A Refined End-to-End Discourse Parser

Jianxiang Wang, Man Lan*
Shanghai Key Laboratory of Multidimensional Information Processing
Department of Computer Science and Technology,
East China Normal University, Shanghai 200241, P. R. China
51141201062@ecnu.cn, mlan@cs.ecnu.edu.cn*

Abstract

The CoNLL-2015 shared task focuses on shallow discourse parsing, which takes a piece of newswire text as input and returns the discourse relations in a PDTB style. In this paper, we describe our discourse parser that participated in the shared task. We use 9 components to construct the whole parser to identify discourse connectives, label arguments and classify the sense of Explicit or Non-Explicit relations in free texts. Compared to previous discourse parser, new components and features are added in our system, which further improves the overall performance of the discourse parser. Our parser ranks the first on two test datasets, i.e., PDTB Section 23 and a blind test dataset.

1 Introduction

An end-to-end discourse parser is given free texts as input and returns discourse relations in a PDTB style, where a connective acts as a predicate that takes two text spans as its arguments. It can benefit many downstream NLP applications, such as information retrieval, question answering and automatic summarization, etc. The extraction of exact argument spans and Non-Explicit sense identification have been shown to be the main challenges of the discourse parsing (Lin et al., 2014).

Since the release of Penn Discourse Treebank (PDTB) (Prasad et al., 2008), much research has been carried out on PDTB to perform the subtasks of a full end-to-end parser, such as identifying discourse connectives, labeling arguments and classifying Explicit or Implicit relations. To identify discourse connectives from non-discourse ones and to classify the Explicit relations, (Pitler and Nenkova, 2009) extracted syntactic features of connectives from the constituent parses, and showed that syntactic features improved performance in both subtasks. For the argument labeling subtask, (Ghosh et al., 2011) regarded it as a token-level sequence labeling task using conditional random fields (CRFs). (Lin et al., 2014) proposed a tree subtraction algorithm to extract the arguments. (Kong et al., 2014) adopted a constituent-based approach to label arguments. As for Implicit sense classification, (Pitler et al., 2009), (Lin et al., 2009) and (Rutherford and Xue, 2014) performed the classification using several linguistically-informed features, such as verb classes, production rules and Brown cluster pair. (Lan et al., 2013) presented a multi-task learning framework with the use of the prediction of explicit discourse connective as auxiliary learning tasks to improve the performance.

All of these research focus on the subtasks of the PDTB, and can be viewed as isolated components of a full parser. (Lin et al., 2014) constructed a full parser on the top of these subtasks, which contained multiple components joined in a sequential pipeline architecture including a connective classifier, argument labeler, explicit classifier, non-explicit classifier, and attribution span labeler. In this paper, we followed the framework of (Lin et al., 2014) to construct a discourse parser. However, our work differs from that of Lin's in that our system introduces new components and features to improve the overall performance. Specifically, (1) we build two different

Proceedings of the Nineteenth Conference on Computational Natural Language Learning: Shared Task, pages 17–24,
Beijing, China, July 26-31, 2015.

extractors for Arg1 and Arg2 respectively for labeling Explicit arguments in the case of PS (i.e., Arg1 is located in some previous sentences of the connective); (2) we add new features to capture more information for classification or recognition; (3) we build two different argument extractors for Non-EntRel relations in Non-Explicit; (4) we use the refined arguments to improve the Non-Explicit sense classification.

The organization of this work is as follows. Section 2 gives a sketch description of our parser in a flow chart and the function of every component in this architecture. Section 3 describes the components and features in detail. Section 4 reports the preliminary experimental results on the training and development dataset, and the final results on two test datasets are shown in Section 5. Section 6 concludes this work.

2 System Overview

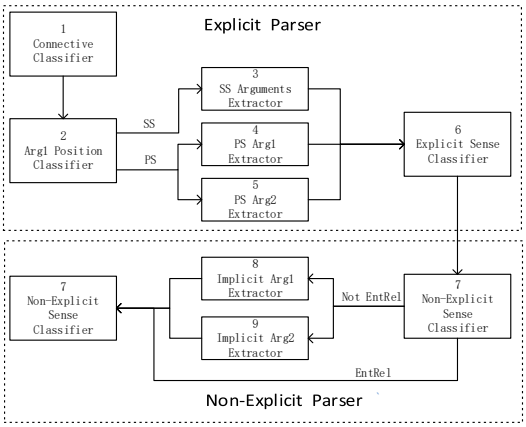

Figure 1: System pipeline for the discourse parser

We design the discourse parser as a sequential pipeline, shown in Figure 1, and the 9 components of our parser are listed as follows.

First, for texts with Explicit connective words:

(1) **Connective Classifier** is to identify the discourse connectives from non-discourse ones.

(2) **Arg1 Position Classifier** is to decide the relative position of Arg1 – whether it is located within the same sentence as the connective (SS) or in some previous sentence of the connective (PS).

(3) **SS Arguments Extractor** is to extract the spans of Arg1 and Arg2 in the SS case.

In the PS case, we build two extractors to identify the text spans for PS Arg1 and PS Arg2 respectively.

(4) **PS Arg1 Extractor** is to extract Arg1 for PS.

(5) **PS Arg2 Extractor** is to extract Arg2 for PS.

(6) **Explicit Sense Classifier** is to identify the sense that this Explicit connective conveys.

Second, for all adjacent sentence pairs within each paragraph, but not identified in any Explicit relation:

(7) **Non-Explicit Sense Classifier** is to classify the sense of each sentence pair into one of the Non-Explicit relation senses.

Since attribution is not annotated for EntRel relations, if the output of the above Non-Explict sense classifier is EntRel, we regard the previous sentence as Arg1 and the next one as Arg2. Otherwise, we build the following two argument extractors to label Arg1 and Arg2.

(8) **Implicit Arg1 Extractor** and (9) **Implicit Arg2 Extractor** extract Arg1 and Arg2 for Non-EntRel relations in Non-explicit, respectively.

3 Components and Features

Generally, our parser consists of 9 components, which compose an Explicit parser and a Non-Explicit parser. Most of features used in our parser are borrowed from previous work (Kong et al., 2014; Lin et al., 2014; Pitler et al., 2009; Pitler and Nenkova, 2009; Rutherford and Xue, 2014).

3.1 Explicit Parser

3.1.1 Connective Classifier

Since the input of the discourse parser is free text, the first thing we need to do is to identify all connective occurrences in text, and then to use the connective classifier to decide whether they function as discourse connectives or not.

For each connective occurrence *C*, we extract features from its context, part-of-speech (POS) and the parse tree of the connective's sentence. Note that $prev_1$ and $next_1$ indicate the first previous word and the first next word of connective *C* respectively. For a node in the parse tree, we use the POS combinations of the node, its parent, its children to represent the *linked context*.

The features we used for connective classsification consist of the following: (1) Pitler's: *C*

string (case-sensitive), *self-category* (the highest node in the parse tree that covers only the connective words), *parent-category* (the parent of the *self-category*), *left-sibling-category* (the left sibling of the *self-category*), *right-sibling-category* (the right sibling of the *self-category*), *C*-Syn interaction (the pairwise interaction features between the connective *C* and each category feature (i.e., self-category, parent-category, left-sibling-category, right-sibling-category)) , Syn-Syn interaction (the interaction features between pairs of category features); (2) Lin's: *C* POS, $prev_1 + C$ string, $prev_1$ POS, $prev_1$ POS + *C* POS, *C* string + $next_1$, $next_1$ POS, *C* POS + $next_1$ POS, path of *C*'s parent → root, compressed path of *C*'s parent → root; (3) our new proposed features: the POS tags of nodes from *C*'s parent → root, *parent-category linked context*, *right-sibling-category linked context*. Our three new features are considered to capture more syntactic context information of the connective *C* for connective classification.

3.1.2 Arg1 Position Classifier

After identifying the discourse connectives from the texts, we come to locate the positions of Arg1 and Arg2 of the connective *C*. Since Arg2 is defined as the argument with which the connective is syntactically associated, its position is fixed once we locate the discourse connective *C*. So we only need to identify the relative position of Arg1 as whether it is located within the same sentence as the connective (SS) or in some previous sentences of the connective (PS). We do not identify the case which Arg2 is located in some sentences following the sentence containing the connective (FS), because the statistical distribution of (Prasad et al., 2008) shows that less than 0.1% are FS for Explicit relations.

The features consist of the following: (1) Lin's: *C* string, *C* position (the position of connective *C* in the sentence: start, middle, or end), *C* POS, $prev_1$, $prev_1$ POS, $prev_1 + C$, $prev_1$ POS + *C* POS, $prev_2$, $prev_2$ POS, $prev_2$ + *C*, $prev_2$ POS + *C* POS; (2) our newly-proposed features: *C* POS + $next_1$ POS, $next_2$, path of *C* → root. Note that $prev_2$ and $next_2$ indicate the second previous word and the second next word of connective *C*, respectively.

3.1.3 Argument Extractor

After the relative position of Arg1 is classified as SS or PS in previous component, the argument extractor is to extract the spans of Arg1 and Arg2 for the identified discourse connectives. According to (Kong et al., 2014), Kong's constituent-based approach outperforms Lin's tree subtraction algorithm for the Explicit arguments extraction. However, Lin only focused on the SS case, and Kong treated the immediately preceding sentence as a special constituent for PS, which means that they just viewed the immediately preceding sentence as Arg1 and only extracted Arg2 for PS. So we only follow Kong's constituent-based approach to extract Arg1 and Arg2 for SS. However, for PS, we build two different extractors for Arg1 and Arg2 separately. Our intuition is that the two arguments have different syntactic and discourse properties and a unified model with the same feature set used for both may not have enough discriminating power.

SS Arguments Extractor: In the case of SS, we adopt (Kong et al., 2014)'s constituent-based approach without Joint Inference to extract Arg1 and Arg2.

For PS, we build two argument extractors for Arg1 and Arg2, respectively, as follows.

PS Arg1 Extractor: We consider the immediately previous sentence of connective *C* as the text span where Arg1 occurs and then build a extractor to label the Arg1 in it. Similar to Lin's Attribution span labeler, this extractor consists of two steps: splitting the sentence into clauses, and deciding, for each clause, whether it belongs to Arg1 or not. First we use nine punctuation symbols (...,.:;?!-∼) to split the sentence into several parts and use the SBAR tag in its parse tree to split each part into clauses. Second, we build a classifier to decide each clause whether it belongs to Arg1 or not.

On the one hand, the attribution relation is annotated in PDTB, which expresses the "ownership" relationship between abstract objects and individuals or agents. However, the attribution annotation is excluded in CoNLL-2015 (Xue et al., 2015). Therefore we borrow several attribution features from (Lin et al., 2014) in order to distinguish the attribution-related span from others. On the other hand, according to the *minimality principle* of PDTB, the argu-

ment annotation includes the minimal span of text that is sufficient for the interpretation of the relation. Since connectives have very close relationship with discourse relation, we consider to adopt connective-related features to capture text span for relation. We choose the following features: (1) attribution-related features from (Lin et al., 2014): lemmatized verbs in *curr*, the first term of *curr*, the last term of *curr*, the last term of *prev* + the first term of *curr*, and (2) our proposed connective-related features: lowercased *C* string and *C* category (the syntactic category of the connective: subordinating, coordinating, or discourse adverbial), where *curr* and *prev* indicate the current and previous clause respectively and the corresponding category for the connective *C* is obtained from the list provided in (Knott, 1996).

PS Arg2 Extractor: The PS Arg2 Extractor is similar to the PS Arg1 Extractor. However, they differ as follows: (1) in the first step, we consider the sentence containing connective *C* as the text span where Arg2 occurs and besides the previous nine punctuation symbols, we also use the connective *C* to split the sentence; (2) we adopt different features to build classifier: lowercased verbs in *curr*, lemmatized verbs in *curr*, the first term of *curr*, the last term of *curr*, the last term of *prev*, the first term of *next*, the last term of *prev* + the first term of *curr*, the last term of *curr* + the first term of *next*, production rules extracted from *curr*, *curr* position (i.e., the position of *curr* in the sentence: start, middle or end), *C* string, lowercased *C* string, *C* position, *C* category, path of C's parent → root, compressed path of *C*'s parent → root.

3.1.4 Explicit Sense Classifier

From previous components, we have identified all discourse connectives and their arguments from the texts. Here, we move to decide what Explicit relation each of them conveys.

The features for this classifier consist of the following: (1) Lin's features: *C* string, *C* POS, $prev_1$ + *C* (2) Pitler's features: *self-category*, *parent-category*, *left-sibling-category*, *right-sibling-category*, *C*-Syn interaction, Syn-Syn interaction. (3) our five newly proposed features: *parent-category linked context*, previous connective and its POS of *as* and previous connective and its POS of *when*. The first *parent-category linked context* feature is to provide more syntactic context information for the classification. The last four features are specially designed to disambiguate the relation senses of the connective *as* or *when*, since the two connectives often have ambiguity between Contingency.Cause.Reason and Temporal.Synchrony. As shown in Example 1, the previous connective of the discourse connective *as* is *But*, therefore the discourse connective *as* usually carries the Contingency.Cause.Reason sense rather than Temporal.Synchrony.

(1) *But the gains in Treasury bonds were pared* as **stocks staged a partial recovery.**
(Contingency.Cause.Reason – WSJ-1213)

3.2 Non-Explicit Parser

In this section, we discuss the identification of the Non-Explicit relations.

Since the Non-Explicit relations are only annotated for adjacent sentence pairs within paragraphs, we first collect all adjacent sentence pairs within each paragraph, but not identified in any Explicit relation. We assume the previous sentence as Arg1 and the next sentence as Arg2, and then identify the sense by the features extracted from (Arg1, Arg2). After that, we use Implicit Arg1 Extractor and Implicit Arg2 Extractor to label Arg1 and Arg2 for Non-EntRel relations in Non-Explicit, and for EntRel relations, we simply label the previous sentence as Arg1 and the next as Arg2.

Moreover, as shown in Figure 1, we use the Non-Explicit sense classifier again to identify the sense on the refined arguments (extracted arguments from Implicit Arg1&Arg2 Extractor) rather than the adjacent sentence pairs (i.e., previous sentence as Arg1, the next sentence as Arg2). Our expectation is that the overall parser performance might be improved if we extract features on refined argument spans rather than original argument spans.

3.2.1 Non-Explicit Sense Classifier

According to previous work, this component is the most difficult one in the discourse parser. And the features we adopted in this component are chosen from (Lin et al., 2009; Pitler et al., 2009; Rutherford and Xue, 2014), including: *production rules*, *dependency rules*, *first-last*, *first3*, *modality*, *verbs*,

Inquirer, *polarity*, *immediately preceding discourse connective of current sentence pair*, *Brown cluster pairs*. For the collection of *production rules*, *dependency rules*, and *Brown cluster pairs*, we used a frequency cutoff of 5 to remove infrequent features, and for Brown cluster, we choose 3,200 classes, as in (Rutherford and Xue, 2014).

3.2.2 Implicit Arg1 Extractor

The implicit Arg1 Extractor is performed to extract Arg1 for Non-EntRel relations in Non-Explicit, which is done similarly to the PS Arg1 Extractor. We first split the sentence into clauses and then decide each clause whether it belongs to Arg1 or not. The features extracted from the current and previous clauses (*curr* and *prev*) are: the first term of *curr*, the last term of *prev*, the cross product of the *prev* and *curr* production rules, the path of the first term of *curr* $\rightarrow$ the last term of *prev*, number of words of *curr*.

3.2.3 Implicit Arg2 Extractor

The implicit Arg2 Extractor is similar to that of Arg1, but different features are extracted from the current, previous, and next clauses (*curr*, *prev*, and *next*), including: lowercased verbs in *curr*, the first term of *curr*, the last term of *prev*, the last term of *prev* + the first term of *curr*, the last term of *curr* + the first term of *next*, *curr* position, the cross product of the *prev* and *curr* production rules, the cross product of the *curr* and *next* production rules, the path of the first term of *curr* $\rightarrow$ the last term of *prev*, number of words of *curr*.

4 Experiments on Training Data

To implement the 9 components described above, we compared two supervised machine learning algorithms, i.e., MaxEnt and Naive Bayes, implemented in MALLET toolkit[1]. For each component, we chose the algorithm with better performance. Specifically, we use Naive Bayes to build Non-Explicit Sense Classifier, and MaxEnt for the other 8 components.

We use PDTB Section 02-21 for training and Section 22 for development, which are provided by CoNLL-2015 with parse trees along with POS tags produced by the Berkeley Parser. And we participate in the closed tracks, that is, only two resources (i.e., Brown Clusters and MPQA Subjectivity Lexicon) are used in our discourse parser.

According to the requirement, a relation is considered to be correct if and only if: (1) the discourse connective is correctly detected (for explicit discourse relations); (2) the sense of a discourse relation is correctly predicted; (3) the text spans of the two arguments as well as their labels (Arg1 and Arg2) are correctly predicted. We use the official measure F_1 (harmonic mean of Precision and Recall) to evaluate performance.

4.1 Results of Explicit Parser

Table 1 reports the results of the explicit discourse parser on development data set of three components (i.e., Connective classifier, Arg1 position classifier and Explicit sense classifier) without error propagation (EP), where our new features are introduced. We find that the F_1 scores of all these classifiers are increased by adding our new features (+new).

Component	P&N and Lin			+ new		
	P	R	F_1 (%)	P	R	F_1 (%)
Connective Classifier	94.80	93.97	94.38	95.28	95.00	95.14
Arg1 Position Classifier	97.82	98.88	98.35	99.77	99.57	99.69
Explicit Sense Classifier	89.11	89.11	89.11	90.14	90.14	90.14

Table 1: Results for three components which add in our new features, no EP

To evaluate the performance of Explicit arguments extraction, we build the PS baseline by labeling the previous sentence of the connective as Arg1, and the text span between the connective and the beginning of the next sentence as Arg2. Table 2 summarizes the results of Explicit arguments extraction with exact matching and without error propagation, and the corresponding PS baseline is shown within parentheses. Note that we removed the leading or tailing punctuation from all text spans before evaluation. We see that the F_1 of PS is improved by a large margin for Arg1, Arg2 and Both by using two separate PS argument extractors, and the overall F_1 of Explicit arguments extraction is also increased by 2.51%.

4.2 Results of Non-Explicit Parser

Table 3 reports the results for Non-Explicit sense classification without error propagation, where we

[1] mallet.cs.umass.edu

	Arg1 F_1 (%)	Arg2 F_1 (%)	Both F_1 (%)
SS	70.56	88.54	64.72
PS	50.64(44.20)	75.10(66.09)	39.91(32.61)
All	64.15(61.93)	84.25(81.15)	56.61(54.10)

Table 2: Results for Explicit arguments extraction, where "All" indicates the arguments extraction for all the Explicit relations, and "Both" indicates Arg1 and Arg2 of a relation are both exactly matched, no EP

	P	R	F_1 (%)
EntRel sense	58.54	66.98	62.47
All Non-Explicit Senses	43.12	42.72	42.92

Table 3: Results for Non-Explicit sense classification, no EP

extract features on gold standard arguments of the Non-Explicit relations. The first row gives the result of the EntRel identification. Since we only extract arguments for Non-EntRel relations in Non-Explicit, the performance on EntRel identification is important, since it affects the performance of arguments extraction on Non-Explicit relations.

Table 4 reports the results for arguments extraction on Non-EntRel relations in Non-Explicit without error propagation, where the first row shows the result of the baseline system by labeling the previous sentence as Arg1 and the next sentence as Arg2, and the second row shows the result when using two Implicit extractors. As we expected, using two separate Implicit extractors achieves much better performance than the baseline. Table 5 reports the comparison results for the overall arguments extraction of parser with error propagation, where the first row indicates the performance when simply using the previous sentence as Arg1 and the next sentence as Arg2 for all Non-Explicit relations, and the second shows the results of using two Implicit argument extractors for Non-EntRel relations. We see that the performance of the arguments extraction increases, but not too much, due to the error propagation from the EntRel identification (P: 39.32%, R: 64.19%, F_1: 48.76%; EP).

Table 6 shows the overall results, where the first row is the overall performance of the parser when identify Non-Explicit sense on original arguments (i.e., adjacent sentence pairs), and the second row is the results on refined arguments. We find that the overall F_1 of the parser is improved 0.41% by extracting features on the refined arguments.

	Arg1 F_1	Arg2 F_1	Both F_1
w/o Impl extractors	61.80	69.92	48.56
with Impl extractors	70.02	77.42	55.85

Table 4: Results for using Implicit Arg1&Arg2 extractors on Non-EntRel relations in Non-Explicit, no EP

	Arg1 F_1	Arg2 F_1	Both F_1
w/o Impl extractors	66.06	77.06	56.31
with Impl extractors	66.97	77.21	57.21

Table 5: Results for overall argument extraction of the parser, EP

	P	R	F_1 (%)
on original arguments	37.18	37.67	37.43
on refined arguments	37.59	38.09	37.84

Table 6: Results of overall parser performance using Non-Explicit sense classifier on original and refined arguments

5 Results on Test Data Sets

The above described discourse parser system is evaluated on two test datasets provided by the shared task: (1) Section 23 in PDTB; (2) blind test set drawn from a similar source and domain in terms of F_1. The officially released results are shown in Table 7. Our parser ranks the first on both test datasets. Although the two test datasets are both from news wire domain and in PDTB style, there are difference between the two datasets. For example, not all discourse connectives in blind test dataset are listed in PDTB, e.g., "upon" is annotated as discourse connective in blind test dataset while it is not in PDTB.

We compare our discourse parser with Lin's on PDTB Section 23. We find that new features proposed in this work do help increase F_1 of Explicit connective classification by 0.54%. And for the Explicit arguments extraction, our parser achieves better performance as well. However, since the sense labels of Explicit and Non-Explicit relations in CoNLL-2015 differ from Lin's, i.e., Lin used partial sense labels of the second level (Type) by excluding several small categories while CoNLL-2015

	on PDTB Section 23						on blind test data set					
	our parser			Lin's parser			our parser			2nd rank parser		
	P	R	F_1 (%)	P	R	F_1 (%)	P	R	F_1 (%)	P	R	F_1 (%)
Explicit connective	94.83	93.49	**94.16**	-	-	93.62	93.48	90.29	**91.86**	92.57	87.41	89.92
Explicit Arg1 extraction	51.05	50.33	**50.68**	-	-	47.68	49.16	47.48	48.31	50.48	47.66	49.03
Explicit Arg2 extraction	77.89	76.79	**77.33**	-	-	70.27	75.61	73.02	**74.29**	72.76	68.71	70.68
Explicit Both extraction	45.54	44.90	**45.22**	-	-	40.37	42.09	40.65	**41.35**	40.76	38.49	39.59
Explicit only sense	35.52	34.69	34.93	-	-	-	29.69	26.24	**25.91**	33.15	24.81	25.22
Non-Explicit Arg1 extraction	64.83	69.50	67.08	-	-	-	58.66	63.25	**60.87**	39.47	47.93	43.29
Non-Explicit Arg2 extraction	66.02	70.78	68.32	-	-	-	71.88	77.49	**74.58**	51.58	62.63	56.57
Non-Explicit Both extraction	51.20	54.89	52.98	-	-	-	48.58	52.37	**50.41**	34.93	42.42	38.31
Non-Explicit only sense	53.18	10.45	9.06	-	-	-	44.74	8.64	**7.69**	37.08	7.83	6.81
All Arg1 extraction	59.20	61.03	60.10	-	-	-	55.12	56.58	**55.84**	44.61	48.64	46.54
All Arg2 extraction	71.43	73.64	72.52	-	-	-	73.49	75.43	**74.45**	60.02	65.43	62.60
All Both extraction	48.62	50.13	49.36	-	-	-	45.77	46.98	**46.37**	37.25	40.61	38.86
Sense (Expicit+Non-Explicit)	31.44	30.42	29.83	-	-	-	25.07	22.13	**21.82**	25.00	19.60	18.87
Overall Parser	29.27	30.08	29.72	-	-	20.64	23.69	24.32	**24.00**	20.94	22.83	21.84

Table 7: Results of our parser on PDTB Section 23 and the blind test dataset, Lin's parser on PDTB Section 23 and the 2nd rank parser on blind test dataset, "All" indicates all relations (Explicit and Non-Explicit relations), "-" indicates not available

used different sense labels (partial of the three sense levels with excluding and/or merging several small categories), the direct comparison on sense classification as well as the parser performance is not possible.

Table 7 also shows the results of our parser and the 2nd rank parser on blind test dataset, we see that our parser achieves better performance, especially on the arguments extraction.

6 Conclusion

In this work, we have implemented a refined discourse parser by adding new components and features based on Lin's system. Specifically, we (1) build two PS arguments extractors (i.e., PS Arg1 Extractor and PS Arg2 Extractor) to improve performance of Explicit arguments extraction, (2) propose new features for building three classifiers (i.e, Connective Classifier, Arg1 Position Classifier, Explicit Sense Classifier), (3) construct two Implicit arguments extractors (i.e., Implicit Arg1 Extractor and Implicit Arg2 Extractor) for Non-EntRel relations, and (4) perform Non-Explicit sense classification on refined arguments. Our system ranks the first on both test data sets, i.e. PDTB Section 23 and a blind test dataset.

Acknowledgements

This research is supported by grants from Science and Technology Commission of Shanghai Municipality under research grant no. (14DZ2260800 and 15ZR1410700) and Shanghai Collaborative Innovation Center of Trustworthy Software for Internet of Things (ZF1213).

References

Sucheta Ghosh, Richard Johansson, and Sara Tonelli. 2011. Shallow discourse parsing with conditional random fields. In *In Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP 2011*. Citeseer.

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

Fang Kong, Hwee Tou Ng, and Guodong Zhou. 2014. A constituent-based approach to argument labeling with joint inference in discourse parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 68–77, Doha, Qatar, October. Association for Computational Linguistics.

Man Lan, Yu Xu, Zheng-Yu Niu, et al. 2013. Leveraging synthetic discourse data via multi-task learning for implicit discourse relation recognition. In *ACL (1)*, pages 476–485. Citeseer.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the Penn Discourse Treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 343–351. 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*, pages 1–34.

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 rela-

tions 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.

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.

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

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*, Beijing, China.