be at an *edu* boundary. Thus it is reasonable to expect the position within a sentence of a token to be a helpful feature. We created 101 indicator features, representing percentages from 0 to 100. For a string of length k, at position i, we round i/k to two decimal places and provide a value of 1 for the corresponding quantized position feature and 0 for the other position features.

2.5.1 Basic finite-state features

Our baseline finite-state feature set includes simple tagger derived features, as well as features based on position in the string and n-grams[4]. We annotate tag sequences onto the word sequence via a competitive discriminatively trained tagger (Hollingshead et al., 2005), trained for each of two kinds of tag sequences: part-of-speech (POS) tags and shallow parse tags. The shallow parse tags define non-hierarchical base constituents ("chunks"), as defined for the CoNLL-2000 shared task (Tjong Kim Sang and Buchholz, 2000). These can either be used as tag or chunk sequences. For example, the tree in Figure 2 represents a shallow (non-hierarchical) parse tree, with four base constituents. Each base constituent X begins with a word labeled with B_X, which signifies that this word begins the constituent. All other words within a constituent X are labeled

[3] Because of the sparsity of boundary tags, Markov dependencies between tags buy no additional system accuracy.

[4] We tried using a list of 311 cue phrases from Knott (1996) to define features, but did not derive any system improvement through this list, presumably because our simple n-gram features already capture many such lexical cues.

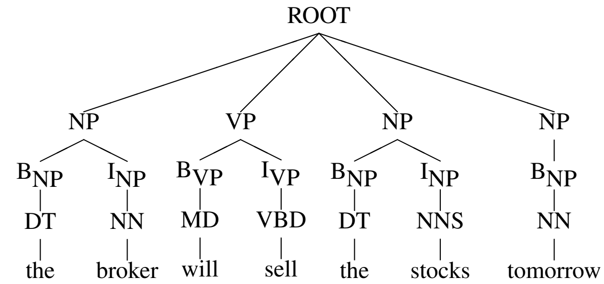

Figure 2: Tree representation of shallow parses, with B(egin) and I(nside) tags

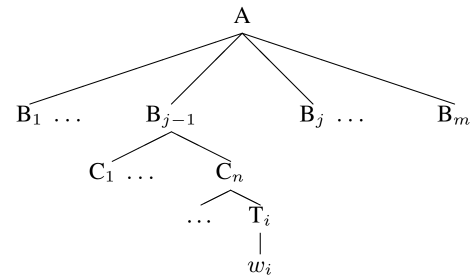

Figure 3: Parse tree schematic for describing context-free segmentation features

I_X, and words outside of any base constituent are labeled O. In such a way, each word is labeled with both a POS-tag and a B/I/O tag.

For our three sequences (lexical, POS-tag and shallow tag), we define n-gram features surrounding the potential discourse boundary. If the current word is w_i, the hypothesized boundary will occur between w_i and w_{i+1}. For this boundary position, the 6-gram including the three words before and the three words after the boundary is included as a feature; additionally, all n-grams for $n < 6$ such that either w_i or w_{i+1} (or both) is in the n-gram are included as features. In other words, all n-grams in a six word window of boundary position i are included as features, except those that include neither w_i nor w_{i+1} in the n-gram. The identical feature templates are used with POS-tag and shallow tag sequences as well, to define tag n-gram features.

This feature set is very close to that used in Sporleder and Lapata (2005), but not identical. Their n-gram feature definitions were different (though similar), and they made use of cue phrases from Knott (1996). In addition, they used a rule-based clauser that we did not. Despite such differences, this feature set is quite close to what is described in that paper.

2.5.2 Context-free features

To describe our context-free features, we first present how SPADE made use of context-free parse trees within their segmentation algorithm, since this forms the basis of our features. The SPADE features are based on productions extracted from full syntactic parses of the given sentence. The primary feature for a discourse boundary after word w_i is based on the lowest constituent in the tree that spans words $w_m \dots w_n$ such that $m \leq i < n$. For example, in the parse tree schematic in Figure 3, the constituent labeled with A is the lowest constituent in the tree whose span crosses the potential discourse boundary after w_i. The primary feature is the production that expands this constituent in the tree, with the proposed segmentation boundary marked, which in this case is: $A \rightarrow B_1 \dots B_{j-1} || B_j \dots B_m$, where $||$ denotes the segmentation boundary. In SPADE, the production is lexicalized by the head words of each constituent, which are determined using standard head-percolation techniques. This feature is used to predict a boundary as follows: if the relative frequency estimate of a boundary given the production feature in the corpus is greater than 0.5, then a boundary is predicted; otherwise not. If the production has not been observed frequently enough, the lexicalization is removed and the relative frequency of a boundary given the unlexicalized production is used for prediction. If the observations of the unlexicalized production are also too sparse, then only the children adjacent to the boundary are maintained in the feature, e.g., $A \rightarrow *B_{j-1}||B_j*$ where $*$ represents zero or more categories. Further smoothing is used when even this is unobserved.

We use these features as the starting point for our context-free feature set: the lexicalized production $A \rightarrow B_1 \dots B_{j-1}||B_j \dots B_m$, as defined above for SPADE, is a feature in our model, as is the unlexicalized version of the production. As with the other features that we have described, this feature is used as an indicator feature in the classifier applied at the word w_i preceding the hypothesized boundary. In addition to these full production features, we use the production with only children adjacent to the boundary, denoted by $A \rightarrow *B_{j-1}||B_j*$. This production is used in four ways: fully lexicalized; unlexicalized; only category B_{j-1} lexicalized; and only category B_j lexicalized. We also use $A \rightarrow *B_{j-2}B_{j-1}||*$ and $A \rightarrow *||B_jB_{j+1}*$ features, both unlexicalized and with the boundary-adjacent category lexicalized. If there is no category B_{j-2} or B_{j+1}, they are replaced with "N/A".

In addition to these features, we fire the same features for all productions on the path from A down

	Segment Boundary accuracy			Bracketing accuracy		
Segmentation system	Recall	Precision	F1	Recall	Precision	F1
SPADE	85.4	85.5	85.5	77.7	77.9	77.8
Classifier: Basic finite-state	81.5	83.3	82.4	73.6	74.5	74.0
Classifier: Full finite-state	84.1	87.9	86.0	78.0	80.0	79.0
Classifier: Context-free	84.7	91.1	87.8	80.3	83.7	82.0
Classifier: All features	89.7	91.3	90.5	84.9	85.8	85.3

Table 2: Segmentation results on all 991 sentences in the RST-DT test set. Segment boundary accuracy is for sentence internal boundaries only, following Soricut and Marcu (2003). Bracketing accuracy is for unlabeled flat bracketing of the same segments. While boundary accuracy correctly depicts segmentation results, the harsher flat bracketing metric better predicts discourse parsing performance.

to the word w_i. For these productions, the segmentation boundary || will occur after all children in the production, e.g., $B_{j-1} \rightarrow C_1 \ldots C_n||$, which is then used in both lexicalized and unlexicalized forms. For the feature with only categories adjacent to the boundary, we again use "N/A" to denote the fact that no category occurs to the right of the boundary: $B_{j-1} \rightarrow *C_n||$N/A. Once again, these are lexicalized as described above.

2.5.3 Finite-state approximation features

An approximation to our context-free features can be made by using the shallow parse tree, as shown in Figure 2, in lieu of the full hierarchical parse tree. For example, if the current word was "sell" in the tree in Figure 2, the primary feature would be ROOT → NP VP||NP NP, and it would have an unlexicalized version and three lexicalized versions: the category immediately prior to the boundary lexicalized; the category immediately after the boundary lexicalized; and both lexicalized. For lexicalization, we choose the final word in the constituent as the lexical head for the constituent. This is a reasonable first approximation, because such typically left-headed categories as PP and VP lose their arguments in the shallow parse.

As a practical matter, we limit the number of categories in the flat production to 8 to the left and 8 to the right of the boundary. In a manner similar to the n-gram features that we defined in Section 2.5.1, we allow all combinations with less than 8 contiguous categories on each side, provided that at least one of the adjacent categories is included in the feature. Each feature has an unlexicalized and three lexicalized versions, as described above.

3 Experiments

We performed a number of experiments to determine the relative utility of features derived from full context-free syntactic parses and those derived solely from shallow finite-state tagging. Our primary concern is with intra-sentential discourse segmentation, but we are also interested in how much the improved segmentation helps discourse parsing.

The syntactic parser we use for all context-free syntactic parses used in either SPADE or our classifier is the Charniak parser with reranking, as described in Charniak and Johnson (2005). The Charniak parser and reranker were trained on the sections of the Penn Treebank not included in the RST-DT test set.

All statistical significance testing is done via the stratified shuffling test (Yeh, 2000).

3.1 Segmentation

Table 2 presents segmentation results for SPADE and four versions of our classifier. The "Basic finite-state" system uses only finite-state sequence features as defined in Section 2.5.1, while the "Full finite-state" also includes the finite-state approximation features from Section 2.5.3. The "Context-free" system uses the SPADE-inspired features detailed in Section 2.5.2, but none of the features from Sections 2.5.1 or 2.5.3. Finally, the "All features" section includes features from all three sections.[5]

Note that the full finite-state system is considerably better than the basic finite-state system, demonstrating the utility of these approximations of the SPADE-like context-free features. The performance of the resulting "Full" finite-state system is not statistically significantly different from that of SPADE (p=0.193), despite no reliance on features derived from context-free parses.

The context-free features, however, even without any of the finite-state sequence features (even lexical n-grams) outperforms the best finite-state system by almost two percent absolute, and the system with all features improves on the best finite-state system by over four percent absolute. The system

[5] In the "All features" condition, the finite-state approximation features defined in Section 2.5.3 only include a maximum of 3 children to the left and right of the boundary, versus a maximum of 8 for the "Full finite-state" system. This was found to be optimal on the development set.

Segmentation	Unlabeled	Nuc/Sat
SPADE	76.9	70.2
Classifier: Full finite state	78.1	71.1
Classifier: All features	83.5	76.1

Table 3: Discourse parsing results on the 951 sentence Soricut and Marcu (2003) evaluation set, using SPADE for parsing, and various methods for segmentation. Scores are unlabeled and labeled (Nucleus/Satellite) bracketing accuracy (F1). The first line shows SPADE performing both segmentation and discourse parsing. The other two lines show SPADE performing discourse parsing with segmentations produced by our classifier using different combinations of features.

with all features is statistically significantly better than both SPADE and the "Full finite-state" classifier system, at $p < 0.001$. This large improvement demonstrates that the context-free features can provide a very large system improvement.

3.2 Discourse parsing

It has been shown that accurate discourse segmentation within a sentence greatly improves the overall parsing accuracy to near human levels (Soricut and Marcu, 2003). Given our improved segmentation results presented in the previous section, improvements would be expected in full sentence-level discourse parsing. To achieve this, we modified the SPADE script to accept our segmentations when building the fully hierarchical discourse tree. The results for three systems are presented in Table 3: SPADE, our "Full finite-state" system, and our system with all features. Results for unlabeled bracketing are presented, along with results for labeled bracketing, where the label is either Nucleus or Satellite, depending upon whether or not the node is more central (Nucleus) to the coherence of the text than its sibling(s) (Satellite). This label set has been shown to be of particular utility for indicating which segments are more important to include in an automatically created summary or compressed sentence (Sporleder and Lapata, 2005; Marcu, 1998; Marcu, 1999; Cristea et al., 2005).

Once again, the finite-state system does not perform statistically significantly different from SPADE on either labeled or unlabeled discourse parsing. Using all features, however, results in greater than 5% absolute accuracy improvement over both of these systems, which is significant, in all cases, at $p < 0.001$.

4 Discussion and future directions

Our results show that context-free parse derived features are critical for achieving the highest level of accuracy in sentence-level discourse segmentation. Given that *edus* are by definition clause-like units, it is not surprising that accurate full syntactic parse trees provide highly relevant information unavailable from finite-state approaches. Adding context-free features to our best finite-state feature model reduces error in segmentation by 32.1%, an increase in absolute F-score of 4.5%. These increases are against a finite-state segmentation model that is powerful enough to be statistically indistinguishable from SPADE.

Our experiments also confirm that increased segmentation accuracy yields significantly better discourse parsing accuracy, as previously shown to be the case when providing reference segmentations to a parser (Soricut and Marcu, 2003). The segmentation reduction in error of 34.5% propagates to a 28.6% reduction in error for unlabeled discourse parse trees, and a 19.8% reduction in error for trees labeled with Nucleus and Satellite.

We have several key directions in which to continue this work. First, given that a general machine learning approach allowed us to improve upon SPADE's segmentation performance, we also believe that it will prove useful for improving full discourse parsing, both at the sentence level and beyond. For efficient inter-sentential discourse parsing, we see the need for an additional level of segmentation at the paragraph level. Whereas most sentences correspond to a well-formed subtree, Sporleder and Lascarides (2004) report that over 20% of the paragraph boundaries in the RST-DT do not correspond to a well-formed subtree in the human annotated discourse parse for that document. Therefore, to perform accurate and efficient parsing of the RST-DT at the paragraph level, the text should be segmented into paragraph-like segments that conform to the human-annotated subtree boundaries, just as sentences are segmented into *edus*.

We also intend to begin work on the other discourse annotated corpora. While most work on textual discourse parsing has made use of the RST-DT corpus, the Discourse GraphBank corpus provides a competing annotation that is not constrained to tree structures (Wolf and Gibson, 2005). Once accurate levels of segmentation and parsing for both corpora are attained, it will be possible to perform extrinsic evaluations to determine their relative utility for different NLP tasks. Recent work has shown promising preliminary results for recognizing and labeling relations of GraphBank structures (Wellner et al., 2006), in the case that the algorithm is provided with manually segmented sentences. Sentence-level segmentation in the GraphBank is very similar to that in the RST-DT, so our segmentation approach should work well for Discourse GraphBank style parsing.

The Penn Discourse Treebank (Miltsakaki et al., 2004), or PDTB, uses a relatively flat annotation of discourse structure, in contrast to the hierarchical structures found in the other two corpora. It contains annotations for discourse connectives and their arguments, where an argument can be as small as a nominalization or as large as several sentences. This approach obviates the need to create a set of discourse relations, but sentence internal segmentation is still a necessary step. Though segmentation in the PDTB tends to larger units than *edus*, our approach to segmentation should be straightforwardly applicable to their segmentation style.

Acknowledgments

Thanks to Caroline Sporleder and Mirella Lapata for their test data and helpful comments. Thanks also to Radu Soricut for helpful input. This research was supported in part by NSF Grant #IIS-0447214. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

References

E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M.P. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of english grammars. In *DARPA Speech and Natural Language Workshop*, pages 306–311.

L. Carlson, D. Marcu, and M.E. Okurowski. 2002. RST discourse treebank. Linguistic Data Consortium, Catalog # LDC2002T07. ISBN LDC2002T07.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of ACL*, pages 173–180.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics*, pages 132–139.

M.J. Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–8.

D. Cristea, O. Postolache, and I. Pistol. 2005. Summarisation through discourse structure. In *6th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing)*.

K. Hollingshead, S. Fisher, and B. Roark. 2005. Comparing and combining finite-state and context-free parsers. In *Proceedings of HLT-EMNLP*, pages 787–794.

A. Knott. 1996. *A Data-Driven Methodology for Motivating a Set of Coherence Relations*. Ph.D. thesis, Department of Artificial Intelligence, University of Edinburgh.

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. 1998. Improving summarization through rhetorical parsing tuning. In *The 6th Workshop on Very Large Corpora*.

D. Marcu. 1999. Discourse trees are good indicators of importance in text. In I. Mani and M. Maybury, editors, *Advances in Automatic Text Summarization*, pages 123–136. MIT Press, Cambridge, MA.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

E. Miltsakaki, R. Prasad, A. Joshi, and B. Webber. 2004. The Penn Discourse TreeBank. In *Proceedings of the Language Resources and Evaluation Conference*.

R. Prasad, A. Joshi, N. Dinesh, A. Lee, E. Miltsakaki, and B. Webber. 2005. The Penn Discourse TreeBank as a resource for natural language generation. In *Proceedings of the Corpus Linguistics Workshop on Using Corpora for Natural Language Generation*.

R. Soricut and D. Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Human Language Technology Conference of the North American Association for Computational Linguistics (HLT-NAACL)*.

C. Sporleder and M. Lapata. 2005. Discourse chunking and its application to sentence compression. In *Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 257–264.

C. Sporleder and A. Lascarides. 2004. Combining hierarchical clustering and machine learning to predict high-level discourse structure. In *Proceedings of the International Conference in Computational Linguistics (COLING)*, pages 43–49.

E.F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL*, pages 127–132.

S. Verberne, L. Boves, N. Oostdijk, and P.A. Coppen. 2006. Discourse-based answering of why-questions. *Traitement Automatique des Langues (TAL)*.

B. Wellner, J. Pustejovsky, C. Havasi, A. Rumshisky, and R. Sauri. 2006. Classification of discourse coherence relations: An exploratory study using multiple knowledge sources. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*.

F. Wolf and E. Gibson. 2005. Representing discourse coherence: A corpus-based analysis. *Computational Linguistics*, 31(2):249–288.

A. Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International COLING*, pages 947–953.