Analysis of Discourse Structure with Syntactic Dependencies and Data-Driven Shift-Reduce Parsing

Kenji Sagae
USC Institute for Creative Technologies
Marina del Rey, CA 90292 USA
sagae@ict.usc.edu

Abstract

We present an efficient approach for discourse parsing within and across sentences, where the unit of processing is an entire document, and not a single sentence. We apply shift-reduce algorithms for dependency and constituent parsing to determine syntactic dependencies for the sentences in a document, and subsequently a Rhetorical Structure Theory (RST) tree for the entire document. Our results show that our linear-time shift-reduce framework achieves high accuracy and a large improvement in efficiency compared to a state-of-the-art approach based on chart parsing with dynamic programming.

1 Introduction

Transition-based dependency parsing using shift-reduce algorithms is now in wide use for dependency parsing, where the goal is to determine the syntactic structure of sentences. State-of-the-art results have been achieved for syntactic analysis in a variety of languages (Bucholz and Marsi, 2006). In contrast to graph-based approaches, which use edge-factoring to allow for global optimization of parameters over entire tree structures using dynamic programming or maximum spanning tree algorithms (McDonald et al., 2005) transition-based models are usually optimized at the level of individual shift-reduce actions, and can be used to drive parsers that produce competitive accuracy using greedy search strategies in linear time.

Recent research in data-driven shift-reduce parsing has shown that the basic algorithms used for determining dependency trees (Nivre, 2004) can be extended to produce constituent structures (Sagae and Lavie, 2005), and more general dependency graphs, where words can be linked to more than one head (Henderson et al., 2008; Sagae and Tsujii, 2008). A remarkably similar parsing approach, which predates the current wave of interest in data-driven shift-reduce parsing sparked by Yamada and Matsumoto (2003) and Nivre and Scholz (2004), was proposed by Marcu (1999) for data-driven *discourse parsing*, where the goal is to determine the rhetorical structure of a document, including relationships that span multiple sentences. The linear-time shift-reduce framework is particularly well suited for discourse parsing, since the length of the input string depends on document length, not sentence length, making cubic run-time chart parsing algorithms often impractical.

Soricut and Marcu (2003) presented an approach to discourse parsing that relied on syntactic information produced by the Charniak (2000) parser, and used a standard bottom-up chart parsing algorithm with dynamic programming to determine discourse structure. Their approach greatly improved on the accuracy of Marcu's shift-reduce approach, showing the value of using syntactic information in discourse analysis, but recovered only discourse relations within sentences.

We present an efficient approach to discourse parsing using syntactic information, inspired by Marcu's application of a shift-reduce algorithm for discourse analysis with Rhetorical Structure Theory (RST), and Soricut and Marcu's use of syntactic structure to help determine discourse structure. Our transition-based discourse parsing framework combines elements from Nivre (2004)'s approach to dependency parsing, and Sagae and Lavie (2005)'s approach to constituent parsing. Our results improve on accuracy over existing approaches for data-driven RST parsing, while also improving on speed over Soricut and Marcu's chart parsing approach, which produces state-of-the-art results for RST discourse relations within sentences.

Proceedings of the 11th International Conference on Parsing Technologies (IWPT), pages 81–84,
Paris, October 2009.

2 Discourse analysis with the RST Discourse Treebank

The discourse parsing approach presented here is based on the formalization of Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) used in the RST Discourse Treebank (Carlson et al., 2003). In this scheme, the discourse structure of a document is represented as a tree, where the leaves are contiguous spans of text, called *elementary discourse units*, or EDUs. Each node in the tree corresponds to a contiguous span of text formed by concatenation of the spans corresponding to the node's children, and represents a rhetorical relation (*attribution*, *enablement*, *elaboration*, *consequence*, etc.) between these text segments. In addition, each node is marked as a *nucleus* or as a *satellite*, depending on whether its text span represents an essential unit of information, or a supporting or background unit of information, respectively. While the notions of *nucleus* and *satellite* are in some ways analogous to *head* and *dependent* in syntactic dependencies, RST allows for multi-nuclear relations, where two nodes marked as *nucleus* can be linked into one node.

Our parsing framework includes three components: (1) syntactic dependency parsing, where standard techniques for sentence-level parsing are applied; (2) discourse segmentation, which uses syntactic and lexical information to segment text into EDUs; and (3) discourse parsing, which produces a discourse structure tree from a string of EDUs, also benefiting from syntactic information. In contrast to the approach of Soricut and Marcu (2003), which also includes syntactic parsing, discourse segmentation and discourse parsing, our approach assumes that the unit of processing for discourse parsing is an entire document, and that discourse relations may exist within sentences as well as across sentences, while Soricut and Marcu's processes one sentence at a time, independently, finding only discourse relations within individual sentences. Parsing entire documents at a time is made possible in our approach through the use of linear-time transition-based parsing. An additional minor difference is that in our approach syntactic information is represented using dependencies, while Soricut and Marcu used constituent trees.

2.1 Syntactic parsing and discourse segmentation

Assuming the document has been segmented into sentences, a task for which there are approaches with very high accuracy (Gillick, 2009), we start by finding the dependency structure for each sentence. This includes part-of-speech (POS) tagging using a CRF tagger trained on the Wall Street Journal portion of the Penn Treebank, and transition-based dependency parsing using the shift-reduce arc-standard algorithm (Nivre, 2004) trained with the averaged perceptron (Collins, 2002). The dependency parser is also trained with the WSJ Penn Treebank, converted to dependencies using the head percolation rules of Yamada and Matsumoto (2003).

Discourse segmentation is performed as a binary classification task on each word, where the decision is whether or not to insert an EDU boundary between the word and the next word. In a sentence of length n, containing the words w_1, w_2 … w_n, we perform one classification per word, in order. For word w_i, the binary choice is whether to insert an EDU boundary between w_i and w_{i+1}. The EDUs are then the words between EDU boundaries (assuming boundaries exist in the beginning and end of each sentence).

The features used for classification are: the current word, its POS tag, its dependency label, and the direction to its head (whether the head appears before or after the word); the previous two words, their POS tags and dependency labels; the next two words, their POS tags and dependency labels; the direction from the previous word to its head; the leftmost dependent to the right of the current word, and its POS tag; the rightmost dependent to the left of the current word, and its POS tag; whether the head of the current word is between the previous EDU boundary and the current word; whether the head of the next word is between the previous EDU boundary and the current word. In addition, we used templates that combine these features (in pairs or triples). Classification was done with the averaged perceptron.

2.2 Transition-based discourse parsing

RST trees can be represented in a similar way as constituent trees in the Penn Treebank, with a few differences: the trees represent entire documents, instead of single sentences; the leaves of the trees are EDUs consisting of one or more contiguous words; and the node labels contain nucleus/satellite status, and possibly the name of a discourse relation. Once the document has been segmented into a sequence of EDUs, we use a transition-based constituent parsing approach (Sagae and Lavie, 2005) to build an RST tree for the document.

Sagae and Lavie's constituent parsing algorithm uses a stack that holds subtrees, and consumes the input string (in our case, a sequence of EDUs) from left to right, using four types of actions: (1) *shift*, which removes the next token from the input string, and pushes a subtree containing exactly that token onto the stack; (2) *reduce-unary-LABEL*, which pops the stack, and push onto it a new subtree where a node with label *LABEL* dominates the subtree that was popped (3) *reduce-left-LABEL*, and (4) *reduce-right-LABEL*, which each pops two items from the stack, and pushes onto it a new subtree with root *LABEL*, which has as right child the subtree previously on top of the stack, and as left child the subtree previously immediately below the top of the stack. The difference between *reduce-left* and *reduce-right* is whether the head of the new subtree comes from the left or right child. The algorithm assumes trees are lexicalized, and in our use of the algorithm for discourse parsing, heads are entire EDUs, and not single words.

Our process for lexicalization of discourse trees, which is required for the parsing algorithm to function properly, is a simple percolation of "head EDUs," performed in the same way as lexical heads can be assigned in Penn Treebank-style trees using a head percolation table (Collins, 1999). To determine head EDUs, we use the nucleus/satellite status of nodes, as follows: for each node, the leftmost child with nucleus status is the head; if no child is a nucleus, the leftmost satellite is the head. Most nodes have exactly two children, one nucleus and one satellite. The parsing algorithm deals only with binary trees. We use the same binarization transform as Sagae and Lavie, converting the trees in the training set to binary trees prior to training the parser, and converting the binary trees produced by the parser at run-time into *n*-ary trees.

As with the dependency parser and discourse segmenter, learning is performed using the averaged perceptron. We use similar features as Sagae and Lavie, with one main difference: since there is usually no single head-word associated with each node, but a EDU that contains a sequence of words, we use the dependency structure of the EDU to determine what lexical features and POS tags should be used as features associated with each RST tree node. In place of the head-word and POS tag of the top four items on the stack, and the next four items in the input, we use subsets of the words and POS tags in the EDUs for each of those items. The subset of words (and POS tags) that represent an EDU contain the first two and last words in the EDU, and each word in the EDU whose head is outside of the EDU. In the vast majority of EDUs, this subset of words with heads outside the EDU (the *EDU head set*) contains a single word. In addition, we extract these features for the top three (not four) items on the stack, and the next three (not four) words in the input. For the top two items on the stack, in addition to subsets of words and POS tags described above, we also take the words and POS tags for the leftmost and rightmost children of each word in the EDU head set. Finally, we use feature templates that combine these and other individual features from Sagae and Lavie, who used a polynomial kernel and had no need for such templates (at the cost of increased time for both training and running).

3 Results

To test our discourse parsing approach, we used the standard training and testing sections of the RST Discourse Treebank and the compacted 18-label set described by Carlson et al. (2003). We used approximately 5% of the standard training set as a development set.

Our part-of-speech tagger and syntactic parser were *not* trained using the standard splits of the Penn Treebank for those tasks, since there are documents in the RST Discourse Treebank test section that are included in the usual training sets for POS taggers and parsers. The POS tagger and syntactic parser were then trained on sections 2 to 21 of the WSJ Penn Treebank, excluding the specific documents used in the test section of the RST Discourse Treebank.

Table 1 shows the precision, recall and f-score of our discourse segmentation approach on the test set, compared to that of Soricut and Marcu (2003) and Marcu (1999). In all cases, results were obtained with automatically produced syntactic structures. We also include the total time required for syntactic parsing (required in our

	Prec.	Recall	F-score	Time
Marcu99	83.3	77.1	80.1	-
S&M03	83.5	82.7	83.1	361s
this work	**87.4**	**86.0**	**86.7**	**40s**

Table 1: Precision, recall, f-score and time for discourse segmenters, tested on the RST Discourse Treebank. Time includes syntactic parsing, Charniak (2000) for S&M03, and our implemetation of Nivre arc-standard for our segmenter.

	F-score	Time
Marcu99	37.2	-
S&M03	49.0	481s
this work	**52.9**	**69s**
human	77.0	-

Table 2: F-score for bracketing of RST discourse trees on the test set of the RST Discourse Treebank, and total time (syntactic parsing, segmentation and discourse parsing) required to parse the test set (S&M03 and our approach were run on the same hardware).

segmentation approach and Soricut and Marcu's) and segmentation. For comparison with previous results, we include only segmentation within sentences (if all discourse boundaries are counted, including sentence boundaries, our f-score is 92.9).

Using our discourse segmentation and transition-based discourse parsing approach, we obtain 42.9 precision and 46.2 recall (44.5 f-score) for all discourse structures in the test set. Table 2 shows f-score of labeled bracketing for discourse relations *within sentences* only, for comparison with previously published results. We note that human performance on this task has f-score 77.0.

While our f-score is still far below that of human performance, we have achieved a large gain in speed of processing compared to a state-of-the-art approach.

4 Conclusion

We have presented an approach to discourse analysis based on transition-based algorithms for dependency and constituent trees. Dependency parsing is used to determine the syntactic structure of text, which is then used in discourse segmentation and parsing. A simple discriminative approach to segmentation results in an overall improvement in discourse parsing f-score, and the use of a linear-time algorithm results in an a large improvement in speed over a state-of-the-art approach.

Acknowledgments

The work described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

References

Buchholz, S. and Marsi, E. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL 2006 Shared Task*.

Carlson, L., Marcu, D., and Okurowski, M. E. 2003. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In J. van Kuppevelt and R. W. Smith, editors, *Current and New Directions in Discourse and Dialogue*. Kluwer Academic Publishers.

Charniak, E. 2000. A maximum-entropy-inspired parser. In *Proc. of NAACL*.

Collins, M. 1999. *Head-driven statistical models for natural language processing*. PhD dissertation, University of Pennsylvania.

Collins, M. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proc. of EMNLP*. Philadelphia, PA.

Gillick, D. 2009. Sentence Boundary Detection and the Problem with the U.S. In *Proc. of the NAACL HLT: Short Papers*. Boulder, Colorado.

Henderson, J., Merlo, P., Musillo, G., Titov, I. 2008. A Latent Variable Model of Synchronous Parsing for Syntactic and Semantic Dependencies. *In Proc. of CoNLL 2008 Shared Task*, Manchester, UK.

Mann, W. C. and Thompson, S. A. 1988. Rhetorical Structure Theory: toward a functional theory of text organization. *Text*, 8(3):243-281.

Marcu, D. 1999. A decision-based approach to rhetorical parsing. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*.

McDonald, R., Pereira, F., Ribarov, K., and Hajic, J. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT/EMNLP*.

Nivre, J. 2004. Incrementality in Deterministic Dependency Parsing. In *Incremental Parsing: Bringing Engineering and Cognition Together* (workshop at ACL-2004). Barcelona, Spain.

Nivre, J. and Scholz, M. 2004. Deterministic Dependency Parsing of English Text. In *Proc. of COLING*.

Sagae, K. and Lavie, A. 2005. A classifier-based parser with linear run-time complexity. In *Proc. of IWPT*.

Sagae, K. and Tsujii, J. 2008. Shift-reduce dependency DAG parsing. In *Proc. of COLING*.

Soricut, R. and Marcu, D. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proc. of NAACL*. Edmonton, Canada.

Yamada, H. and Matsumoto, Y. 2003. Statistical dependency analysis with support vector machines. In *Proc. of IWPT*.