The CLaC Discourse Parser at CoNLL-2015

Majid Laali **Elnaz Davoodi** **Leila Kosseim**
Department of Computer Science and Software Engineering,
Concordia University, Montreal, Quebec, Canada
{m_laali, e_davoo, kosseim}@encs.concordia.ca

Abstract

This paper describes our submission (*kosseim15*) to the CoNLL-2015 shared task on shallow discourse parsing. We used the UIMA framework to develop our parser and used ClearTK to add machine learning functionality to the UIMA framework. Overall, our parser achieves a result of 17.3 F_1 on the identification of discourse relations on the blind CoNLL-2015 test set, ranking in sixth place.

1 Introduction

Today, discourse parsers typically consist of several independent components that address the following problems:

1. *Discourse Connective Classification*: The concern of this problem is the identification of discourse usage of discourse connectives within a text.

2. *Argument Labeling*: This problem focuses on labeling the text spans of the two discourse arguments, namely ARG1 and ARG2.

3. *Explicit Sense Classification*: This problem can be reduced to the sense disambiguation of the discourse connective in an explicit discourse relation.

4. *Non-Explicit Sense Classification*: The target of this problem is the identification of implicit discourse relations between two consecutive sentences.

To illustrate these tasks, consider Example (1):

(1) *We would stop index arbitrage* when **the market is under stress**.[1]

The task of *Discourse Connective Classification* is to determine if the marker *"when"* is used to mark a discourse relation or not. *Argument Labeling* should segment the two arguments ARG1 and ARG2 (in this example, ARG1 *is italicized* while **ARG2 is bolded**). Finally, *Explicit Sense Classification* should identify which discourse relation is signaled by *"when"* - in this case CONTINGENCY.CONDITION.

In this paper, we report on the development and results of our discourse parser for the CoNLL 2015 shared task. Our parser, named *CLaC Discourse Parser*, was built from scratch and took about 3 person-month to code. The focus of the CLaC Discourse Parser is the treatment of explicit discourse relations (i.e. problem 1 to 3 above).

2 Architecture of the CLaC Discourse Parser

We developed our parser based on the UIMA framework (Ferrucci and Lally, 2004) and we used ClearTK (Bethard et al., 2014) to add machine learning functionality to the UIMA framework. The parser was written in Java and its source code is distributed under the BSD license[2].

Figure 1 shows the architecture of the CLaC Discourse Parser. Motivated by Lin et al. (2014), the architecture of the CLaC Discourse Parser is a pipeline that consists in five components: *CoNLL Syntax Reader*, *Discourse Connective Annotator*, *Argument Labeler*, *Discourse Sense Annotator* and *CoNLL JSON Exporter*. Due to lack of time, we did not implement a *Non-Explicit Classification* in our pipeline and only focused on explicit discourse relations.

The *CoNLL Syntax Reader* and the *CoNLL JSON Exporter* were added to the CLaC Discourse Parser in order for the input and the output of the parser to be compatible with the CoNLL

[1] The example is taken from the CoNLL 2015 trial dataset.

[2] All the source codes can be downloaded from https://github.com/mjlaali/CLaCDiscourseParser.git

Figure 1: Components of the CLaC Discourse Parser

2015 Shared Task specifications. The *CoNLL Syntax Reader* parses syntactic information (i.e. POS tags, constituent parse trees and dependency parses) provided by CoNLL organizers and adds this syntactic information to the documents in the UIMA framework. To create a stand-alone parser, the *CoNLL Syntax Reader* can be easily replaced with the `cleartk-berkeleyparser` component in the CLaC discourse Parser pipeline. This component is a wrapper around the Berkeley syntactic parser (Petrov and Klein, 2007) and distributed with ClearTK. The Berkeley syntactic parser was actually used in the CoNLL shared task to parse texts and generate the syntactic information.

The *CoNLL JSON Exporter* reads the output discourse relations annotated in the UIMA documents and generates a JSON file in the format required for the CoNLL shared task. We will discuss the other components in details in the next sections.

2.1 Discourse Connective Annotator

To annotate discourse connectives, the *Discourse Connective Annotator* first searches the input texts for terms that match a pre-defined list of discourse connectives. This list of discourse connectives was built solely from the CoNLL training dataset of around 30K explicit discourse relations and contains 100 discourse connectives. Each match of discourse connective is then checked to see if it occurs in discourse usage or not.

Inspired by (Pitler et al., 2009), we built a binary classifier with six local syntactic and lexicalized features of discourse connectives to classify discourse connectives as discourse usage or non-discourse usage. These features are listed in Table 1 in the row labeled *Connective Features*.

2.2 Argument Labeler

When ARG1 and ARG2 appear in the same sentence, we can exploit the syntactic tree to label boundaries of the discourse arguments. Motivated by (Lin et al., 2014), we first classify each constituent in the parse tree into to three categories: part of ARG1, part of ARG2 or NON (i.e. is not part of any discourse argument). Then, all constituents which were tagged as part of ARG1 or as part of ARG2 are merged to obtain the actual boundaries of ARG1 and ARG2.

Previous studies have shown that learning an argument labeler classifier when all syntactic constituents are considered suffers from many instances being labeled as NON (Kong et al., 2014). In order to avoid this, we used the approach proposed by Kong et al. (2014) to prune constituents with a NON label. This approach uses only the nodes in the path from the discourse connective (or *SelfCat* see Table 1) to the root of the sentence (*Connective-Root path nodes*) to limit the number of the candidate constituents. More formally, only constituents that are directly connected to one of the *Connective-Root path nodes* are considered for the classification.

For example, consider the parse tree of Example (1) shown in Figure 2. The path from the discourse connective *"when"* to the root of the sentence contains these nodes: {WRB, WHADVP, SBAR, VP_2, VP_1, S_1}. Therefore, we only consider {S_2, NP_2, VB, MD, NP_1} for obtaining discourse arguments.

If the classifier does not classify any constituent as a part of ARG1, we assume that the ARG1 is not in the same sentence as ARG2. In such a scenario, we consider the whole text of the previous sentence as ARG1.

In the current implementation, we made the assumption that discourse connectives cannot be multiword expressions. Therefore, the Argument Labeler cannot identify the arguments of parallel discourse connectives (e.g. *either..or*, *on one hand..on the other hand*, etc.)

We used a sub-set of 9 features proposed by Kong et al. (2014) for the Argument Labeler classifier. The complete list of features is listed in Table 1.

2.3 Discourse Sense Annotator

Although some discourse connectives can signal different discourse relations, the naïve approach that labels each discourse connective with its most

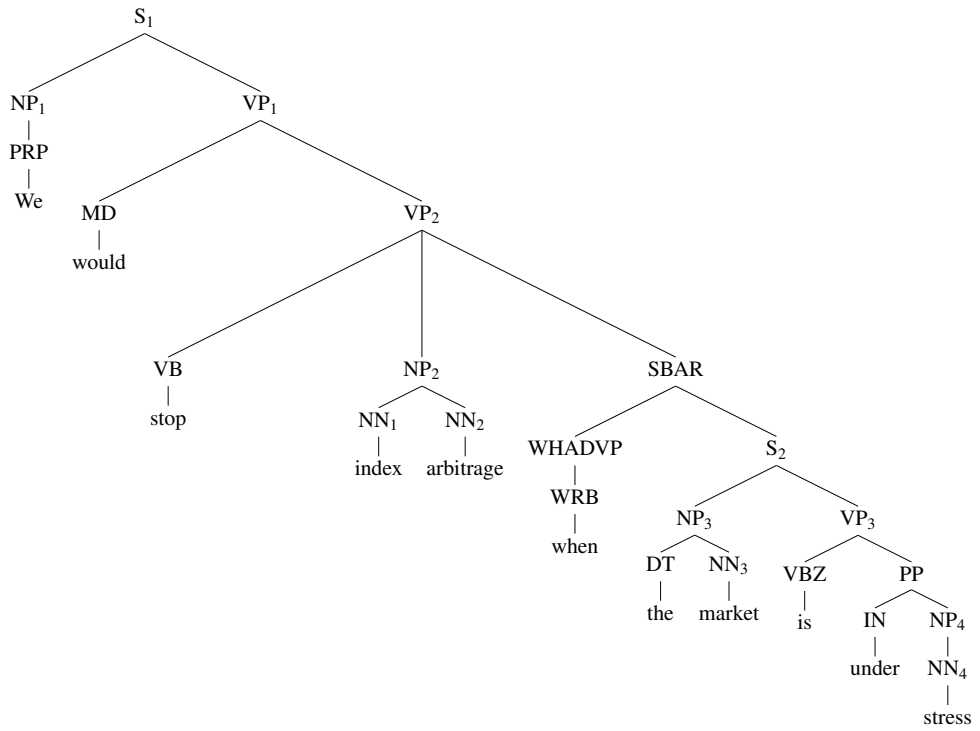

Figure 2: The Parse Tree Provided by CoNLL 2015 for Example (1)

Category	Description	Example
Connective Features	1. The discourse connective text in lowercase.	*when*
	2. The categorization of the case of the connective: *all lowercase*, *all uppercase* and *initial uppercase*	all lowercase
	3. The highest node in the parse tree that covers the connective words but nothing more	WRB
	4. The parent of *SelfCat*	WHADVP
	5. The left sibling of *SelfCat*	null
	6. The right sibling of *SelfCat*	S
Syntactic Node Features	7. The path from the node to the *SelfCat* node in the parser tree	$S \uparrow SBAR \downarrow WHADVP$
	8. The context of the node in the parse tree. The context of a node is defined by its label the label of its parent, the label of left and right sibling in the parse tree.	S-SBAR-WHADVP-null
	9. The position of the node relative to the *SelfCat* node: *left* or *right*	left

Table 1: Features Used in the CLaC Discourse Parser

frequent relation performs rather well. According to Pitler et al. (2009), such an approach can achieve an accuracy of 85.86%. Due to lack of time, we implemented this naïve approach for the Discourse Sense Annotator, using the 100 connectives mined from the dataset (see Section 2.1) and their most frequent relation as mined from the CoNLL training dataset.

3 Experiments and Results

As explained in Section 2, the CLaC Discourse Parser contains two main classifiers, one for the *Discourse Connective Annotator* and one for the *Argument Labeler*. We used the off-the-shelf implementation of the C4.5 decision tree classifier (Quinlan, 1993) available in WEKA (Hall et al., 2009) for the two classifiers and trained them us-

	Discourse Connective Classifier	Argument Labeler	Discourse Parsing (explicit only)	Discourse Parsing (explicit and implicit)
Best Result	91.86%	41.35%	30.58%	24.00%
CLaC Parser	**90.19%**	**36.60%**	**27.32%**	**17.38%**
Average	74.20%	23.89%	18.28%	13.25%
Standard deviation	23.24%	13.01%	9.93%	6.41%

Table 2: Summary of the Results of the CLaC Discourse Parser in the CoNLL 2015 Shared Task.

ing the CoNLL training dataset.

Although the CLaC discourse parser only considers explicit discourse relations (which only accounts for about half of the relations), the parser ranked 6th among the 17 submitted discourse parsers. The overall F_1 score of the parser and the individual performance of the *Discourse Connective Classifier* and the *Argument Labeler* in the blind CoNLL test data are shown in Table 2. As Table 2 shows, the performance of the parser is consistently above the average. In addition, the performance of the *Discourse Connective Classifier* is very close to the best result.

Note that all numbers presented in Table 2 were obtained when errors propagate through the pipeline. That is to say, if a discourse connective is not correctly identified by the *Discourse Connective Classifier* for example, the arguments of this discourse connective will not be identified. Thus, the recall of the *Argument Labeler* will be affected.

The CoNLL 2015 results of the submitted parsers show that the identification of ARG1 is more difficult than ARG2. In line with this, the CLaC Discourse Parser performed better on the identification of ARG2 (with the F_1 score of 69.18%) than ARG1 (with the F_1 score of 45.18%). Table 3 provides a summary of the results for the identification of Arg1 and Arg2. An important source of errors in the identification of ARG1 is that *attribute spans* are contained within ARG1. For example in (2), the CLaC Discourse Parser incorrectly includes the text "*But the RTC also requires "working" capital*" within ARG1.

	Arg1	Arg2
Best Result	49.68%	74.29%
CLaC Parser	**45.18%**	**69.18%**
Average	30.77%	50.91%
Std. deviation	15.31%	20.58%

Table 3: Results of the Identification of ARG1 and ARG2.

(2) But the RTC also requires "working" capital *to maintain the bad assets of thrifts that are sold* until **the assets can be sold separately**.[3]

With regards to the identification of ARG2, we observed that subordinate and coordinate clauses are an important source of errors. For example in (3), the subordinate clause "*before we can move forward*" is erroneously included in the ARG2 span when the CLaC Discourse Parser parses the text. The cause of such errors are usually rooted in an incorrect syntax parse tree that was fed to the parser. For instance in (3), the text "*we have collected on those assets before we can move forward*" was incorrectly parsed as a single clause covered by an S node with the subordinate "*before we can move forward*" as a child of this S node. However, in the correct parse tree the subordinate clause should be a sibling of the S node.

(3) *We would have to wait* until **we have collected on those assets** before we can move forward.[3]

4 Conclusion

In this paper, we described the CLaC Discourse Parser which was developed from scratch for the CoNLL 2015 shared task. This 3 person-month effort focused on the task of the *Discourse Connective Classification* and *Argument Labeler*. We used a naïve approach for sense labelling and consider only explicit relations. Yet, the parser achieves an overall F_1 measure of 17.38%, ranking in 6th place out of the 17 parsers submitted to the CoNLL 2015 shared task.

5 Acknowledgement

The authors would like to thank the CoNLL 2015 organizers and the anonymous reviewers. This

[3]The example is taken from the CoNLL 2015 development dataset.

work was financially supported by NSERC.

References

[Bethard et al.2014] Steven Bethard, Philip Ogren, and Lee Becker. 2014. ClearTK 2.0: Design patterns for machine learning in UIMA. LREC.

[Ferrucci and Lally2004] David Ferrucci and Adam Lally. 2004. UIMA: An architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348.

[Hall et al.2009] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

[Kong et al.2014] 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.

[Lin et al.2014] 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.

[Petrov and Klein2007] Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *Proceedings of NAACL HLT 2007*, page 404–411, Rochester, NY, April.

[Pitler et al.2009] Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, page 683–691, Suntec, Singapore, August.

[Quinlan1993] J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.