Improving a Pipeline Architecture for Shallow Discourse Parsing

Yangqiu Song Haoruo Peng Parisa Kordjamshidi Mark Sammons Dan Roth

Cognitive Computation Group, University of Illinois at Urbana-Champaign
{yqsong,hpeng7,kordjam,mssammon,danr}@illinois.edu

Abstract

We present a system that implements an end-to-end discourse parser. The system uses a pipeline architecture with seven stages: preprocessing, recognizing explicit connectives, identifying argument positions, identifying and labeling arguments, classifying explicit and implicit connectives, and identifying attribution structures. The discourse structure of a document is inferred based on these components. For NLP analysis, we use Illinois NLP software[1] and the Stanford Parser. We use lexical and semantic features based on function words, sentiment lexicons, brown clusters, and polarity features. Our system achieves an F1 score of 0.2492 in overall performance on the development set and 0.1798 on the blind test set.

1 Introduction

The Illinois discourse parsing system builds on existing approaches, using a series of classifiers to identify different elements of discourse structures such as argument boundaries and types along with discourse connectives and senses. In developing the components of this pipeline, we investigated different kinds of features to try to improve abstraction while retaining sufficient expressivity. To that end, we investigated a combination of parse and lexical (function word) features; brown clusters; and relations between verb-argument structures in consecutive sentences.

2 System Description

In this section, we describe the system we developed, and introduce the features we used in each component. For our starting point, we implemented a pipeline architecture based on the description in Lin et al. (2014), then investigated features and inference approaches to improve the system. The pipeline includes seven components. After a preprocessing step, the system identifies explicit connectives, determines the positions of the arguments relative to the connective, identifies and labels arguments, classifies explicit and implicit connectives, and identifies attribution structures. The system architecture is presented in Figure 1. All the classifiers are built based on LibLinear (Fan et al., 2008) via the interface of Learning Based Java (LBJava) (Rizzolo and Roth, 2010).

2.1 Preprocessing

The preprocessing stage identifies tokens, sentences, part-of-speech (Roth and Zelenko, 1998), shallow parse chunks (Punyakanok and Roth, 2001), lemmas[2], syntactic constituency parse (Klein and Manning, 2003), and dependency parse (de Marneffe et al., 2006). This stage also generates a mapping from our own token indexes to those provided in the gold standard data, to allow evaluation with the official shared task scoring code.

2.2 Recognizing Explicit Connectives

To recognize explicit connectives, we construct a list of existing connectives labeled in the Penn Discourse Treebank (Prasad et al., 2008a). Since not all the words of the connective list are necessarily true connectives when they appear in text, we build a binary classifier to determine when a word matching an entry in the list represents an actual connective. We only focus on the connectives with consecutive tokens and ignore the non-consecutive connectives. We generate lexico-syntactic and path features associated with the

[1] http://cogcomp.cs.illinois.edu/page/software

[2] http://cogcomp.cs.illinois.edu/page/software_view/illinois-lemmatizer

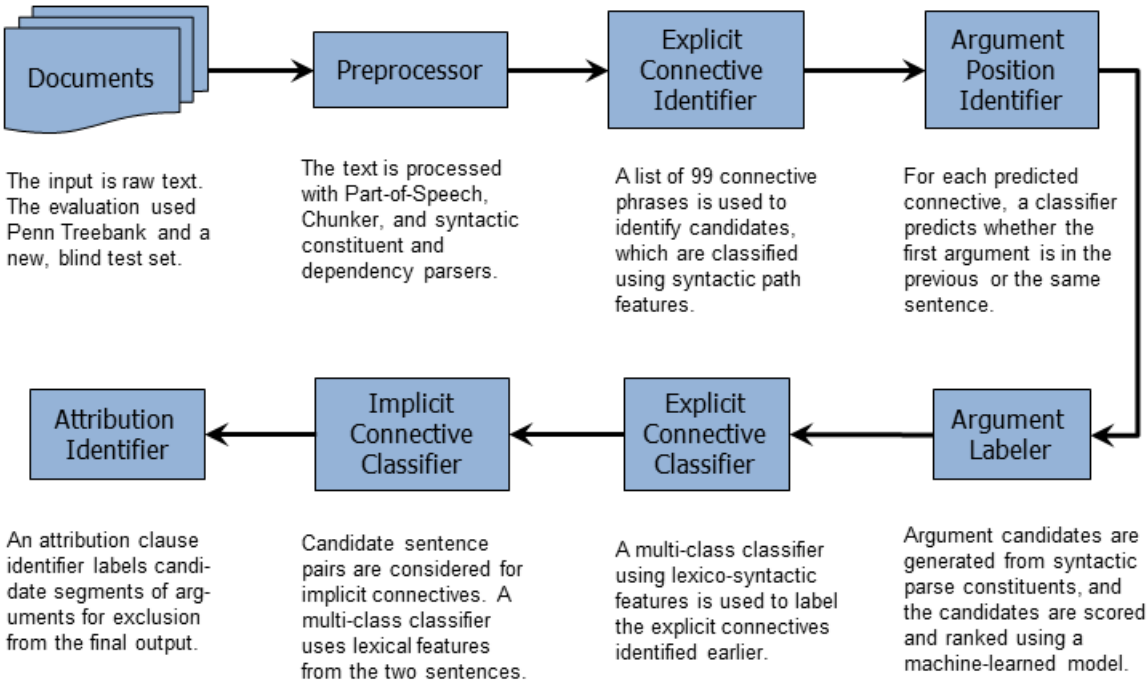

Figure 1: System architecture

connectives following Lin et al. (2014).

2.3 Identifying Arguments and Argument Positions

For each explicit connective, we first identify the relevant argument positions – whether Arg1 is in the previous sentence relative to the connective or in the same sentence. We build a classifier incorporating the contextual lexico-syntactic features. We then detect the spans of Arg1 and Arg2 based on these two decisions. The features used are similiar to those of Lin et al. (2014).

To generate candidate arguments, we enumerate all subtrees in the parse tree of each sentence. We then classify the subtrees to be Arg1, Arg2, or None. In addition to the features used by Lin et al. (2014), we also add function words to the path features. If we detect a word which is in the lexicon of function words, we replace the corresponding tag used in the path with the word's surface string.

2.4 Classifying Explicit Connectives

To classify explicit connectives, we build a multi-class classifier to identify the senses. In addition to the features used by Lin et al. (2014), we also incorporate Brown cluster (Brown et al., 1992) features generated by Liang (2005). The Brown clustering algorithm produces a binary tree, where each word can be uniquely identified by its path from the root, and this path can be compactly represented with a bit string. Different lengths of the prefix of this root-to-leaf path provide different levels of word abstraction. In this implementation, we set the length of the prefix to 6.

2.5 Classifying Implicit Connectives

We classify the sense of implicit connectives based on four sets of features. We follow Lin et al. (2014) to generate the product rules of both constituent parse and dependency parse features, and to generate the word pair features to enumerate all the word pairs in each pair of sentences. In addition, similar to Rutherford and Xue (2014), we incorporate Brown cluster pairs by replacing each word with its Brown cluster prefixes (length of 4) in the way described in Section 2.4. We also use another rich source of information - the polarity of context, which has been previously shown to be useful for coreference problems (Peng et al., 2015). We extract polarity information using the data provided by Wilson et al. (2005) given the predicates of two discourse arguments. The data contains 8221 words, each of which is labeled with a polarity of *positive*, *negative*, or *neutral*. We use

the extracted polarity values to construct three features: Two individual polarities of the two predicates and the conjuction of them. We also implement feature selection to remove the features that are active in the corpus less than five times. As a result, we have 16,989 parse tree features, 4,335 dependency tree features, 77,677 word pair features, and 67,204 Brown cluster features.

2.6 Identifying Attribution Structures

We train two classifiers for attribution identification based on the original PDTB data (sections 2-21). The first classifier is similar to that developed by Lin et al. (2014). We use the patterns proposed by Skadhauge and Hardt (2005) to enumerate all the candidate attribution spans. The coverage of our implementation is 59.8%, which means that we can only enumerate the candidates that contain the attribution covering 59.8% of the annotation. We then build a classifier following Lin et al. (2014) to decide whether the candidate is a valid attribution or not. Using the sentences that contain the attribution(s), we also train a tagger. The tagger is designed following the features used in Illinois Chunker (Punyakanok and Roth, 2001).

3 Evaluation and Results

In this section, we present the data we used and results of the evaluation based on both cross-validation on the training data (computed within our software) and using the official CoNLL 2015 Shared Task evaluation framework of Xue et al. (2015).

3.1 Data

The data we used is provided through the CoNLL-2015 shared task (Xue et al., 2015), which is a modification of Penn Discourse Treebank (PDTB) (Prasad et al., 2008b) sections 2 through 21. The training data for attribution identification is obtained from the original PDTB release, also sections 2 through 21.

3.2 Cross-Validation Results

We first present the cross validation results for each component using the training data (Table 1). All the results are averaged over 10-fold cross validation of all the examples we generated, using our own predicted features and our own evaluation code. Each component is evaluated in isolation, assuming the inputs are from gold data (for example: for connective classification, it is assumed we have the correct connective and arguments provided as input). This means that these results are higher than they would be if evaluated using inputs generated by previous stages of the system, although in this evaluation they still use the predicted part-of-speech, chunk, and parse information.

Table 1: Cross validation results of all the components.

	P	R	F1
Explicit Connectives	92.97	93.91	93.44
Argument Positions	98.15	98.15	98.15
Exact Arg1	64.41	64.95	64.68
Exact Arg2	87.06	86.06	86.56
Partial Arg1	77.02	77.66	77.34
Partial Arg2	94.74	93.65	94.19
Explicit Sense	83.18	83.18	83.18
Implicit Sense	34.58	34.58	34.58
Attribution Identification	82.94	58.02	68.27
Attribution Tagger	59.75	56.49	58.08

3.3 CoNLL Results

We also present the results evaluated by the CoNLL-2015 shared task scorer on dev, test, and blind sets in Tables 2, 3, and 4. The results are consistent with the cross validation results, allowing for the fact that components are working with predicted inputs rather than gold annotations. The sense performance reported here is the macro-average of explicit and implicit senses.

Compared with the best results on the blind set, which is shown in Table 5, our main weaknesses lie in the sense classifier and argument detector. Since we presently ignore the disjoint connectives, our results can be improved if we incorporate those missing connectives. We do not presently incorporate the argument information in connective detection and sense classification for the explicit parser. Connective detection and classification can be improved if we also incorporate more features from arguments or perform joint learning.

4 Discussion

In this section we point to some types of errors in our system's predictions and the complications of working on the provided corpus.

Table 2: CoNLL results on dev set.

	P	R	F1
Explicit connective	93.27	89.71	91.45
Exact Arg1	50.88	56.62	53.59
Exact Arg2	62.27	69.29	65.59
Exact Arg1 & Arg2	41.24	45.89	43.44
Sense	33.88	17.87	21.27
Parser	23.65	26.32	24.92

Table 3: CoNLL results on test set.

	P	R	F1
Explicit connective	92.33	91.33	91.83
Exact Arg1	45.93	52.71	49.09
Exact Arg2	58.97	67.66	63.02
Exact Arg1 & Arg2	35.73	41.00	38.18
Sense	27.01	17.54	15.72
Parser	18.97	21.76	20.27

Table 4: CoNLL results on blind set.

	P	R	F1
Explicit connective	89.11	86.87	87.98
Exact Arg1	49.52	51.61	50.55
Exact Arg2	66.83	69.64	68.21
Exact Arg1 & Arg2	40.48	42.18	41.31
Sense	21.02	16.81	16.49
Parser	17.62	18.36	17.98

Table 5: CoNLL best results on blind set.

	P	R	F1
Explicit connective	93.48	90.29	91.86
Exact Arg1	55.12	56.58	55.84
Exact Arg2	73.49	75.43	74.45
Exact Arg1 & Arg2	45.77	46.98	46.37
Sense	23.29	20.56	20.27
Parser	23.69	24.32	24.00

4.1 Error Analysis

We provide examples of the errors in the systems's predictions that show where the system can be improved, but also some that may indicate possible improvements in the task annotations themselves.

The argument boundaries in our system are predicted based on parse tree constituents. Some mistakes occur when an argument is non-contiguous, and some extraneous content is included. However, the UI-CCG system also generated correct arguments with erroneous token offsets due to tokenization differences.

Predicted Argument:
Golden West Financial Corp. , riding above the turbulence that has troubled most of the thrift industry , posted a 16 % increase of third-quarter earnings to $41.8 millon , or 66 cents a share .
Gold Argument:
Golden West Financial Corp posted a 16% increase of third-quarter earnings to $41.8 millon, or 66 cents a share

Although the shared task specification indicates that attribution detection is not evaluated, our results suggest that it is helpful for some cases in order to obtain the correct argument boundaries.

Predicted Argument:
In savings activity , Mr. Sandler said consumer deposits have enjoyed a steady increase throughout 1989 , and topped $11 billion at quarter 's end for the first time in the company 's history .
Gold Argument:
consumer deposits have enjoyed a steady increase throughout 1989, and topped $11 billion at quarter's end for the first time in the company's history

In some cases, there seems to be a legitimate alternative explanation. The issue of whether connectives should be included in arguments seems somewhat hard to pin down.

Prediction:
argument1 then giving up some of its gains
connective/sense as (Contingency.Cause.Reason)
argument2 the dollar recovered
Gold:
argument1 **and** then giving up some of its gains
connective/sense as (Temporal.Synchrony)
argument2 the dollar recovered

We also found examples where it is hard to make sense of the gold token offsets.

Prediction – argument 1:
text: The more active December delivery gold settled with a gain of $3.90 an ounce at $371.20
tokens: 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282
Gold – argument 1:
text: The more active December delivery gold settled with a gain of $3.90 an ounce at $371.20
tokens: 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284

4.2 Task-specific Complications

Since we wanted to use our own NLP software and to build a general-purpose system that will process raw text from new sources, we parsed the raw text files provided for the PDTB data. However, the shared task formulation requires outputs to be specified in terms of token indexes. We must therefore align our tokenization to that of the task-provided parse files, which were based on the gold tokenization of the Penn Treebank. We found

some disagreements in tokenization decisions by our NLP tools with the gold standard which introduced unwelcome complications and consequent errors. Possibly, character spans from the original text might provide an accurate but less constraining basis for evaluating system outputs, although the gold standard tokenization appears to omit terminal periods from abbreviated words.

5 Extending the Submitted System

Comparing the experimental results of other participating systems in the shared task to our model, it seems that sense disambiguation is the critical step in our pipeline that requires further improvements. We designed an initial model for making global decisions for the sequence of senses that can occur in a paragraph. The idea is to consider the label of the neighboring senses when predicting the sense of any implicit or explicit discourse relation candidate. To implement this idea we designed a constrained conditional model (CCM) (Chang et al., 2012) in LBJava. In this model two basic classifiers are trained. A first classifier, $C1$, is trained to predict the sense of each candidate explicit/implicit relation and a second classifier $C2$ is trained to predict the cooccurrence of the senses of any neighboring pair of candidate explicit/implicit relations. These classifiers are trained independently using features similar to those in the previous pipeline model. At prediction time, using $C1$ and $C2$ we make a global decision for the whole paragraph by modeling the component decisions as a sequence tagging task formulated as a CCM. In this model, the sequential joint prediction is made by adding two global constraints on the predictions made by $C1$ and $C2$: a) $C2$ is applied on all pairs of neighboring relation candidates contained in a paragraph and the label assignments to any pair should be consistent with the label assignments made by $C1$ to the relations in that pair. b) For any two neighboring pairs in a paragraph that share a relation, the label assignments should be consistent; that is, if a pair $p1$ contains relation i and relation $i+1$ and the next pair $p2$ contains relation $i+1$ and relation $i+2$ then the assignments to the shared relation, $i+1$, should be the same. This constraint should hold for the whole paragraph. Using this model, we consider both emission and transition factors in making a global decision for the whole sequence of senses in a paragraph. However, this initial experiment on joint inference did not yield a significant improvement when tested on the sense prediction layer given all ground-truth labels of the previous layers in the pipeline.

6 Conclusions and Future Work

We have built a reasonably effective discourse parser using a pipeline architecture, and identified some features that improve performance over the previous reported state-of-the-art, including features based on Brown Clusters and on argument polarity information. We have also begun investigating the use of constrained conditional models for global inference.

Two natural extensions are: a) Improved features for sense classification. Our sense classification accuracy is relatively low. We need to improve the features we extract from the candidate arguments, and ideally these will reflect a higher level of semantic abstraction than the brown cluster features we used here. b) Global inference over multiple component decisions using Constrained Conditional Models.

Acknowledgments

This work builds on a core implementation developed by Yee Seng Chan. We thank the reviewers for their helpful advice, and the Shared Task organizers for their hard work and support. This work is supported by DARPA under agreement number FA8750-13-2-0008; by grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k. nih.gov); and by the Multimodal Information Access & Synthesis Center at UIUC, part of CCICADA, a DHS Science and Technology Center of Excellence. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the organizations that supported the work.

References

P. Brown, V. Della Pietra, P. deSouza, J. Lai, and R. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

M. Chang, L. Ratinov, and D. Roth. 2012. Structured learning with constrained conditional models. *Machine Learning*, 88(3):399–431, 6.

M. de Marneffe, B. MacCartney, and C. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

D. Klein and C. D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, volume 15. MIT Press.

P. Liang. 2005. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

Z. Lin, H. T. Ng, and M.-Y. Kan. 2014. A PDTB-styled end-to-end discourse parser. *Natural Language Engineering*, 20(2):151–184.

H. Peng, D. Khashabi, and D. Roth. 2015. Solving hard coreference problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics*, Denver, Colorado, June. Association for Computational Linguistics.

R. Prasad, N. Dinesh, A. Lee, E. Miltsakaki, L. Robaldo, A. Joshi, and B. Webber. 2008a. The penn discourse treebank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*.

R. Prasad, N. Dinesh, A. Lee, E. Miltsakaki, L. Robaldo, A. Joshi, and B. Webber. 2008b. The penn discourse treebank 2.0. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, and Daniel Tapias, editors, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

V. Punyakanok and D. Roth. 2001. The use of classifiers in sequential inference. In *Proc. of the Conference on Neural Information Processing Systems (NIPS)*, pages 995–1001. MIT Press.

N. Rizzolo and D. Roth. 2010. Learning based java for rapid development of nlp systems. In *Proc. of the International Conference on Language Resources and Evaluation (LREC)*, Valletta, Malta, 5.

D. Roth and D. Zelenko. 1998. Part of speech tagging using a network of linear separators. In *Coling-Acl, The 17th International Conference on Computational Linguistics*, pages 1136–1142.

A. Rutherford and N. Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 645–654. Association for Computational Linguistics.

P. R. Skadhauge and D. Hardt. 2005. Syntactic identification of attribution in the rst treebank. In *Proceedings of the Sixth International Workshop on Linguistically Interpreted Corpora (LINC-2005)*.

T. Wilson, P. Hoffmann, S. Somasundaran, J. Kessler, J. Wiebe, Y. Choi, C. Cardie, E. Riloff, and S. Patwardhan. 2005. Opinionfinder: A system for subjectivity analysis. In *Proceedings of HLT/EMNLP on interactive demonstrations*, pages 34–35. Association for Computational Linguistics.

N. Xue, H. T. Ng, S. Pradhan, R. Prasad, C. Bryant, and A. Rutherford. 2015. The conll-2015 shared task on shallow discourse parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning: Shared Task*, Beijing, China.