Finding Arguments as Sequence Labeling in Discourse Parsing

Ziwei Fan, Zhenghua Li*, Min Zhang
Soochow University, Suzhou, China
fanniufen@gmail.com, {zhli13,minzhang}@suda.edu.cn

Abstract

This paper describes our system for the CoNLL-2016 Shared Task on Shallow Discourse Parsing on English. We adopt a cascaded framework consisting of nine components, among which six are casted as sequence labeling tasks and the remaining three are treated as classification problems. All our sequence labeling and classification models are implemented based on linear models with averaged perceptron training. Our feature sets are mostly borrowed from previous works. The main focus of our effort is to recall cases when Arg1 locates at sentences far before the connective phrase, with some yet limited success.

1 General Description

This paper descirbes our participating system for CoNLL-2016 discourse parsing shared task (Xue et al., 2016). We participate in the closed track, and due to the time limitation, we focus on English. Given an document, which contains several paragraphs and each paragraph is composed of a few sentences, discourse parsing aims to identify explicit and non-explict discourse relations, including explicit connnective phrases (CP), explicit/non-explicit arguments and senses. Figure 1 presents a graphical illustration of the task.

Following the official requirement, we use Section 2-21 of the PDTB 2.0 (Prasad et al., 2008; Prasad et al., 2014) as the training data, Section 22 as the development data, and Section 23 as the test data. A blind test is also used for evaluation. Table 1 presents the data statistics.

Due to the complexity of the task, our system follows previous practice and employs a cascaded framework and comprises 9 components, as shown in Figure 2. In the following, we will introduce each component in detail. The codes are released at `http://hlt.suda.edu.cn/~zhli` for future research study.

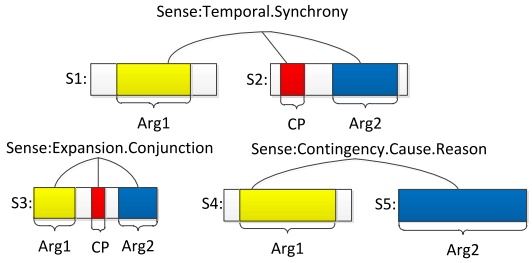

Figure 1: Illustration of discourse parsing.

	Train	Dev
Document	2000	100
Paragraph	17619	783
Sentence	38967	1675
Explicit relations	14722	680
Non-explicit relations	17813	756

Table 1: Data statistics of English.

2 Classification and Sequence Labeling Based on Linear Model

In this work, we implement our classification and sequence labeling models based on linear model due to its simplicity and good performance on variety of natural language processing tasks (Collins, 2002). Given an input instance x and a label y, a linear model defines the score of labeling x as y:

$$Score(x, y) = \mathbf{w} \cdot \mathbf{f}(x, y)$$

*Correspondence author.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 150–157,
Berlin, Germany, August 7-12, 2016.

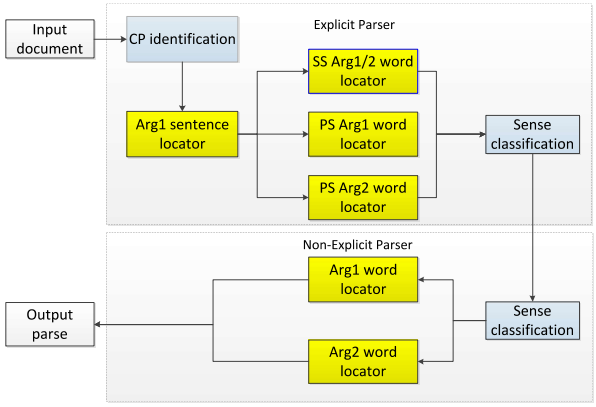

Figure 2: Framework of our system.

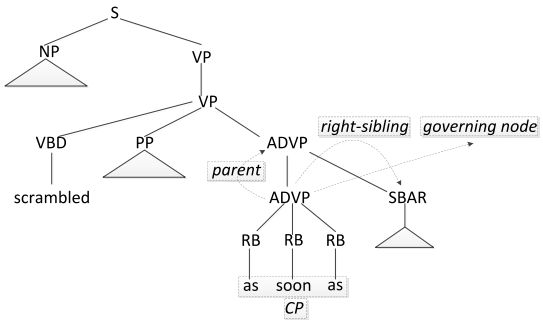

Figure 3: Example of a parse tree from which we extract features.

where $\mathbf{f}(.)$ is a feature vector constructed according to a hand-crafted feature template list and $\mathbf{w}$ is the corresponding feature weight vector.

The decoding task in the linear model is to find the maximum-scoring label:

$$\hat{y} = \arg\max_{y} Score(x, y)$$

To learn $\mathbf{w}$, we use the standard online training procedure, which use one instance for feature weight update at a time:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \mathbf{f}(x, y^*) - \mathbf{f}(x, \hat{y})$$

where t is the global time of feature weight updates (i.e., the number of instances used for feature weight updates so far); $\hat{y}$ is the best label according to the current feature weights $\mathbf{w}^{(t)}$; y^* is the gold-standard label. In this sense, online training is also known as decoding-based training, meaning that decoding is invoked during training.

Following Collins (2002), after training, we use the averaged feature weights $\sum_{t=1}^{T} \mathbf{w}^{(t)}/T$ for final evaluation, which is known as *averaged perceptron*.

For sequence labeling tasks, y is a sequence of labels instead of a single label. Besides many unigram features which only consider the label in the current position, as used in multi-class classification tasks, we also use label-transition bigram features in our sequence labeling models. The training procedure is nearly the same with the case of classification problems, except that a dynamic programming based decoding algorithm is need for exact search for the optimal label sequence $\hat{y}$.

3 CP Identification

Given an input document, the first task is to extract all connective phrases (CPs) (e.g., "so that") in the document,[1] which we refer to as *CP identification*. We directly adopt the method described in previous works (Wang and Lan, 2015; Kong et al., 2015), and take two steps for this task.

1. **Candidate CP extraction.** We extract all candidate CPs in the input document by exact matching with a phrase dictionary. If a string in a sentence exactly matchs a phrase in the dictionary, it then is considered as a candidate CP and will be verified in the second step. The dictionary is provided by the official organizer and contains 100 phrases.

2. **CP classification.** In this step, we use a statistical classifier based on the linear model to check whether each candidate CP functions as a CP or not.

We directly borrow and merge the features proposed in Lin et al. (2014) and Pitler and Nenkova (2009), as listed in Table 2. We spent little time on feature engineering, since we found our model achieved similar accuracy to last year's best system (Wang and Lan, 2015) using these features. On the dev data, our proposed CP identification method achieves 95.23% precison, 93.96% recall, and 94.59% F score. Figure 3 gives an example of the parse tree to better illustrate the features.

[1] Since a discourse connective may contain more than one words, we use "connective phrase" as a more accurate terminology in this paper.

Lexical:	$word$(CP), POS(CP), $POS(prev_1$(CP)), $POS(next_1$(CP)), $word(prev_1$(CP)) + $word$(CP), $POS(prev_1$(CP)) + POS(CP), $POS(prev_1$(CP)) + $POS(first_1$(CP)), $word$(CP) + $word(next_1$(CP)), POS(CP) + $POS(next_1$(CP)), $POS(last_1$(CP)) + $POS(next_1$(CP))
Syntactic:	$label(govern\_node$(CP)), $label(parent(govern\_node$(CP))), $label(l\_sib(govern\_node$(CP))), $label(r\_sib(govern\_node$(CP))), $path\_to\_root(govern\_node$(CP)),

Table 2: Features for CP classification. $word(p)$: word sequence of the given phrase p; $POS(p)$: POS tag sequence of p; $prev_1(p)$: the first previous position of p; $next_1(p)$: the first next position of p; $first(p)$: the first position of p; $last(p)$: the last position of p; $govern\_node(p)$: the highest node in the parse tree that covers p; $parent(o)$: the parent node of the node o in the parse tree; $l\_sib(o)$: the left sibling node of o in the parse tree; $r\_sib(o)$: the right sibling node of o in the parse tree; $path\_to\_root(o)$: the label sequence along the path from the node o to the root node in the parse tree.

Distance	Train	Dev
0	8880	447
1	4047	162
2	560	18
3	244	11
4	131	6
5	79	9
$\geq$6	202	7

Table 3: Distribution of instances in terms of distance between the sentence containing Arg1 and the sentence containing CP, where "0" means that Arg1 locates at the same sentence containing CP, "1" means that Arg1 is in the previous sentence of the sentence containing CP, and so on. We throw instances in which Arg1 or Arg2 locates at multiple sentences.

#Sentence	Train	Dev
1	14231 (0)	661 (0)
2	364 (44)	14 (1)
3	70 (18)	4 (2)
$\geq$4	57(22)	1(1)

Table 4: Distribution of instances in terms of the number of sentences that one Arg1 locates at, where the numbers in parenthesis mean the case when the sentences are discontinuous.

4 Explicit-Arg1 Sentence Locator: Sequence Labeling

As far as we know, most previous participating systems last year assume that Arg1 lies in the same sentence or the previous sentence of CP. However, we find that there exist many cases that Arg1 locates at longer-distance sentences from the CP. Table 3 shows data statistics regarding the sentence-level distance of Arg1 and CP.

We also find that there are cases that Arg1 locates at more than one sentences, and the sentences may be discontinuous, as shown in Table 4. However, for simplicity, in this work we throw away training instances when Arg1 locates at more than one sentence.

For the Explicit-Arg1 sentence locator, we adopt a sequence labeling model and try to recall cases of long-distance Arg1. The model starts from the sentence containing CP (with an index 0), and perform dynamic programming based search from right to left. For simplicity, we set the window size to 6, meaning that the model considers at most six sentences, from the 0^{th} sentence containing CP, to the 5^{th} sentence in front. For the features, we directly adopt those described in Lin et al. (2014), Pitler et al. (2009), Pitler and Nenkova (2009), and Knott (1996).

Especially, we design a three-tag label set in order to enforce the model to return exactly one sentence with Arg1.

1. *Arg1_yes*: the current sentence contains Arg1.
2. *None_yes*: the current sentence does not contain Arg1, but some sentence in its right does contain Arg1.
3. *None_no*: the current sentence and all sentences in its right do not contain Arg1.

Using such label set, we can conveniently constrain the model not to return a sequence where Arg1 occurs more than once by *constrained decoding*. The idea is that during decoding we do not allow a set of illegal transitions: $\{$*Arg1_yes* $\rightarrow$ *Arg1_yes*, *Arg1_yes* $\rightarrow$ *None_no*, *None_yes* $\rightarrow$ *None_no*, *None_yes* $\rightarrow$ *Arg1_yes*$\}$.

Label Set	Constrained	w/o Error Propagation Accuracy	Explicit, w/ Error Propagation			
			Arg1	Arg2	Arg1/2	Sense
3 tags	none	86.18	50.85	73.39	42.25	38.70
	test	**88.09**	**51.00**	**73.98**	**42.70**	**39.30**
	train & test	81.03	48.78	70.42	40.03	36.53
2 tags	none	85.74	51.15	73.68	42.70	39.15
PS/SS classification	none	**89.12**	**51.89**	**74.13**	**43.14**	**39.61**

Table 5: Results of different Explicit-Arg1 sentence locators on dev data.

Label Set	Constrained	Distance					
		0	1	2	3	4	5
3 tags	none	424 (424)	256 (**162**)	0	0	0	0
	test	458 (**445**)	222 (154)	0	0	0	0
	train & test	415 (409)	197 (131)	34 (**5**)	12 (**3**)	6 (0)	16 (**3**)
2 tags	none	497 (**444**)	182 (139)	1 (0)	0	0	0
PS/SS classification	none	445 (**444**)	235 (**162**)	–	–	–	–

Table 6: Result analysis of different Explicit-Arg1 sentence locators on dev data. We report the distribution of the outputs of each model in terms of distance between the predicted sentence containing Arg1 and the sentence with CP, where numbers in parenthesis count correct prediction according to gold-standard answers.

As discussed in Section 2, our model is based on a linear model and uses online training to learn the feature weights. Moreover, online training is a decoding-based training procedure, meaning that a best result is found by the decoding procedure based on the current feature weights, and the result is then used for weight update. Therefore, we have three options for applying constrained decoding.

1. **None**: We do not use any constraints and apply post-processing to handle inconsistent outputs. When the model classifies multiple sentence into *Arg1*, we only keep the nearest sentence tagged as *Arg1*. If no sentence is tagged as *Arg1*, we use the sentence containing CP as *Arg1*.

2. **Test**: We add constraints during the test phase. In the train phase, the optimal $\hat{y}$ is directly used for feature weight update without post-processing. However, we may also post-process $\hat{y}$ so that it contains exactly one Arg1 label before feature update weight during training, which we leave for future work.

3. **Train & test**: We add constraints during both train and test phases.

For comparison, we also implement a model based on a two-tag label set of $\{$*Arg1*, *None*$\}$, in which we cannot guarantee the output label sequence always contains only one *Arg1* through constrained decoding. Therefore, we post-process the results in the similar way to the case of the three-tag model with no constraint.

Table 5 reports the results both with and without error propagation. The "PS/SS classification" model is our re-implementation of the method described in Wang and Lan (2015) under our linear model framework with only unigram features, which only considers the current and previous sentences of CP with a binary classifier. The three-tag model performs best with "test" constraints, and surprisingly worse with "train & test" constraints. Even though the "PS/SS classification" model is very simple, it is very competitive and achieves better results on the dev data than our proposed three-tag sequence labeling model. We will look into this issue in future.

Table 6 further investigates the ability of different models on recalling cases when the sentence containing Arg1 locates far before the sentence containing CP. Although using "train & test" constraints leads to bad performance, we actually find that the model can actually recall cases when Arg1 locates at long-distance sentences, whereas the model with "test" constraints and the model with "none" constraints almost always return re-

sults that Arg1 locates at the sentence with CP or the previous sentence. We will look into this problem in future.

5 Explicit-Arg1/2 Word Locator: Sequence Labeling

Data statistics show that for explicit relations, nearly all Arg2 locates at the the same sentence with CP. Therefore, based on the results of Arg1 sentence locator, we have two cases to handle: Arg1 and Arg2 locate at the same sentence with CP (SS), or Arg1 locates at a previous sentence of CP (PS). Then, we use three sequence labeling models to locate the exact words of Arg1/2. All three models perform at the level of words, and each time assign a "*Arg1/Arg2/None*" tag to a word.

Many systems in CoNLL-2015 (Xue et al., 2015) evaluation also treat Arg1/2 word location as a sequence labeling problem, and uses conditional random filed (CRF) based models (Stepanov et al., 2015; Nguyen et al., 2015; Lalitha Devi et al., 2015) or recurrent neural networks (RNN) (Wang et al., 2015).

5.1 Explicit: SS Arg1/2 Word Locator

For the SS case, the sequence labeling model performs decoding from left to right on the CP sentence, and classifies each word into four categories: "*Arg1/Arg2/None/CP*". The words inside the CP (given as input) are fixed to be "*CP*" before decoding, and all other words are not allowed to be tagged as "*CP*" during decoding. For the features, we directly adopt those described in Lin et al. (2014), Pitler et al. (2009), Pitler and Nenkova (2009), Knott (1996), Kong et al. (2015). On the dev data, the model achieves an word-level accuracy of 53.45% without error propagation.

5.2 Explicit: PS Arg1 Word locator

For the PS case, we first use a sequence labeling model to locate the words of Arg1. The model perform decoding from left to right on the sentence returned by the Explicit-Arg1 sentence locator, and classifies each word into two categories: "*Arg1/None*". For the features, we directly adopt those described in Lin et al. (2014), Pitler et al. (2009), Knott (1996). On the dev data, the model achieves an word-level accuracy of 67.14% without error propagation.

	True Positive	False Positive
Train	16940	4850
Dev	718	200

Table 7: Distribution of adjacent sentences having non-explicit relation.

5.3 Explicit: PS Arg2 Word Locator

To locate the Arg2 words in the PS case, we use a sequence labeling model to perform decoding from left to right on the CP sentence, and classifies each word into two categories: "*Arg2/None*". Please note that the words in CP always have a special tag "CP" when decoding. For the features, we directly adopt those described in Lin et al. (2014), Pitler et al. (2009), Wang and Lan (2015), Kong et al. (2015), Knott (1996). On the dev data, the model achieves an word-level accuracy of 67.14% without error propagation.

6 Explicit Sense Classification

After obtaining the CP and the Arg1/2 words, we then use a linear model based classifier to classify the sense of each explicit relation. We directly adopt the features described in Lin et al. (2014). On the dev data, the model achieves an accuracy of 87.65% without error propagation.

7 Non-explicit Sense Classification

After processing the explicit relations, we then turn to the problem of non-explicit relation parsing. As suggested by the official organizer, if two adjacent sentences do not have explicit relation after previous processing, we consider them as a candidate sentence pair having non-explicit relation. Please note that we only consider sentence pairs that are in the same paragraph.

As far as we know, most previous work directly considers all adjacent sentences without explicit relation as having non-explicit relation, and use a classifier to predict their non-explicit senses. However, our data statistics in Table 7 show that there exist many false non-explicit cases, which we call negative instances. We add a special tag "*None*" into the non-explicit sense set and use such false non-explicit cases as negative training instances , so that the trained classifier can make not-a-non-explicit-relation decision. However, our preliminary results show that adding negative instances does not improve parser performance on

Component	Dev			Test			Blind test		
	P	R	F	P	R	F	P	R	F
All Arg1 extractor	57.31	62.40	59.75	53.43	56.86	55.09	41.85	53.89	47.11
All Arg2 extractor	70.06	76.27	73.03	67.30	71.62	69.40	57.73	74.33	64.99
All Arg1&Arg2 extractor	47.84	52.08	49.87	42.75	45.50	44.08	33.66	43.34	37.90
All Sense	32.72	30.47	**31.56**	27.47	25.84	**26.63**	24.49	18.94	**21.36**
Explicit Connectives	93.53	95.07	94.29	94.69	94.79	94.74	89.57	92.57	91.04
Explicit Arg1 extractor	50.59	51.42	51.00	44.96	45.01	44.99	41.19	42.57	41.86
Explicit Arg2 extractor	73.38	74.59	73.98	72.05	72.13	72.09	68.71	71.00	69.84
Explicit Arg1&Arg2 extractor	42.35	43.05	42.70	37.38	37.42	37.40	32.91	34.01	33.46
Explicit Sense	39.16	39.45	39.30	32.97	32.97	32.97	27.99	26.98	27.47
Non-Explicit Arg1 extractor	62.17	72.31	66.86	59.74	67.44	63.36	41.81	68.08	51.80
Non-Explicit Arg2 extractor	67.06	78.00	72.12	62.99	71.11	66.81	48.39	78.80	59.96
Non-Explicit Arg1&Arg2 extractor	52.78	61.38	56.76	47.64	53.78	50.52	34.30	55.86	42.50
Non-Explicit Sense	26.12	22.56	24.21	21.83	19.35	20.51	19.95	12.10	15.06

Table 8: Official results of our system on the dev, test, and blind test datasets. "All" means both explicit and non-explicit relations.

the dev data. We will look into this problem in future.

For the features, we directly adopt those described in Lin et al. (2014), Pitler et al. (2009), Rutherford and Xue (2014), Kong et al. (2015). On the dev data, the model achieves an accuracy of 34.04% without error propagation.

8 Non-explicit Arg1/2 Word Locator: Sequence Labeling

According to data statistics, if two adjacent sentences have non-explicit relation, Arg1 locates at the first sentence while Arg2 locates at the second sentence. Therefore, we use two separate sequence labeling models to locate Arg1/2 words in the two sentences respectively. If the non-explicit sense is "*EntRel*", we directly label the whole first sentence as Arg1 and the whole second sentence as Arg2, according to data statistics. For the features, we directly adopt those described in Lin et al. (2014), Pitler et al. (2009), Wang and Lan (2015), Kong et al. (2015). On the dev data, the two models achieve word-level accuracy of 68.14% on Arg1 and 75.82% on Arg2 without error propagation.

9 Final Results

Table 8 shows the official results of our system on the dev, test and blind test datasets from the organizers through the TIRA platform (Potthast et al., 2014). Our system ranks the 7th place among 14 systems in both test and blind test datasets in the closed track of CoNLL-2016 shared task on shallow discourse parsing of English.

	Linear	Maximum Entropy
Explicit Sense	39.30	44.60 (+5.30)
All Sense	31.56	32.81 (+1.25)

Table 9: Comparison of the linear model and the maximum entropy model on Explicit relations with error propagation on dev data.

10 Explicit Sense Classification with a Maximum Entropy Model

After obtaining the evaluation results of all systems, we find that our system achieves clearly lower performances on sense classifications than other systems. Therefore, we replace the linear classification model with a log-linear maximum entropy model in the Explicit sense classification task. We use AdaGrad for deciding the feature update step (Duchi et al., 2011). Table 9 shows the results. We can see that using maximum entropy leads to large improvement.

We then try to replace the linear model with the maximum entropy model in the CP classification task, but obtain very little gain, possibly because the accuracy is already very high with the linear model. We plan to use the maximum entropy model for non-explicit sense classification.

11 Conclusions and Future Work

So far, our approach is composed of too many components without any interaction. In the future, we would like to pursue two directions. First, we will try to design a more principled and unified framework so that tasks at different levels can influence each other. Second, we plan to try other machine learning techniques such as neural networks for better representing and modeling discourse-level information.

Acknowledgments

The authors would like to thank the anonymous reviewers for the helpful comments. In building our system, we use a lot of codes of two participating teams last year, and we are very grateful for their kind sharing (Kong et al., 2015; Wang and Lan, 2015). We also would like to thank the organizer of this shared task for their hard work, especially in data annotation and preparation. This work was supported by National Natural Science Foundation of China (Grant No. 61502325, 61525205), and the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (No. 15KJB520031).

References

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP 2002*, pages 1–8.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Alistair Knott. 1996. A data-driven methodology for motivating a set of coherence relations.

Fang Kong, Sheng Li, and Guodong Zhou. 2015. The sonlp-dp system in the conll-2015 shared task. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 32–36, Beijing, China, July. Association for Computational Linguistics.

Sobha Lalitha Devi, Sindhuja Gopalan, Lakshmi S, Pattabhi RK Rao, Vijay Sundar Ram, and Malarkodi C.S. 2015. A hybrid discourse relation parser in conll 2015. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 50–55, Beijing, China, July. Association for Computational Linguistics.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, 20(02):151–184.

Son Nguyen, Quoc Ho, and Minh Nguyen. 2015. Jaist: A two-phase machine learning approach for identifying discourse relations in newswire texts. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 66–70, Beijing, China, July. Association for Computational Linguistics.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 13–16. Association for Computational Linguistics.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 683–691. Association for Computational Linguistics.

Martin Potthast, Tim Gollub, Francisco Rangel, Paolo Rosso, Efstathios Stamatatos, and Benno Stein. 2014. Improving the Reproducibility of PAN's Shared Tasks: Plagiarism Detection, Author Identification, and Author Profiling. In Evangelos Kanoulas, Mihai Lupu, Paul Clough, Mark Sanderson, Mark Hall, Allan Hanbury, and Elaine Toms, editors, *Information Access Evaluation meets Multilinguality, Multimodality, and Visualization. 5th International Conference of the CLEF Initiative (CLEF 14)*, pages 268–299.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *LREC*. Citeseer.

Rashmi Prasad, Bonnie Webber, and Aravind Joshi. 2014. Reflections on the penn discourse treebank, comparable corpora, and complementary annotation. *Computational Linguistics*.

Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *EACL*, volume 645, page 2014.

Evgeny Stepanov, Giuseppe Riccardi, and Ali Orkan Bayer. 2015. The UniTN discourse parser in conll 2015 shared task: Token-level sequence labeling with argument-specific models. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 25–31, Beijing, China, July. Association for Computational Linguistics.

Jianxiang Wang and Man Lan. 2015. A refined end-to-end discourse parser. In *Proceedings of the Nine-*

teenth Conference on Computational Natural Language Learning - Shared Task, pages 17–24, Beijing, China, July. Association for Computational Linguistics.

Longyue Wang, Chris Hokamp, Tsuyoshi Okita, Xiaojun Zhang, and Qun Liu. 2015. The dcu discourse parser for connective, argument identification and explicit sense classification. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 89–94, Beijing, China, July. Association for Computational Linguistics.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi Prasad, Christopher Bryant, and Attapol Rutherford. 2015. The conll-2015 shared task on shallow discourse parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 1–16, Beijing, China, July. Association for Computational Linguistics.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Bonnie Webber, Attapol Rutherford, Chuan Wang, and Hongmin Wang. 2016. The conll-2016 shared task on multilingual shallow discourse parsing. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning - Shared Task*, Berlin, Germany, August. Association for Computational Linguistics.