IIT (BHU) Submission for the CoNLL-2016 Shared Task: Shallow Discourse Parsing using Semantic Lexicon

Manpreet Kaur, Nishu Kumari, Anil Kumar Singh, Rajeev Sangal
Department of Computer Science, IIT-BHU,
Varanasi, Uttar Pradesh, India - 221005
f2012687@goa.bits-pilani.ac.in, {kumari.nishu.apm11,
aksingh.cse}@iitbhu.ac.in, sangal@iiit.ac.in

Abstract

This paper describes the Shallow Discourse Parser (SDP) submitted as a part of the Shared Task of CoNLL, 2016. The discourse parser takes newswire text as input and outputs relations between various components of the text. Our system is a pipeline of various sub-tasks which have been elaborated in the paper. We choose a data driven approach for each task and put a special focus on utilizing the resources allowed by the organizers for creating novel features. We also give details of various experiments with the dataset and the lexicon provided for the task.

1 Introduction

Shallow Discourse Parsing (SDP) is a linguistic task that identifies semantic relations between a pair of lexical units in a piece of discourse. Discourse relation is defined by three entities: a connective, a pair of lexical units between which the relation exists and the type or sense of relation between them (Xue et al., 2016). The discourse relations can be explicit, in which relations are expressed by certain words or phrases, or implicit, where words are not directly used to convey the relation, but instead, the meaning is implied. These words or phrases which convey the existence of a discourse relation directly are called connectives. The lexical units between which relation exists, could be a pair of clauses, a pair of sentences or even multiple sentences which can be adjacent or non-adjacent. These are called arguments.

A discourse treebank called the Penn Discourse TreeBank or PDTB (Prasad et al., 2008) serves as the gold standard for this task and is used as training data. The output of our system follows the same format as PDTB. Development data is also provided to perform experiments on the system. Phrase structure and dependency parses of both the training and development data have also been provided to assist in the task. Further details of the Shared Task can be found in the overview paper (Xue et al., 2016). Final evaluation of the parser is on test and blind data sets through TIRA platform set up by (Potthast et al., 2014). Besides automating the submission and evaluation system, TIRA also has provision for plagiarism detection, author identification and author profiling.

The SDP task can be broadly classified into two categories of explicit and non-explicit relation detection. We discuss the pipeline for explicit parser in section 2 and non-explicit parser in Section 3. Various results and experiments carried out are reported in the relevant sub-sections. These results are based on individual stages without error propagation from previous stages, unless specified otherwise. We report results on test and blind datasets and conclude our work in Section 4 and 5 respectively.

2 Explicit SDP

Identification of explicit discourse relations consists of several stages. First stage is the detection of discourse connectives in the text. This connective binds the arguments syntactically and semantically (Prasad et al., 2008) which is helpful in feature creation for the following tasks of argument position detection and argument span extraction. Once the arguments of the relation are extracted, we perform sense classification of the relation.

2.1 Connective Detection

This is the first stage of the parser which detects the existence of discourse connectives in the text. The input to this stage is raw text and we analyze the entire text for the presence of connectives which could form a discourse relation. Around 100 connective spans have been identified upon extensive research by the team that annotated PDTB (Prasad et al., 2008). However, the occur-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 108–114,
Berlin, Germany, August 7-12, 2016.

rence of these words does not guarantee that it will form a discourse connective as can be seen in the following example:

My Father once worked on that project.

- 'once' is a non-discourse connective

You cannot change your statement once it comes out of your mouth.

-'once' is a discourse connective

Here the connective 'once' acts as a discourse connective based on the context. A string matching script is not sufficient for this task and we therefore use Maximum Entropy Classifier to identify whether a potential connective keyword actually forms a discourse relation or not. This task has been sufficiently mastered and high F1 scores have been reported by previous teams. Mostly syntactic features have been used for this classification task such as Connective, connectivePOS, PrevWord, PrevPOSTag, PrevPOS + connectivePOS, nextWord, nextPOSTag, nextPOS + connectivePOS, root2Leaf, root2LeafCompressed, leftSibling, rightSibling, parentCategory. These features have been borrowed from previous work of (Wang and Lan, 2015).

2.2 Argument Labeler

After identifying the discourse connective span present in the input text, we need to locate the relative position of the arguments w.r.t. the sentence containing the connective. Arg2 is taken as the argument which occurs in the same sentence as the connective and is therefore syntactically associated with it (Prasad et al., 2008). Hence, we identify the position of Arg1 relative to Arg2 and the connective. The Argument Labeling task can be divided into the following sub-tasks:

- Identifying the relative position of Arg1 w.r.t. Arg2 (and the connective)
- Extracting clauses which are potential argument spans
- Classifying the candidate clauses into Arg1, Arg2 or Null

2.2.1 Argument Position Classifier

We need to identify whether arguments are located in the same sentence (the SS case) or in a sentence before the connective (the PS case). We ignore the following sentence (FS) case and the non-adjacent PS case since these types have a small percentage of instances.

Features used for Argument Position Classifier are connectiveString, connecivePOS, connecivePosition, prevWord, prevWord+connecive, prevPOS, prevPOS+connecivePOS, prev2Word, prev2Word+connecive, prev2POStag, prev2POS+connecivePOS. The feature names are self-explanatory. Connective string itself is a very good feature for this stage. For instance, when the connective token is 'And' (with first letter capitalized), there is a continuation of an idea from previous sentence and thus Arg1 is likely to be in PS. Whereas, when the first letter of connective is in lowercase such as 'and', Arg1 is very likely to be the clause on the left-hand side of 'and', making Arg1 in SS as connective. Connective position, which takes the values 'start', 'middle' and 'end' is also a very useful feature. This argument position classifier is trained using Maximum Entropy Classifier.

2.2.2 Argument Span Extractor

This stage of the pipeline extracts the span of the arguments from the sentence or sentences containing the discourse relation. To extract arguments, we first break the sentence into clauses. Two methods have been proposed in literature to carry out this task: Lin's tree subtract method (Lin et al., 2014) and Kong's constituency based method (Kong et al., 2014). According to (Kong et al., 2014), Kong's constituency based approach outperforms Lin's tree subtraction algorithm. However, since Kong's method is based on using the connective node in the parse tree as the base node for recursion, we can only use this method for those sentences which contain the connective. Hence, we use Kong's extraction method for Same Sentence Argument Extraction.

SS Argument Extractor: Kong's constituency-based approach is a recursion in which the connective's lowest tree node is chosen as the target node, and its left and right siblings are chosen as candidates for arguments. The target node is updated to the current target node's parent and the process is repeated. There is a slight modification in the algorithm for multi-word connectives. Similar to Kong et al's approach for multi-word connectives, we choose the immediate left siblings of the first word in the connective and immediate right siblings of the last word of the connective as candidate arguments

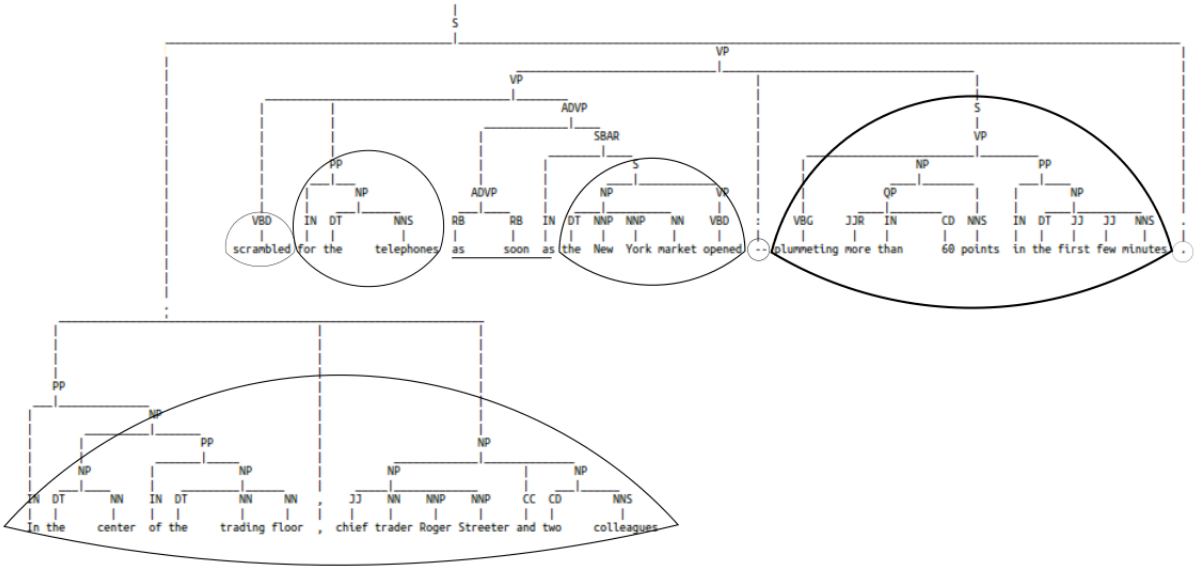

Figure 1: Parse tree showing candidate argument nodes using Kong's Extraction algorithm for a multi-word connective

in addition to taking left and right siblings of the lowest node that covers the entire connective. This modification of algorithm for multi-word cases is important as the modified algorithm extracts more refined constituents from the sentence. In the following example, the updated algorithm extracts *'the New York market opened'* as a constituent, whereas the algorithm without multi-word case would not have extracted it at all.

Consider the following example with its gold-standard parse tree as shown in Figure 1:

(1) *In the center of the trading floor, chief trader Roger Streeter and two colleagues scrambled for the telephones* **as soon as** *the new York market opened – plummeting more than 60 points in the first few minutes.*

Argument candidates detected are: *'In the center of the trading floor'* , ',' , *'chief trader Roger Streeter and two colleagues'* , *'scrambled'* , *'for the telephones'* , *'the New York market opened'* , '–' , *'plummeting more than 60 points in the first few minutes'* , '.'

The final extracted arguments are:

Arg1 - 'In the center of the trading floor, chief trader Roger Streeter and two colleagues scrambled for the telephones'

Arg2 - 'the New York market opened'

Table 1 compares the results of Kong's Extractor with and without incorporating multi-word scenario. As expected, the F1 score of Arg2 with multi-word case is better by about 2.7%.

	Arg1 F1 score	Arg2 F1 score
Kong's pruning algo (w/o multi-word case)	50.11	70.02
Kong's pruning algo (w/ multi-word case)	50.11	72.71

Table 1: Argument Extraction experiments on development data

PS Argument Extractor: In this case, we take the entire previous sentence as Arg1. Arg2 is taken as the sentence containing the connective after subtracting connective tokens from the sentence. For this task, we can also use Lin's clause tree subtraction method (2014) to extract Arg1 and Kong's constituency based approach (2014) to extract Arg2 for better performance.

2.2.3 Argument Classification

Features used for classification of extracted phrases into arguments have been borrowed from previous works of Kong (Kong et al., 2014) and Wang (Wang and Lan, 2015). These features are used to classify each candidate into one of the three categories: 'Arg1', 'Arg2' or 'Null'. Both connective and constituent-candidate based features are used: `Connective String`, `POS tag` of the connective, `leftSiblingNo` is

			Connective	Arg1	Arg2	Arg1+Arg2	Sense	Overall
Explicit	**Our Parser**	**Dev**	93.39	51.35	65.02	39.79	34.94	34.94
		Test	93.01	43.93	58.47	34.10	29.74	29.74
		Blind	89.03	40.37	60.09	29.49	23.43	23.43
	Wang's Parser	**Test**	94.16	50.68	77.33	45.22	34.93	-
		Blind	91.86	48.31	74.29	41.35	25.91	-
Implicit	**Our Parser**	**Dev**	-	41.22	40.96	32.25	4.59	4.59
		Test	-	38.59	36.44	27.42	3.44	3.44
		Blind	-	37.03	41.49	26.24	8.78	8.78
	Wang's parser	**Test**	-	67.08	68.32	52.98	9.06	-
		Blind	-	60.87	74.58	50.41	7.69	-
Overall	**Our parser**	**Dev**	93.39	45.52	49.63	34.97	15.41	15.41
		Test	93.01	41.34	44.36	29.82	12.82	12.82
		Blind	89.03	38.77	48.19	27.41	12.41	12.41
	Wang's parser	**Test**	94.16	60.10	72.52	49.36	29.83	29.72
		Blind	91.86	55.84	74.45	46.37	21.82	24.00

Table 2: System performance and comparison on development, test and blind datasets

	F1 score	Accuracy
Naive Bayes	82.27	88.2
Maxent	75.6	86.9

Table 3: Explicit sense classification: feature experimentation with NB and MaxEnt on development data

the number of left siblings of the connective, `rightSiblingNo` is the number of right siblings of the connective, `ConnCat` is the syntactic category of the connective which takes the values 'subordinating', 'coordinating' or 'discourse adverbial', `clauseRelPos` is position of the constituent candidate relative to the connective which takes the values 'left' , 'right' or 'previous', `clausePOS` is the POS tag of the constituent candidate, `clauseContext` is the context of the constituent, i.e., POS combination of the constituent, its parent, left sibling and right sibling (when there is no parent or sibling, it is marked as NULL), `conn2clausePath` is the path from connective node to the node of the constituent.

Once, the classifier tags the clauses with its labels, Arg1 and Arg2 are obtained by stitching the strings of ordered arg1 clauses and arg2 clauses respectively.

2.3 Explicit Sense Classification

After determining the spans of Arg1 and Arg2, we feed these arguments into the next stage of the pipeline which detects the sense of the explicit discourse relation. The connective string itself is a good indicator of the sense of the relation due to lexical mapping between them. However, there are cases of ambiguity, in which a connective word is used to describe multiple senses. For this reason, the task requires machine learning in which the classifier uses other syntactic features to determine the sense of the relation.

The features we used for training are connectiveString, connectiveHead, connectivePOS, connectivePrev, connectivePosition, connectiveCategory, subjectivityStrengthArg1, subjectivityStrengthArg2, verbNetClassArg2 and verbNetClassArg2. Subjectivity Strength and VerbNet class features are created from the semantic lexicons provided for the shared task and are described in the next section.

From the results in Table 3, we note that Naive Bayes performs better than MaxEnt classifier. We conjecture the cause for this is that MaxEnt classifier tends to overfit the data. Hence, Naive Bayes is chosen to perform sense classification.

3 Non-Explicit SDP

There are three types of Non-Explicit relations: Implicit, EntRel and AltLex. The remaining pair of sentences which did not contain explicit connectives are fed into this stage of the pipeline. Our system now treats all the remaining adjacent sentences as Implicit relations. This hard coding cost us a high performance dip as EntRel relations con-

stitute about a third of the non-explicit data (215 EntRel relations and 522 Implicit relations in development data) and not all remaining sentences contain an implicit relation.

In our implicit argument span detector, we treat the first sentence in adjacent sentence pair as Arg1 and the second sentence as Arg2. Next, we focus on the Implicit Sense Classification task.

3.1 Implicit Sense Classification

This task is considered as the bottleneck of SDP systems and is especially challenging due to lack to connective based features. We create a different set of baseline features as borrowed from (Lin et al., 2014). We describe semantic features used for this task in detail in this section.

3.1.1 Baseline features

Baseline features chosen for this task are syntactic features created from the dependency and constituency parses of the two arguments. First, both dependency and constituency parses from the entire training corpus were extracted. This created around 12,489 constituency parses and 89 dependency parses. However, the number of parse features was too high and unnecessary to work with. Hence, we put a frequency cap of 5 on the feature set which brought the features down to 2,515. We also used NLTK's stop-words to filter out dependency parses created by common words as these parse rules are highly recurring over the distribution of the entire corpus.

3.1.2 Semantic Features

Sense detection is essentially a semantic task since we are trying to determine the "meaning" of the relation. For this reason, we have experimented with semantic tools like MPQA Subjectivity, VerbNet classes and Word2Vec. Each of these tools and lexicons have been provided for the closed track of Shallow Discourse Parsing.

MPQA Subjectivity: The semantic feature created using MPQA subjectivity lexicon measures the negativity and positivity strength of the arguments. For calculating the subjectivity strength of the arguments, subjectivity annotation for each word of the argument is taken. If the word has negative and strong polarity, it is assigned -2, for negative and weak polarity it is assigned -1, for strong positive polarity +2 and for weak positive polarity +1 respectively. The subjectivity strength of all words in the argument is summed up. If the sum is 0 then it is neutral, otherwise it is positive or negative.

VerbNet Classes: VerbNet is a verb lexicon with mappings to WordNet and FrameNet. VerbNet is organized into classes (with subclasses) on the basis of syntactic and semantic similarity (Kipper et al., 2006). We have created `verbNetClassArg1` and `verbNetClassArg2` features, which contain the VerbNet class of the lemmatized forms of the main verbs of the respective arguments (Zhou, 2015). VerbNet classes are important features and this was verified by analyzing the most informative features of this classification task. We find that many VerbNet classes are more informative than even baseline features.

Word2Vec: Subjectivity strength and VerbNet classes only capture information about specific albeit important words of a sentence. To capture the context of the entire argument and interaction between the arguments, we use the Word2Vec tool. Word2Vec is a deep learning tool that outputs vector representation of an input word in a large-dimensional vector space. We have used Google's Word2Vec model trained on a part of Google News dataset of 100 billion words. This Word2Vec model contains 300 dimensional vectors for 3 million words and phrases.

Words with similar meaning are expected to have vectors in close proximity in the vector space (Mikolov et al., 2013). Inspired from (Yih et al., 2013), a work on Question-Answering system, we represent the entire Arg1 and Arg2 as vectors. We take each argument, drop the stopwords and then take a weighted sum over the vector representations of remaining words of the argument. Even after removing stop words, there is a difference in importance and relevance of the remaining words. This is why we choose to take a weighted sum of the word vectors. We chose `TF-IDF` (Term Frequency-Inverse Document Frequency) scores as weights. The TF-IDF value increases proportionally with the number of times a word appears in the document and decreases with the frequency of the word in the corpus. This balances the weights of words which occur more frequently in literature.

We created three features using Word2Vec tool: `Arg1Cluster`, `Arg2Cluster` and `cosineDistance`. We perform PCA (Principle-Component Analysis) over the Arg

vectors to reduce the vector dimensions from 300 to 3 as the depth of sense classes is also three or less. By intuition, we only require three dimensions to represent the three levels of sense classes. We perform K-Means clustering over Arg vectors of the training data and assign clusters to Arg1 and Arg2 of development, test and blind data as `Arg1Cluster` and `Arg2Cluster`. We used sklearn's TfIdfVectorizer to compute the `TF-IDF` scores and sklearn's PCA and K-Means to perform clustering over the vectors.

The `cosineDistance` feature is a dot product of Arg1 and Arg2 vectors. We hope to capture the similarity or closeness of the two arguments using this numerical value. Following is the formula used for calculating cosine distance:

$$d = \sum_{k=1}^{n} \left(\left(\sum_{w_i \in Arg1} tfidf(w_i) * word2vec(w_i) \right)_k * \left(\sum_{w_j \in Arg2} tfidf(w_j) * word2vec(w_j) \right)_k \right) \quad (1)$$

Here, d is the cosineDistance and n is 300, the dimension of the vector space of GoogleNews-vectors-negative300.bin, the word2vec model trained on Google News dataset.

3.1.3 Experimentation

We used a combination of the features described above to gauge their performance on sense classification task. VerbNet and Subjectivity features are known to perform well according to previous literature. Hence, we test the novel Word2Vec features on top of baseline features, Subjectivity strength and VerbNet classes. For this reason, we call the combination of baseline features, Subjectivity strength and VerbNet classes as baseline in Table 4.

The results reported the Table 4 are on development dataset. As expected, Word2Vec features improve the F1 score by about 2.3%. Thus, we use a combination of all the baseline features, Subjectivity features, VerbNet features and Word2Vec features in the Implicit Sense classification task. Also, the number of parse features is very high (2515), making total number of features equal to 2522. Therefore, we use NLTK's Naive Bayes Classifier over Maximum Entropy as NLTK's implementation of Maximum Entropy is not able to handle the vast number of features.

	F1 score	**Accuracy**
baseline	11.17	25.47
baseline+cosineDist	12.00	24.52
baseline+cosineDist +argclusters	13.44	24.52

Table 4: Implicit sense feature experimentation on development dataset

4 Results

Table 2 contains the results of our updated system on development, test and blind datasets. In the updated system, we fixed a small bug in argument index alignment code which doubled our overall parser F1 score on the development data. Hence, we report the updated results in the paper. We also used Word2Vec features in our updated system. The Word2Vec features did not improve the F1 score of Implicit Sense Classification on development and test datasets. This is probably because of error propagation from previous stages. Surprisingly, the updated Implicit Classifier performs better on blind dataset as compared to development and test dataset.

There are several weak links in our pipeline. For instance, the PS-Explicit and Implicit argument extractors are naive and hard-coded. This is one major cause of low F1 scores as compared to Wang et al. We feel that by fixing these links, we can improve the result by a significant margin.

5 Conclusion

In this work, we have implemented a discourse parser trained on PDTB corpus with a special focus on using semantic lexicons. We have described the system architecture and various experimentation results in the paper. Our contribution to the SDP system is the introduction of novel features to the bottleneck of SDP systems, i.e., the Implicit Sense Classification task. Specifically, we have created `Arg1Cluster`, `Arg2Cluster` and `cosineDistance` features using Word2Vec tool for Implicit Sense Classification task, which improved F1 score of the task by about 2.3%. The task of Shallow Discourse Parsing will give more promising results by making use of other lexical and semantic tools, thus encouraging further research to obtain better results.

References

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2006. Extending verbnet with novel verb classes. In *Proceedings of LREC*. Citeseer.

Fang Kong, Hwee Tou Ng, and Guodong Zhou. 2014. A constituent-based approach to argument labeling with joint inference in discourse parsing. In *EMNLP*, pages 68–77.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A pdtb-styled end-to-end discourse parser. volume 20, pages 151–184. Cambridge Univ Press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

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, Berlin Heidelberg New York, September. Springer.

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.

Jianxiang Wang and Man Lan. 2015. A refined end-to-end discourse parser. In *The Nineteenth Conference on Computational Natural Language Learning*, pages 17–24, Beijing, China, July.

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.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. Citeseer.

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