Improving the Recall of a Discourse Parser by Constraint-based Postprocessing

Sucheta Ghosh*, Richard Johansson†, Giuseppe Riccardi*, Sara Tonelli‡

*DISI, University of Trento
Via Sommarive 14, 38123 Trento (TN), Italy
{ghosh, riccardi}@disi.unitn.it

†Språkbanken, Department of Swedish, University of Gothenburg
Box 100, SE-40530 Gothenburg, Sweden
richard.johansson@gu.se

‡FBK-IRST
Via Sommarive 18, 38123 Trento (TN), Italy
satonelli@fbk.eu

Abstract

We describe two constraint-based methods that can be used to improve the recall of a shallow discourse parser based on conditional random field chunking. These method uses a set of natural structural constraints as well as others that follow from the annotation guidelines of the Penn Discourse Treebank. We evaluated the resulting systems on the standard test set of the PDTB and achieved a rebalancing of precision and recall with improved F-measures across the board. This was especially notable when we used evaluation metrics taking partial matches into account; for these measures, we achieved F-measure improvements of several points.

Keywords: Discourse structure, constraint-based methods, evaluation

1. Introduction

Automatic analysis of the discourse structure of a text is a complex task with a wide range of potential applications. The release of the Penn Discourse Treebank (Prasad et al., 2008a) has resulted in a recent flurry of work in discourse parsing. In particular, there is a growing body of literature describing systems that extract arguments of explicit discourse connectives (Wellner and Pustejovsky, 2007; Elwell and Baldridge, 2008; Ghosh et al., 2011b; Ghosh et al., 2011a).

We previously presented a method for automatic argument extraction based on chunking with conditional random fields (Ghosh et al., 2011a). In contrast to previous approaches to argument extaction, our chunking system is very loosely coupled with the syntactic representation: It is completely straightforward to use one or more constituent, dependency, or shallow parsers in any combination since the argument boundaries are not tied to any particular constituent span. Other advantages include the simplicity of implementation by using standard chunking tools. The runtime of the system is also very low, with most of the processing time spent on feature extraction (i.e. running syntactic parsers).

However, while the chunking-based approach has the advantage of flexibility and speed, it is unable to take the global argument structural constraints into account. In particular, the PDTB annotation guidelines specify that exactly one `Arg1` and one `Arg2` must be annotated for every connective, while we often noticed that our system predicted no arguments. This causes our recall values to be low compared to the precision.

In this paper, we show that adding these constraints to the inference step improves the performance of the discourse parser. In particular, we see strong recall improvements. Global inference methods, including constraint-based as well as learning-based methods (often implemented as rerankers), have seen much use in NLP recently. Inference with constraints in particular has been successful in improving tasks such as semantic role labeling (Punyakanok et al., 2008). This approach may be seen as a simple way to introduce long-distance structural relationships while still keeping the machine learning models simple.

2. The Penn Discourse Treebank

The Penn Discourse Treebank (PDTB) is a resource containing one million words from the Wall Street Journal corpus (Marcus et al., 1993) annotated with discourse relations. While the PDTB is annotated on an English corpus, there are also preliminary efforts to annotate PDTB-style discourse treebanks in other languages including Hindi (Prasad et al., 2008b) and Turkish (Zeyrek and Webber, 2008).

Based on the observation that "no discourse connective has yet been identified in any language that has other than two arguments" (Webber et al. (2010), p. 15), connectives in the PTDB are treated as discourse predicates taking two text spans as *arguments*, i.e. parts of the text that describe events, propositions, facts, situations. These two types of arguments in the PDTB are called `Arg1` and `Arg2`, with the numbering not necessarily corresponding to their order in text. Instead, `Arg2` is the argument syntactically bound to the connective, while `Arg1` is the other one. While the `Arg2` is typically very close to the connective, the `Arg1` may be much more distant, and may even occur in other

sentences. Table 1 shows some statistics about how often the Arg1 occurs intersententially.

In the PDTB, discourse relations can be either explicitly or implicitly expressed. However, in this paper we focus exclusively on *explicit* connectives and the identification of their arguments, including the exact spans. This kind of classification is very complex, since Arg1 and Arg2 can occur in many different configurations. In particular, an explicit connective can occur between two arguments (e.g. clauses connected by *because*) or at the beginning of the sentence (for example, when a sentence begins with *since*). It can also appear inside an argument, for instance with *instead* or *however* in sentence-internal position.

Arg1 in same sentence as connective	60.9%
Arg1 in previous, adjacent sentence	30.1%
Arg1 in previous, non adjacent sentence	9.0%

Table 1: Statistics about the position of the Arg1 with respect to the explicit discourse connective. Taken from Prasad et al. (2008a).

3. Implementation

Our system for the automatic extraction of discourse arguments for explicit connectives (Ghosh et al., 2011a) consists of a pipeline, illustrated in Figure 1. Firstly, we assume that the explicit discourse connectives (and their high-level senses) are given to the system as input. They can be taken from the gold standard or automatically identified and disambiguated (Pitler and Nenkova, 2009), and for simplicity we used gold-standard connectives in this work. We then apply a module to extract the Arg2 arguments, which are the easiest to identify since they are syntactically connected to the discourse connectives. After the Arg2s have been identified, we finally apply the Arg1 extractor.

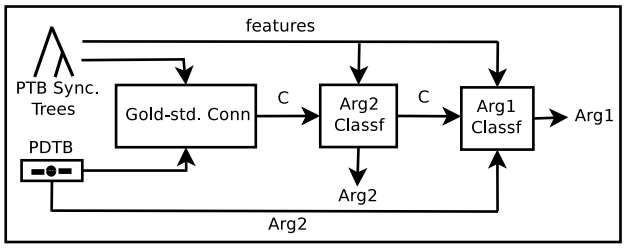

Figure 1: Pipeline for argument detection given a connective.

The Arg2 and Arg1 extractors are implemented as conditional random field sequence labelers, which use a set of syntactic and structural features (see Ghosh et al. (2011a) for a full discussion). In order to reduce the processing time, we apply the sequence labelers to the sentence containing the connective, and a context window of up to two sentences before and after.

3.1. Adding Constraints

In our evaluations (Ghosh et al., 2011a), recall was always lower than precision. We noticed that the system often failed to predict any argument at all. This was especially true for Arg1s, which are not always syntactically connected to the connective and thus typically more distant than the Arg2s. However, since the PDTB annotation guidelines specify that exactly one Arg1 and one Arg2 must be annotated for every connective, we may force the system to output arguments of each type. To improve the recall, we therefore implemented a weighted constraint-based postprocessor to make the system produce output satisfying the requirements defined by the annotation guidelines.

In order to find the best solution with a minimum of constraint violations, we generated the top k analyses output by the CRF for every sentence; these analyses can then be combined to form the k top analyses for the whole 5-sentence window around the connective. This combination is most efficiently carried out using a priority queue similar to a chart cell in the k-best parsing algorithm by Huang and Chiang (2005).

The algorithm then proceeds through the k-best list and outputs an argument segmentation with the minimal number of constraint violations. If there are more than one such segmentation, we select the one with the highest probability. We note that the search for the optimum could as well have been implemented directly in the CRF inference as a modified Viterbi procedure, with a slightly more complex dynamic programming table. We leave the implementation of this algorithm to future work.

We counted the following five conditions as constraint violations:

Overgeneration. This constraint is violated if an Arg1 or Arg2 is split over multiple sentences. However, due to the fact that an argument may be split into several pieces (because of attribution spans, nonprojective syntactic constructions, or embedded connectives), we allow an argument to be split into more than one part in the same sentence.

Undergeneration. Since every connective must have arguments of each type, this constraint is violated if an argument is missing.

Intersentential Arg2. We count every Arg2 outside the sentence containing the connective as a violation, since they are required to be syntactically connected to the connective.

Arg1 after the connective sentence. We count every Arg1 after the sentence containing the connective as a violation.

Argument overlapping with the connective. Arguments are not allowed to overlap with the connective, since PDTB uses discontinuous argument spans to encode situations where a connective is embedded in an argument span.

3.2. Soft Constraints

In addition, we investigated an implementation based on *soft* constraints. For a hypothesis h with a set of violated constraints $V(h)$, we define a scoring function $f(h)$ based

on the score assigned by the base CRF and a set of *constraint weights*, with one weight w_C for every violated constraint C. Our system then selects the hypothesis h that maximizes $f(h)$.

$$f(h) = \log P_{\text{CRF}}(h) - \sum_{C \in V(h)} w_C$$

Based on tuning on a development set, we set all the constraint weights to 1, except the weight for *Undergeneration* which was set to 2.

4. Analysis

We first report the argument extraction performance for the constraint-based postprocessors and compare it to the baseline CRF, and then analyze various aspects of the performance.

4.1. Performance Measurements

Table 2 shows the performance of the baseline system (Ghosh et al., 2011a). As in that paper, we show precision and recall values using three evaluation protocols: *exact*, where an argument must have exactly the same boundaries to be counted as correct; *overlap*, where an argument is counted as correct if it overlaps with a gold standard argument; and *partial*, where a weight between 0 and 1 is used to measure the extent to which a segment corresponds to the gold standard (Johansson and Moschitti, 2010). As previously noted, the recall values are fairly low compared to the precision values.

		P	R	F1
Arg2	Exact	83.4	75.1	79.1
	Partial	93.4	84.2	88.6
	Overlap	97.2	87.5	92.1
Arg1	Exact	69.9	48.5	57.3
	Partial	82.9	61.7	70.7
	Overlap	91.0	63.1	74.6

Table 2: Performance of the baseline discourse parser.

Table 3 shows the effect of the postprocessing with hard constraints, using a k of 8. We note that recall is improved in all settings, in particular for `Arg1`. The increased recall is offset by lower values of precision. However, F-measure always improves, especially for the partial and overlap measures.

		P	R	F1
Arg2	Exact	80.8	77.9	79.3
	Partial	92.8	89.0	90.9
	Overlap	96.9	93.4	95.1
Arg1	Exact	58.9	57.8	58.4
	Partial	73.6	75.7	74.6
	Overlap	80.5	79.0	79.7

Table 3: Results with constraint-based postprocessing.

Table 4 shows the corresponding table for the postprocessor using soft constraints, again with a k of 8. This postprocessor strikes a middle ground between the precision-oriented baseline system and the postprocessor with hard constraints, which is very recall-oriented. We also note that this system scores achieves the highest exact F-measure, while the other postprocessor has higher values for partial and overlap F-measures.

		P	R	F1
Arg2	Exact	81.8	77.1	79.4
	Partial	93.0	87.6	90.2
	Overlap	97.1	91.5	94.2
Arg1	Exact	66.8	53.1	59.2
	Partial	80.6	68.0	73.7
	Overlap	88.3	70.1	78.1

Table 4: Results with postprocessing using soft constraints.

4.2. Intersentential Arguments

The most challenging arguments to extract are the *intersentential* `Arg1`. Table 5 shows the performance of the three systems on these arguments. For these arguments, the postprocessor with hard constraints stands out from the other two: it is much more recall-oriented, while the other two have fairly similar performances. However, the constraint-based systems always outperform the baseline for all types of F-measure.

		P	R	F1
Baseline	Exact	52.9	27.5	36.2
	Partial	68.6	40.2	50.7
	Overlap	78.8	41.0	53.9
Postprocessing (hard)	Exact	39.1	37.8	38.5
	Partial	55.9	56.4	56.1
	Overlap	62.4	60.3	61.4
Postprocessing (soft)	Exact	49.2	29.8	37.1
	Partial	65.9	44.1	52.7
	Overlap	75.0	45.5	56.6

Table 5: Intersentential Arg1 extraction results.

Because of our window-based pruning strategy, the constraints naturally lead to a certain amount of overgeneration: in about 6% of the cases, the gold-standard `Arg1` is located outside the 5-sentence window, while the constraints still force the system to predict an `Arg1` inside the window. This lowers the upper bound on the precision that our system can possible achieve.

4.3. The Effect of the Number of Hypotheses

In any method based on generation of multiple hypotheses from an underlying base system, it is important to investigate the question of how many hypotheses are needed to reach the best achievable performance, since generating a large set of hypotheses may be inefficient. Table 6 shows the effect of the k value on the overlap F-measure for the task of `Arg1` extraction, along with the oracle F-measure for the same task.

k	1	2	4	8	16
F1	74.6	79.1	79.4	79.7	79.7
Oracle F1	74.6	84.5	88.8	92.6	94.8

Table 6: Arg1 overlap F-measure for different values of k.

As is typical for these approaches, the largest gain is achieved immediately, when going from one to two hypotheses. However, in contrast to approaches based on reranking (see e.g. (Johansson and Moschitti, 2010)), our performance reaches a plateau very quickly when increasing the hypothesis set size. This can be explained by the fact that our method immediately returns when finding a hypothesis without constraint violations. Table 7 shows the distribution of the positions of the first violation-free hypothesis. We note that a violation-free hypothesis was available among the four top-scored hypothesis in 97% of the cases.

1	2	3	4	5	6	7	8	>8
1,088	370	55	35	15	10	5	3	10

Table 7: Distribution of the position in the k-best list of the frst hypothesis without constraint violations.

5. Conclusion

We have presented a constraint-based method that improves a shallow discourse parser based on chunking with conditional random fields. The method converts a severely undergenerating output into one where precision and recall are balanced, and where the requirements imposed by the annotations guidelines are fulfilled. The recall improvements are particularly visible when we use evaluation protocols with reduced strictness in boundary checking.

The method we have presented here is simple to implement, but it would also be interesting to see how well it compares to other global approaches. In particular, it would be very straightforward to replace our weighted constraint system by a reranker trained using standard machine learning techniques. Even in that case, the constraint system could serve as a filter to reduce the hypothesis set size for the reranker. However, the development of useful features for a reranker is an open problem.

6. Acknowledgements

This work was partially funded by the LiveMemories project (`www.livememories.org`). Richard Johansson was funded by the EC FP7 under grant 231126: LivingKnowledge – Facts, Opinions and Bias in Time, and by the Centre for Language Technology at Gothenburg University.

7. References

Robert Elwell and Jason Baldridge. 2008. Discourse connective argument identification with connective specific rankers. In *In Proceedings of the 2nd IEEE International Conference on Semantic Computing (ICSC 2008)*, pages 198–205, Santa Clara, United States.

Sucheta Ghosh, Richard Johansson, Giuseppe Riccardi, and Sara Tonelli. 2011a. Shallow discourse parsing with conditional random fields. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 1071–1079, Chiang Mai, Thailand.

Sucheta Ghosh, Sara Tonelli, Giuseppe Riccardi, and Richard Johansson. 2011b. End-to-end discourse parser evaluation. In *Proceedings of the Fifth IEEE International Conference on Semantic Computing (ICSC 2011)*, Palo Alto, United States.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT 2005)*, pages 53–64, Vancouver, Canada.

Richard Johansson and Alessandro Moschitti. 2010. Syntactic and semantic structure for opinion expression detection. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 67–76.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing*.

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

Rashmi Prasad, Samar Husain, Dipti Sharma, and Aravind Joshi. 2008b. Towards an annotated corpus of discourse relations in Hindi. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, Hyderabad, India.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Bonnie Webber, Markus Egg, and Valia Kordoni. 2010. Discourse Structure and Language Technology. *Natural Language Engineering*, 1(1):1–49.

Ben Wellner and James Pustejovsky. 2007. Automatically identifying the arguments of discourse connectives. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 92–101, Prague, Czech Republic.

Deniz Zeyrek and Bonnie Webber. 2008. A discourse resource for Turkish: Annotating discourse connectives in the METU corpus. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, pages 65–71, Hyderabad, India.