A Joint Learning Approach to Explicit Discourse Parsing via Structured Perceptron

Sheng Li, Fang Kong* and Guodong Zhou

School of Computer Sciences and Technology, Soochow University, Suzhou, Jiangsu, 215006, China
qcl6355@gmail.com, kongfang@suda.edu.cn, gdzhou@suda.edu.cn

Abstract. Discourse parsing is a challenging task and plays a critical role in discourse analysis. In this paper, we focus on building an end-to-end PDTB-style explicit discourse parser via structured perceptron by decomposing it into two components, i.e., a connective labeler, which identifies connectives from a text and determines their senses in classifying discourse relationship, and an argument labeler, which identifies corresponding arguments for a given connective. Particularly, to reduce error propagation and incorporate the interaction between the two components, a joint learning approach via structured perceptron is proposed. Evaluation on the PDTB corpus shows that our two-components explicit discourse parser can achieve comparable performance with the state-of-the-art one. It also shows that our joint learning approach can significantly outperform the pipeline ones.

1 Introduction

[1] Discourse parsing determines the internal structure of a text, e.g., the discourse relationship between its text units. Recently it has become the research focus due to its critical importance on the downstream natural language processing (NLP) applications, such as coherence modeling [1, 11], text summarization [9], statistical machine translation [14] and information extraction [16].

As the largest discourse corpus, the Penn Discourse TreeBank (PDTB) corpus [19] adds a layer of discourse annotations on the top of the Penn TreeBank (PTB) corpus [13] and has attracted increasing attention in current discourse related work [5, 18, 10, 20, 7, 6, 12]. However, although much research work has been carried out on certain components since the release of the PDTB corpus, there is little work on constructing an end-to-end discourse parser.

In this paper, we focus on building an end-to-end PDTB-style discourse parser for explicit connectives. Particularly, we decompose the explicit discourse parser into two component, i.e., a connective labeler, which identifies connectives from a text and determines their senses in classifying discourse relationship, and an argument labeler, which identifies corresponding arguments for a given connective. Besides, a joint learning approach via structured perceptron is

[1] * Corresponding Author

introduced to reduce error propagation and incroporate the interaction between the two components. Evaluation on the PDTB corpus shows the appropriateness of our framework and the effectiveness of our joint learning approach.

The rest of this paper is organized as follows. After introducing the PDTB corpus in Section 2, we briefly review the related work on explicit discourse parser on the PDTB corpus in Section 3. Section 4 describes our end-to-end explicit discourse parser in detail. Then our joint learning approach is proposed in Section 5. After reporting the experiment results, we give a discussion in Section 6. Finally, we conclude the paper in Section 7.

2 Penn Discourse Tree Bank

The PDTB corpus follows the lexically grounded, predicate-argument approach in the D-LTAG framework [21]. It regards discourse connectives as discourse-level predicates that take exactly two text spans as their arguments. The span to which the connective is syntactically bound is labeled Arg2, while the other is labeled Arg1.

In the PDTB corpus, there are two types of discourse connectives, either explicit or implicit, among which explicit connectives occupy 53.49% (18459 tokens). Besides, each connective is assigned a sense from a three-level hierarchy, as shown in table 1 [19]. In this paper, we limit to the 16 Level-2 types.

Table 1. The hierarchy structures on discourse relationship of the PDTB corpus, Level-3 is not showed.

Level-1	Temporal	Contingency	Comparison	Expansion
Level-2	Synchrony	Cause	Contrast	Conjunction
	Asynchronous	Pragmatic Cause	Pragmatic Contrast	Instantiation
		Condition	Concession	Restatement
		Pragmatic Condition	Pragmatic Concession	Alternative
				Exception
				List

Example 1 shows an explicit discourse relation from the article wsj_2337 with the connective while underlined, the Arg1 span *italicized*, and the Arg2 span **bold**. The last line of this example shows the relation sense. It will be used as a running example throughout this paper.

(1) *Total advertising linage was modestly lower as classified-ad volume increased,* while **there was softer demand for detail and national ad linage.**

(Comparison-Contrast)

3 Related Work

Since the release of the PDTB corpus, much research work has been carried out on certain components of discourse parsing, with focus on explicit connectives. For example, Pitler and Nenkova[18] proposed various kinds of syntactic features to identify explicit connectives and determine the sense of an exlicit connective on marking the discourse relationship. On gold parse trees, they achieved 94.19% in F-measure and 94.15% in accuracy for the connective identification and the Level-1 sense disambiguation, respectively. Dinesh et al.[4] introduced a tree subtraction algorithm for labeling only subordinate discourse connectives [2]. Wellner and Pustejovsky [22] proposed several machine learning approaches to identify the head words of the two arguments for an explicit connective. Following this work, Elwell and Baldridge [5] combined several general and connective specific rankers to improve the performance of labeling the head words of the two arguments. Ghosh et al.[7] viewed argument labeling as a sequence labeling problem, and used two conditional random fields (CRFs) to label Arg1 and Arg2, respectively. Besides, they exploited the sense of connective to improve the performance of argument labeling.

Compared with the research on certain components of explicit discourse parsing, there is little work on constructing a complete end-to-end explicit discourse parser. The only exceptions are Lin et al. [10, 12], which designed an end-to-end explicit discourse parser into three components, i.e., a connective identifier, which identifies connectives from a text, a connective sense disambiguator, which determines the sense of a given connective, and an argument labeler, which identifies corresponding arguments for a given connective.

In this paper, we propose an end-to-end PDTB-style explicit discourse parser. Different from Lin et al. [10, 12], we decompose it into two components, i.e., a connective labeler and an argument labeler, by combing the connective identifier and and the connective sense disambiguator into a single connective labeler. Particularly, a joint learning approach is proposed to reduce error propagation and capture the interaction between the two components.

4 End-to-End Explicit Discourse Parsing

In this section, we will introduce our end-to-end PDTB-style explicit discourse parser, which consists of two components, i.e. a connective labeler, which identifies connectives from a text and determines their senses in classifying discourse relationship, and an argument labeler, which identifies corresponding arguments for a given connective.

4.1 Connective Labeling

Since only true connectives can convey the sense of a discourse relationship, we view the non-discourse usage of a connective candidate as special sense "Nil".

[2] According to the PDTB, from the grammatical usage, connective can divide in three types: subordinate, coordinate and discourse-adverbial

As this paper deals with the 16 Level-2 types in the connective sense hierachy, our connective labler becomes as a 17-category classfication task.

Table 2 lists various kinds of lexical and syntactic features employed in our connective labeler with the fourth column showing the feature instances corresponding to the gold parse tree of Example 1 as shown in Figure 1, taking [$_{IN}$ while] as the given connective candidate in question.

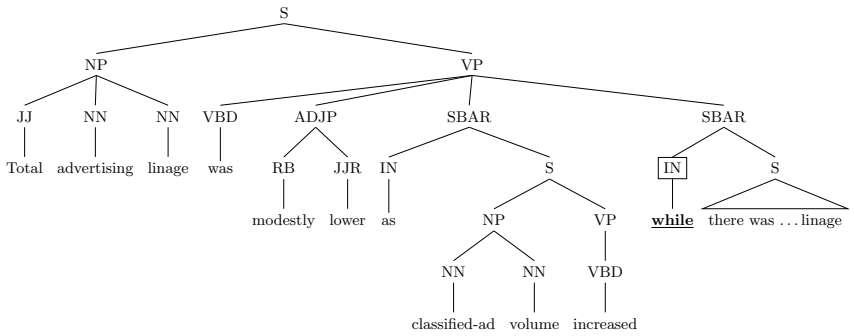

Fig. 1. Gold parse tree corresponding to Example 1

4.2 Argument Labeling

As stated above, the PDTB views a connective as the predicate of a discourse relation. Similar to semantic role labeling (SRL), we propose a constituent-based approach to do this task.

Firstly, a simple prune algorithm is employed to remove the constituents which are clearly not arguments to the connective in question, and all the remaining constituents are kept as the argument candidates of the given connective. Here, the pruning algorithm works recursively. Starting from the lowest node dominating the connective (called the connective target node), it collects all the siblings of the connective target node as argument candidates, moves on to the parent of the connective target node and collects its siblings as argument candidates. This process repeats until it reaches the root of the parse tree. Please note that, if the connective target node does not cover the connective exactly, the children of the connective target node are also collected as argument candidates. On the gold parse tree as shown in Figure 1 and given [$_{IN}$ while] as connective. We can get five argument candidates, i.e. [$_{S}$ there was . . . linage], [$_{SBAR}$ as classified-ad volume increased], [$_{ADJP}$ modestly lower], [$_{VBD}$ was], [$_{NP}$ Total advertising linage].

Secondly, a machine learning approach is employed to determine the role of each argument candidate as whether "Arg1", "Arg2", or "Nil". Prasad et al. [19] reported that Arg1 can be located within the same sentence as the connective

Table 2. Features employed in our discourse relation classification

Cat.	Feature	Description	Example
Lex.	cStr	the string of the candidate connective (case-sensitive)	while
	cStrLc	the lower case string of the candidate connective	while
	cCat	the syntactic usage of the candidate connective: subordinate, coordinate or discourse adverbial	subordinate
	cPos	the PoS of the given connective (when the connective is a phrase, we combine PoS of each token)	IN
	pWPos	the previous word string and its PoS	increased, VBD
	nWPos	the next word string and its PoS	there, EX
Syn.	selfCat	the highest node which covers only the candidate connective	IN
	pCat	the parent node of the *selfCat* node	SBAR
	lSib	the left sibling of the *selfCat* node	"Nil"
	rSib	the right sibling of the *selfCat* node	S
	rVp	whether the right sibling contains a VP	yes
	pToRt	the syntactic path from *selfCat* node to the root	IN ↑ SBAR ↑ VP ↑ S
	cPToRt	the compressed syntactic path from *selfCat* node to the root	IN ↑ SBAR ↑ VP ↑ S

(SS), in some previous sentence of the connective (PS), or in some sentence following the sentence containing connective (FS). Our statistics on the PDTB corpus shows that in the PS case, the distribution of immediate previous sentence as Arg1 accounts 76.9% (5549 of 7215). Thus we can resolve the PS case by viewing the root of the parse tree of the immediate previous sentence as a special argument candidate. Table 3 lists various kinds of lexical and syntactic features used in our argument labeler, on reflecting the properties of the connective, the argument candidate and the relationship between them, with the fourth of column showing the feature instances corresponding to Figure 1, given$[_{IN}$ while] as the connective and $[_{S}$ there was ... linage] as the argument candidate in question.

Finally, we merge all the determined candidates to obtain Arg1 and Arg2 respectively.

5 Joint Learning via Structured Perceptron

With the sequential pipeline architecture as described in Section 4, our two-components explicit discourse parser ignores the interaction between the connective labeler and the argument labeler. To reduce error propagation and capture the interaction between the two components, we propose a joint learning approach via structured perceptron.

Table 3. Features used in our argument labeling

Cat.	Feature	Description	Example
Lex.	cStr	the string of the given connective (case-sensitive)	while
	connStrLc	the lowercase of the given connective	while
	cCat	the syntactic usage of the given connective: subordinate, coordinate, or discourse adverbial	subordinate
	cSen	the Level-2 relation sense conveyed by the given connective	Contrast
Syn.	nLSib	number of left siblings of the connective	0
	nRSib	number of right siblings of the connective	1
	ctOfNode	the context of the constituent node. We use PoS combination of the constituent, its parent, left sibling and right sibling to present this feature. When no parent or siblings, it is marked Nil	S-SBAR-IN-Nil
	pToConn	the syntactic path from the parent node the given connective to the constituent node	S ↑ SBAR ↓ IN
	rPOfNode	the relative position of the constituent node to the given connective: left, right or previous	right
	phToCSib	the syntactic path from the parent node of the connective to the constituent node and whether the number of the left siblings of the connective is greater than one	S ↑ SBAR ↓ IN:¡1

Structured perceptron is an extension to the stand linear perceptron for structured prediction, which was proposed in [2]. Given an instance $x \in X$, which in our case is a set of constituent nodes in a parse tree, the algorithm involves the following decoding problem which finds the best configuration $d \in Y$ according to the current model w:

$$d = \underset{d' \in Y(x)}{\arg\max} \quad \mathbf{w} \cdot \mathbf{f}(x, d') \tag{1}$$

where $\mathbf{f}(x, d')$ represents the feature vector of instance x under configuration d'.

5.1 Training

The perceptron learns the model w in an online fashion. Let $D = (x^i, y^i)_{i=1}^N$ be the set of training instances. In each iteration, the algorithm finds the best configuration d for instance x under current model w (Eq 1). If d is incorrect, the weights are updated as follows:

$$\mathbf{w} = \mathbf{w} + \mathbf{f}(x, y) - \mathbf{f}(x, d) \tag{2}$$

Figure 2 describes the general training framework of structured perceptron algorithm. The key step of the training and testing is the decoding procedure,

Input: Training data $D = (x^i, y^i)_{i=1}^N$, T iteration number, N training data number
Output: Model parameters $\mathbf{w}$

Initialization: Set $\mathbf{w} = 0$
for $t \leftarrow 1$ **to** T **do**
  for $i \leftarrow 1$ **to** N **do**
    $d^i \leftarrow \texttt{decode}(x^i, y^i, \boldsymbol{w})$
    if $d^i \neq y^i$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{f}(x^i, y^i_{[1:|d^i|]}) - \mathbf{f}(x^i, d^i)$
    end
  end
end

Fig. 2. General structured perceptron training framework. Here $y_{[1:i]}$ indicates previous i elements.

which aims to find the best configuration under current model parameters. As like many NLP tasks (e.g., syntactic parsing, part-of-speech tagging) have too many states, the exact inference is often intractable. Instead, we employ beam-search to perform inexact inference. Collins and Roark [3] proposed the early-update idea for inexact inference, and later Huang et al [8] proved that structured perceptron could converge even under exact inference, and they pointed out "violation" is just the need for the algorithm. Based on these suggestions and in order to reduce overfitting, we use a variant of stand perceptron called averaged perceptron [2] with beam-search for inexact inference to train our explicit discourse parser.

5.2 Decoding

Our decoding algorithm will joint two components. Here we use $\mathcal{R}$ to denote the discourse sense label alphabet, where $\mathcal{R}$ consists of 16 Level-2 sense types and a "Nil" sense which indicates non-discourse usage. Similarly, $\mathcal{A} = \{Arg1, Arg2, Nil\}$ denotes the argument label alphabet. We use $\mathcal{K}$ to denote the beam-size for the decoding procedure.

Let $x = (c, a_1, \ldots, a_m)$ be input of decoding algorithm, where c represents a given connective, and $a_1, \ldots, a_m$ are the pruning argument candidates. We use $y = (g(c), g(a_1), \ldots, g(a_m))$ to denote the corresponding oracle output, where function $g(\cdot)$ maps current constituent node to its output label. Consider Example 1, we have c=[$_{IN}$ while], and a_1=[$_S$ there was ... linage], ...,a_5=[$_{NP}$ Total advertising linage]. Finally, the input is $x = (c, a_1, a_2, a_3, a_4, a_5)$, and according to the annotation, the output is $y = (Contrast, Arg2, Arg1, Arg1, Arg1, Arg1)$.

Figure 3 shows the beam-search algorithm with early-update for explicit discourse parsing. For each instance, there are two sub-steps:

– **connective labeling** We enumerate all possible sense label alphabet, and select K best sense candidates for current connective.

Input: Instance $x = (c, a_1, \ldots, a_m)$, and the oracle output y
$\mathcal{K}$: beam size; $\mathcal{R}$: relation sense label alphabet; $\mathcal{A}$: argument label alphabet
Output: the best prediction d for instance x

Initialization: $B \leftarrow [\epsilon]$, t $\leftarrow [\epsilon]$
for $d^{'} \in B$ **do** `// search relation senses`
  $t \leftarrow t \cup \{d^{'} \oplus r | r \in \mathcal{R}\}$
end
$B \leftarrow$ `K-Best(`t, $\mathcal{K}$, c, f_c`) // `f_c`:connective feature function`
if $y_{[1]} \notin B$ **then**
  return $B_{[0]}$ `// early-update`
end
for $i = 1$ **to** m **do** `// search argument roles`
  for $d^{'} \in B$ **do**
    $t \leftarrow t \cup \{d^{'} \oplus s | s \in \mathcal{A}\}$
  end
  $B \leftarrow$ `K-Best(`t, $\mathcal{K}$, a_i, f_a`) // `f_a`:argument feature function`
  if $y_{[1:i+1]} \notin B$ **then**
    return $B_{[0]}$ `// early-update`
  end
end
return $B_{[0]}$

Fig. 3. Decoding algorithm for explicit discourse parsing, $d \oplus s$ means append s in d

– **argument labeling** After the relation step, we traverse all configurations in the beam. Then the procedure would select K best argument labels based on previous configuration.

After the second step, the rank of different sense assignments can be changed because of the argument candidates selection. Likewise, the choice of later argument candidates would be affected by previous argument assignments.

5.3 Features and Representation

In this joint framework, we use the same features as described in Table 2 and Table 3. In general, each feature f is a function $f : X \times Y \rightarrow R$, which maps x and y to a feature value. In this framework, we use indicator function to represent each feature like the following format:

$$f_1(x, i, y_i) = \begin{cases} 1 \text{ if } y_i = \text{“Contrast”} \text{ and } x_i = \text{“while”} \\ 0 \text{ otherwise} \end{cases}$$

6 Experimentation

We have systematically evaluated our end-to-end explicit discourse parser on the PDTB corpus, with focus on the effectiveness of joint learning via structured perceptron.

6.1 Experimental Setting

We follow the PDTB recommendation [17] to use Section 02-21 in the PDTB for training, Section 22 for development, and Section 23 for testing. All staged classifiers are trained with the OpenNLP maximum entropy package[3] with default parameters.

We evaluate our discourse parser using gold and automatic parse trees, respectively. We use the NIST text segmenter[4] to split sentences and the Charniak parser[5] to parse sentences.

In the PDTB, some explicit relations are annotated with two senses. During training we chose the first sense as training instance. During test if the classifier assigns either of the two senses, we consider it as correct. For fair comparison with the state-of-the-art system, we report 16 types sense performance excluding "Nil" senses which indicate non-discourse connective usage. We evaluate our argument labeling using exact match metric [15] without all punctuation at the boundaries.

6.2 Parameters Selection

There are two important parameters in our joint learning framework: the maximum iteration number T and the beam size K. The harmonic mean of discourse relation classification F_1 and two arguments both right F_1 using gold parse trees is employed to measure the overall performance of our system on the development set.

Figure 4 shows the learning curves under four different beam size settings. As we can see that almost all curves converge around iteration 19. We also find that our system can achieve the best performance on the development set when beam-size $K = 4$. At this point, discourse relation classification achieves 83.67% in F_1, argument labeling achieves 58.61% in two arguments both right F_1, and the harmonic mean equals 68.93%. Therefore we set the maximum iteration number $T = 19$ and beam-size $K = 4$ for the remaining experiments.

6.3 Results and Analysis

For comparison, We also construct an end-to-end explicit discourse parser in pipeline way (thereafter called as "our pipeline system"). Table 4 lists the re-

[3] http://maxent.sourceforge.net/
[4] http://duc.nist.gov/duc2004/software/duc2003.breakSent.tar.gz
[5] ftp://ftp.cs.brown.edu/pub/nlparser

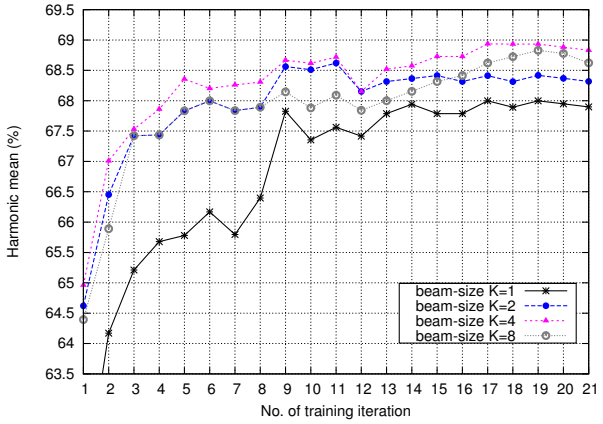

Fig. 4. Learning curves using harmonic mean on development set under different beam size settings

Table 4. Performance on test set using gold stand parse tree and charniak auto parse tree

	Systems	Relation Classification (%)			Argument Labeling (%)		
		P	R	F_1	Arg1 F_1	Arg2 F_1	Both F_1
Gold	Lin et al.[12]	83.19	82.65	82.92	57.64	79.80	52.29
	Our pipeline system	81.93	79.09	80.49	61.08	80.93	54.13
	Our joint system	82.93	81.04	81.97	60.27	78.25	53.48
Auto	Lin et al.[12]	81.19	80.04	80.61	47.68	70.27	40.37
	Our pipeline system	81.04	77.36	79.16	49.78	75.28	43.35
	Our joint system	81.55	79.96	80.74	53.94	73.41	46.94

sults of three different systems on test set using gold and automatic parse trees, respectively. [6] From the results, we can find that:

– On the necessary of independent connective identification:
In comparison with Lin et al. [12], using gold and automatic parse trees, the performance of discourse relation classification of our pipeline system reduces about 2.5% and 1.5% in F-measure, respectively. Just noted above, the only difference between these two systems is with or without independent connective identification component. With it, and selecting specific features for connective identification and discourse relation classification respectively, we can achieve better performance of discourse relation classification. Without it, however, we can simplify the following joint learning framework and be helpful for unified resolving of explicit and implicit discourse relations. Fur-

[6] All the improvements in this paper are significant with $p < 0.001$.

thermore, joint learning approach can improve the performance of discourse relation classification without independent connective identification.

– On the impact of parse trees:
  Three systems achieve better results on both components using gold parse trees than automatic parse trees.
  For relation classification, using automatic parse trees, the performance of Lin et al. [12] reduces about 2.3% in F-measure. Accordingly, our pipeline and joint learning systems reduce about 1.3% and 1.2% in F-measure, respectively. In comparison with Lin et al. [12], our connective labeler reduces the dependency to the parser's performance. We employ the same features (containing the features employed in connective identification and discourse relation classification) as Lin et al. [12] to classify discourse relation. The difference between our approach and Lin et al. [12] is that we cast connective identification as a part of discourse relation classification. So without independent connective identification component can reduce the dependency to the parser.
  For argument labeling, all the three systems heavily depend on the parse trees. Using automatic parse trees, the performance of Lin et al. [12] and our pipeline system reduce more than 10% in both arg1 and arg2 F-measure. Accordingly, the performance of our joint learning system reduces about 6.5% in both arg1 and arg2 F-measure. So our joint learning approach can reduce argument labelings' dependency to the parser.
– On the impact of joint learning:
  In comparison with our pipeline system, using gold parse trees, our joint learning approach can improve the performance of discourse relation classification and slightly reduce the performance of argument labeling. It may be due to that our constituent-based argument labeling system can work well with gold parse trees and the additional iteration in our joint learning system will introduce some noise. When using automatic parse trees, our joint learning approach can improve the performance of both discourse relation classification and argument labeling.
  In comparison with Lin et al. [12], using gold or automatic parse trees, our joint learning approach can achieve better performance on argument labeling and comparable performance on discourse relation classification.

7 Conclusion

In this paper, we focus on building an end-to-end PDTB-style explicit discourse parser by decomposing it into two components, i.e., a connective labeler, which identifies connectives from a text and determines their senses in classifying discourse relationship, and an argument labeler, which identifies corresponding arguments for a given connective. Particularly, to reduce error propagation and incorporate the interaction between the two components, a joint learning approach via structured perceptron is proposed. Experimental results show that our end-to-end explicit discourse parser can perform well using both gold and automatic parse trees.

Acknowledgements

This work is supported by grants from National Natural Science Foundation of China (No. 61333018, No. 6127320), and National 863 program of China (No.2012AA01112).

References

1. Barzilay, R., Lapata, M.: Modeling local coherence: An entity-based approach. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05). pp. 141–148. Association for Computational Linguistics, Ann Arbor, Michigan (June 2005), http://www.aclweb.org/anthology/P05-1018
2. Collins, M.: Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In: Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10. pp. 1–8. Association for Computational Linguistics (2002)
3. Collins, M., Roark, B.: Incremental parsing with the perceptron algorithm. In: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics. p. 111. Association for Computational Linguistics (2004)
4. Dines, N., Lee, A., Miltsakaki, E., Prasad, R., Joshi, A., Webber, B.: Attribution and the (non-)alignment of syntactic and discourse arguments of connectives. In: Proceedings of the Workshop on Frontiers in Corpus Annotations II: Pie in the Sky. pp. 29–36. CorpusAnno '05, Association for Computational Linguistics, Stroudsburg, PA, USA (2005), http://dl.acm.org/citation.cfm?id=1608829.1608834
5. Elwell, R., Baldridge, J.: Discourse connective argument identification with connective specific rankers. In: Semantic Computing, 2008 IEEE International Conference on. pp. 198–205 (2008)
6. Ghosh, S.: End-to-End Discourse Parsing with Cascaded Structured Prediction. Ph.D. thesis, University of Trento (2012)
7. Ghosh, S., Johansson, R., Riccardi, G., Tonelli, S.: Shallow discourse parsing with conditional random fields. In: Proceedings of 5th International Joint Conference on Natural Language Processing. pp. 1071–1079. Asian Federation of Natural Language Processing, Chiang Mai, Thailand (November 2011), http://www.aclweb.org/anthology/I11-1120
8. Huang, L., Fayong, S., Guo, Y.: Structured perceptron with inexact search. In: Proceedings of NAACL 2012 (2012)
9. Lin, Z., Liu, C., Ng, H.T., Kan, M.Y.: Combining coherence models and machine translation evaluation metrics for summarization evaluation. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1006–1014. Association for Computational Linguistics, Jeju Island, Korea (July 2012), http://www.aclweb.org/anthology/P12-1106
10. Lin, Z., Ng, H.T., Kan, M.Y.: A pdtb-styled end-to-end discourse parser. Technical report, School of Computing, National University of Singapore (2010)
11. Lin, Z., Ng, H.T., Kan, M.Y.: Automatically evaluating text coherence using discourse relations. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. pp. 997–1006. Association for Computational Linguistics (2011)
12. Lin, Z., Ng, H.T., Kan, M.Y.: A pdtb-styled end-to-end discourse parser. Natural Language Engineering FirstView, 1–34 (8 2013), http://journals.cambridge.org/article_S1351324912000307

13. Marcus, M., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of english: The penn treebank. Computational Linguistics 19(2), 313–330 (June 1993)
14. Meyer, T., Webber, B.: Implicitation of discourse connectives in (machine) translation. In: Proceedings of the Workshop on Discourse in Machine Translation. pp. 19–26. Association for Computational Linguistics, Sofia, Bulgaria (August 2013), http://www.aclweb.org/anthology/W13-3303
15. Miltsakaki, E., Prasad, R., Joshi, A., Webber, B.: The penn discourse treebank pp. 2237–2240 (2004), http://www.lrec-conf.org/proceedings/lrec2004/pdf/618
16. Ng, J.P., Kan, M.Y., Lin, Z., Feng, W., Chen, B., Su, J., Tan, C.L.: Exploiting discourse analysis for article-wide temporal classification. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. pp. 12–23. Association for Computational Linguistics, Seattle, Washington, USA (October 2013), http://www.aclweb.org/anthology/D13-1002
17. the PDTB Research Group: the Penn Discourse Treebank 2.0 Annotation Manual (December 2007)
18. Pitler, E., Nenkova, A.: Using syntax to disambiguate explicit discourse connectives in text. In: Proceedings of the ACL-IJCNLP 2009 Conference Short Papers. pp. 13–16. Association for Computational Linguistics, Suntec, Singapore (August 2009), http://www.aclweb.org/anthology/P/P09/P09-2004
19. Prasad, R., Dinesh, N., Lee, A., Miltsakaki, E., Robaldo, L., Joshi, A., Webber, B.: The penn discourse treebank 2.0 pp. 2961–2968 (2008), http://www.lrec-conf.org/proceedings/lrec2008/pdf/754
20. Prasad, R., Joshi, A., Webber, B.: Exploiting scope for shallow discourse parsing. In: Chair), N.C.C., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M., Tapias, D. (eds.) Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10). European Language Resources Association (ELRA), Valletta, Malta (may 2010)
21. Webber, B.: D-ltag: Extending lexicalized tag to discourse. Cognitive Science 28(5), 751–779 (2004)
22. Wellner, B., Pustejovsky, J.: 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). pp. 92–101. Association for Computational Linguistics, Prague, Czech Republic (June 2007), http://www.aclweb.org/anthology/D/D07/D07-1010