A Constituent-Based Approach to Argument Labeling with Joint Inference in Discourse Parsing

Fang Kong[1*] **Hwee Tou Ng**[2] **Guodong Zhou**[1]
[1] School of Computer Science and Technology, Soochow University, China
[2] Department of Computer Science, National University of Singapore
kongfang@suda.edu.cn nght@comp.nus.edu.sg gdzhou@suda.edu.cn

Abstract

Discourse parsing is a challenging task and plays a critical role in discourse analysis. In this paper, we focus on labeling full argument spans of discourse connectives in the Penn Discourse Treebank (PDTB). Previous studies cast this task as a linear tagging or subtree extraction problem. In this paper, we propose a novel constituent-based approach to argument labeling, which integrates the advantages of both linear tagging and subtree extraction. In particular, the proposed approach unifies intra- and inter-sentence cases by treating the immediately preceding sentence as a special constituent. Besides, a joint inference mechanism is introduced to incorporate global information across arguments into our constituent-based approach via integer linear programming. Evaluation on PDTB shows significant performance improvements of our constituent-based approach over the best state-of-the-art system. It also shows the effectiveness of our joint inference mechanism in modeling global information across arguments.

1 Introduction

Discourse parsing determines the internal structure of a text and identifies the discourse relations between its text units. It has attracted increasing attention in recent years due to its importance in text understanding, especially since the release of the Penn Discourse Treebank (PDTB) corpus (Prasad et al., 2008), which adds a layer of discourse annotations on top of the Penn Treebank (PTB) corpus (Marcus et al., 1993). As the largest available discourse corpus, the PDTB corpus has become the defacto benchmark in recent studies on discourse parsing.

Compared to connective identification and discourse relation classification in discourse parsing, the task of labeling full argument spans of discourse connectives is much harder and thus more challenging. For connective identification, Lin et al. (2014) achieved the performance of 95.76% and 93.62% in F-measure using gold-standard and automatic parse trees, respectively. For discourse relation classification, Lin et al. (2014) achieved the performance of 86.77% in F-measure on classifying discourse relations into 16 level 2 types. However, for argument labeling, Lin et al. (2014) only achieved the performance of 53.85% in F-measure using gold-standard parse trees and connectives, much lower than the inter-annotation agreement of 90.20% (Miltsakaki et al., 2004).

In this paper, we focus on argument labeling in the PDTB corpus. In particular, we propose a *novel* constituent-based approach to argument labeling which views constituents as candidate arguments. Besides, our approach unifies intra- and inter-sentence cases by treating the immediately preceding sentence as a special constituent. Finally, a joint inference mechanism is introduced to incorporate global information across arguments via integer linear programming. Evaluation on the PDTB corpus shows the effectiveness of our approach.

The rest of this paper is organized as follows. Section 2 briefly introduces the PDTB corpus. Related work on argument labeling is reviewed in Section 3. In Section 4, we describe our constituent-based approach to argument labeling. In Section 5, we present our joint inference mechanism via integer linear programming (ILP). Section 6 gives the experimental results and analysis. Finally, we conclude in Section 7.

[*] The research reported in this paper was carried out while Fang Kong was a research fellow at the National University of Singapore.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 68–77,
October 25-29, 2014, Doha, Qatar.

2 Penn Discourse Treebank

As the first large-scale annotated corpus that follows the lexically grounded, predicate-argument approach in D-LTAG (Lexicalized Tree Adjoining Grammar for Discourse) (Webber, 2004), the PDTB regards a connective as the predicate of a discourse relation which takes exactly two text spans as its arguments. In particular, the text span that the connective is syntactically attached to is called Arg2, and the other is called Arg1.

Although discourse relations can be either explicitly or implicitly expressed in PDTB, this paper focuses only on explicit discourse relations that are explicitly signaled by discourse connectives. Example (1) shows an explicit discourse relation from the article wsj_2314 with connective so underlined, Arg1 span *italicized*, and Arg2 span **bolded**.

(1) *But its competitors have much broader business interests* **and** so **are better cushioned against price swings .**

Note that a connective and its arguments can appear in any relative order, and an argument can be arbitrarily far away from its corresponding connective. Although the position of Arg2 is fixed once the connective is located, Arg1 can occur in the same sentence as the connective (SS), in a sentence preceding that of the connective (PS), or in a sentence following that of the connective (FS), with proportions of 60.9%, 39.1%, and less than 0.1% respectively for explicit relations in the PDTB corpus (Prasad et al., 2008). Besides, out of all PS cases where Arg1 occurs in some preceding sentence, 79.9% of them are the exact immediately preceding sentence. As such, in this paper, we only consider the current sentence containing the connective and its immediately preceding sentence as the text span where Arg1 occurs, similar to what was done in (Lin et al., 2014).

3 Related Work

For argument labeling in discourse parsing on the PDTB corpus, the related work can be classified into two categories: locating parts of arguments, and labeling full argument spans.

As a representative on locating parts of arguments, Wellner and Pustejovsky (2007) proposed several machine learning approaches to identify the head words of the two arguments for discourse connectives. Following this work, Elwell and Baldridge (2008) combined general and connective specific rankers to improve the performance of labeling the head words of the two arguments. Prasad et al. (2010) proposed a set of heuristics to locate the position of the Arg1 sentences for inter-sentence cases. The limitation of locating parts of arguments, such as the positions and head words, is that it is only a partial solution to argument labeling in discourse parsing.

In comparison, labeling full argument spans can provide a complete solution to argument labeling in discourse parsing and has thus attracted increasing attention recently, adopting either a subtree extraction approach (Dinesh et al. (2005), Lin et al. (2014)) or a linear tagging approach (Ghosh et al. (2011)).

As a representative subtree extraction approach, Dinesh et al. (2005) proposed an automatic tree subtraction algorithm to locate argument spans for intra-sentential subordinating connectives. However, only dealing with intra-sentential subordinating connectives is not sufficient since they constitute only 40.93% of all cases. Instead, Lin et al. (2014) proposed a two-step approach. First, an argument position identifier was employed to locate the position of Arg1. For the PS case, it directly selects the immediately preceding sentence as Arg1. For other cases, an argument node identifier was employed to locate the Arg1- and Arg2-nodes. Next, a tree subtraction algorithm was used to extract the arguments. However, as pointed out in Dinesh et al. (2005), it is not necessarily the case that a connective, Arg1, or Arg2 is dominated by a single node in the parse tree (that is, it can be dominated by a set of nodes). Figure 1 shows the gold-standard parse tree corresponding to Example (1). It shows that Arg1 includes three nodes: [$_{CC}$ But], [$_{NP}$ its competitors], [$_{VP}$ have much broader business interests], and Arg2 includes two nodes: [$_{CC}$ and], [$_{VP}$ are better cushioned against price swings]. Therefore, such an argument node identifier has inherent shortcomings in labeling arguments. Besides, the errors propagated from the upstream argument position classifier may adversely affect the performance of the downstream argument node identifier.

As a representative linear tagging approach, Ghosh et al. (2011) cast argument labeling as a linear tagging task using conditional random fields. Ghosh et al. (2012) further improved the perfor-

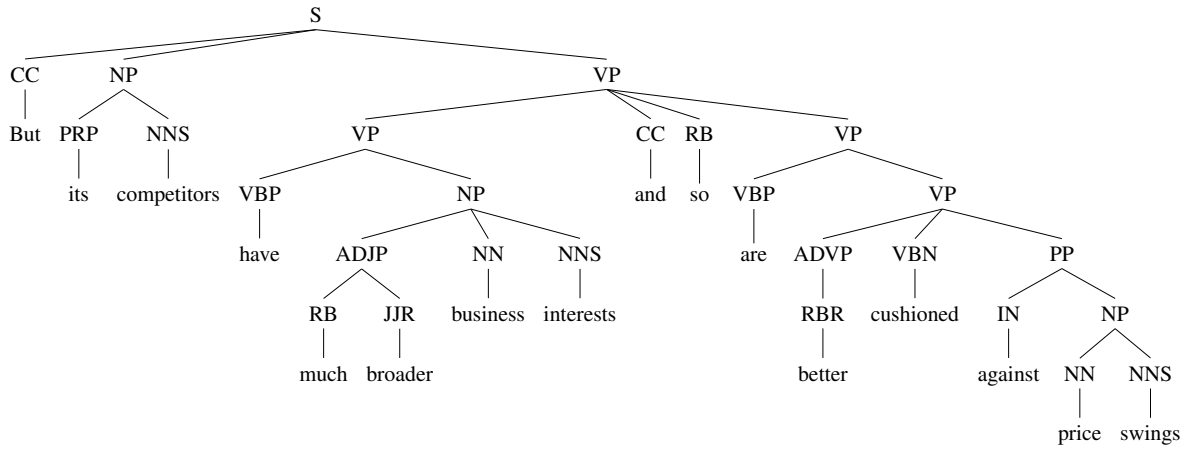

Figure 1: The gold-standard parse tree corresponding to Example (1)

mance with integration of the n-best results.

While the subtree extraction approach locates argument spans based on the nodes of a parse tree and is thus capable of using rich syntactic information, the linear tagging approach works on the tokens in a sentence and is thus capable of capturing local sequential dependency between tokens. In this paper, we take advantage of both subtree extraction and linear tagging approaches by proposing a novel constituent-based approach. Furthermore, intra- and inter-sentence cases are unified by treating the immediately preceding sentence as a special constituent. Finally, a joint inference mechanism is proposed to add global information across arguments.

4 A Constituent-Based Approach to Argument Labeling

Our constituent-based approach works by first casting the constituents extracted from a parse tree as argument candidates, then determining the role of every constituent as part of Arg1, Arg2, or NULL, and finally, merging all the constituents for Arg1 and Arg2 to obtain the Arg1 and Arg2 text spans respectively. Obviously, the key to the success of our constituent-based approach is constituent-based argument classification, which determines the role of every constituent argument candidate.

As stated above, the PDTB views a connective as the predicate of a discourse relation. Similar to semantic role labeling (SRL), for a given connective, the majority of the constituents in a parse tree may not be its arguments (Xue and Palmer, 2004). This indicates that negative instances (constituents marked NULL) may overwhelm positive instances. To address this problem, we use a simple algorithm to prune out these constituents which are clearly not arguments to the connective in question.

4.1 Pruning

The pruning algorithm works recursively in preprocessing, starting from the target connective node, i.e. the lowest node dominating the connective. First, all the siblings of the connective node are collected as candidates, then we move on to the parent of the connective node and collect its siblings, and so on until we reach the root of the parse tree. In addition, if the target connective node does not cover the connective exactly, the children of the target connective node are also collected.

For the example shown in Figure 1, we can locate the target connective node [$_{RB}$ so] and return five constituents — [$_{VP}$ have much broader business interests], [$_{CC}$ and], [$_{VP}$ are better cushioned against price swings], [$_{CC}$ But], and [$_{NP}$ its competitors] — as argument candidates.

It is not surprising that the pruning algorithm works better on gold parse trees than automatic parse trees. Using gold parse trees, our pruning algorithm can recall 89.56% and 92.98% (489 out of 546 Arg1s, 808 out of 869 Arg2s in the test data) of the Arg1 and Arg2 spans respectively and prune out 81.96% (16284 out of 19869) of the nodes in the parse trees. In comparison, when automatic parse trees (based on the Charniak parser (Charniak, 2000)) are used, our pruning algorithm can recall 80.59% and 89.87% of the Arg1 and Arg2 spans respectively and prune out 81.70% (16190

Feature	Description	Example
CON-Str	The string of the given connective (case-sensitive)	so
CON-LStr	The lowercase string of the given connective	so
CON-Cat	The syntactic category of the given connective: subordinating, coordinating, or discourse adverbial	Subordinating
CON-iLSib	Number of left siblings of the connective	2
CON-iRSib	Number of right siblings of the connective	1
NT-Ctx	The context of the constituent. We use POS combination of the constituent, its parent, left sibling and right sibling to represent the context. When there is no parent or siblings, it is marked NULL.	VP-VP-NULL-CC
CON-NT-Path	The path from the parent node of the connective to the node of the constituent	$RB \uparrow VP \downarrow VP$
CON-NT-Position	The position of the constituent relative to the connective: left, right, or previous	left
CON-NT-Path-iLsib	The path from the parent node of the connective to the node of the constituent and whether the number of left siblings of the connective is greater than one	$RB \uparrow VP \downarrow VP$:>1

Table 1: Features employed in argument classification.

out of 19816) of the nodes in the parse trees.

4.2 Argument Classification

In this paper, a multi-category classifier is employed to determine the role of an argument candidate (i.e., Arg1, Arg2, or NULL). Table 1 lists the features employed in argument classification, which reflect the properties of the connective and the candidate constituent, and the relationship between them. The third column of Table 1 shows the features corresponding to Figure 1, considering [$_{RB}$ so] as the given connective and [$_{VP}$ have much broader business interests] as the constituent in question.

Similar to Lin et al. (2014), we obtained the syntactic category of the connectives from the list provided in Knott (1996). However, different from Lin et al. (2014), only the siblings of the root path nodes (i.e., the nodes occurring in the path of the connective to root) are collected as the candidate constituents in the pruning stage, and the value of the relative position can be left or right, indicating that the constituent is located on the left- or right-hand of the root path respectively. Besides, we view the root of the previous sentence as a special candidate constituent. For example, the value of the feature CON-NT-Position is *previous* when the current constituent is the root of the previous sentence. Finally, we use the part-of-speech (POS) combination of the constituent itself, its parent node, left sibling node and right sibling node to represent the context of the candidate constituent. Intuitively, this information can help determine the role of the constituent.

For the example shown in Figure 1, we first employ the pruning algorithm to get the candidate constituents, and then employ our argument classifier to determine the role for every candidate. For example, if the five candidates are labeled as Arg1, Arg2, Arg2, Arg1, and Arg1, respectively, we merge all the Arg1 constituents to obtain the Arg1 text span (i.e., *But its competitors have much broader business interests*). Similarly, we merge the two Arg2 constituents to obtain the Arg2 text span (i.e., **and are better cushioned against price swings**).

5 Joint Inference via Integer Linear Programming

In the above approach, decisions are always made for each candidate independently, ignoring global information across candidates in the final output. For example, although an argument span can be split into multiple discontinuous segments (e.g., the Arg2 span of Example (1) contains two discontinuous segments, **and**, **are better cushioned against price swings**), the number of discontinuous segments is always limited. Statistics on the PDTB corpus shows that the number of discontin-

uous segments for both Arg1 and Arg2 is generally ($>= 99\%$) at most 2. For Example (1), from left to right, we can obtain the list of constituent candidates: [$_{CC}$ But], [$_{NP}$ its competitors], [$_{VP}$ have much broader business interests], [$_{CC}$ and], [$_{VP}$ are better cushioned against price swings]. If our argument classifier wrongly determines the roles as Arg1, Arg2, Arg1, Arg2, and Arg1 respectively, we can find that the achieved Arg1 span contains three discontinuous segments. Such errors may be corrected from a global perspective.

In this paper, a joint inference mechanism is introduced to incorporate various kinds of knowledge to resolve the inconsistencies in argument classification to ensure global legitimate predictions. In particular, the joint inference mechanism is formalized as a constrained optimization problem, represented as an integer linear programming (ILP) task. It takes as input the argument classifiers' confidence scores for each constituent candidate along with a list of constraints, and outputs the optimal solution that maximizes the objective function incorporating the confidence scores, subject to the constraints that encode various kinds of knowledge.

In this section, we meet the requirement of ILP with focus on the definition of variables, the objective function, and the problem-specific constraints, along with ILP-based joint inference integrating multiple systems.

5.1 Definition of Variables

Given an input sentence, the task of argument labeling is to determine what labels should be assigned to which constituents corresponding to which connective. It is therefore natural that encoding the output space of argument labeling requires various kinds of information about the connectives, the argument candidates corresponding to a connective, and their argument labels.

Given an input sentence s, we define following variables:

(1) P: the set of connectives in a sentence.

(2) $p \in P$: a connective in P.

(3) $C(p)$: the set of argument candidates corresponding to connective p. (i.e., the parse tree nodes obtained in the pruning stage).

(4) $c \in C(p)$: an argument candidate.

(5) L: the set of argument labels {Arg1, Arg2, NULL }.

(6) $l \in L$: an argument label in L.

In addition, we define the integer variables as follows:

$$Z^l_{c,p} \in \{0, 1\} \tag{1}$$

If $Z^l_{c,p} = 1$, the argument candidate c, which corresponds to connective p, should be assigned the label l. Otherwise, the argument candidate c is not assigned this label.

5.2 The Objective Function

The objective of joint inference is to find the best arguments for all the connectives in one sentence. For every connective, the pruning algorithm is first employed to determine the set of corresponding argument candidates. Then, the argument classifier is used to assign a label to every candidate. For an individual labeling $Z^l_{c,p}$, we measure the quality based on the confidence scores, $f_{l,c,p}$, returned by the argument classifier. Thus, the objective function can be defined as

$$\max \sum_{l,c,p} f_{l,c,p} Z^l_{c,p} \tag{2}$$

5.3 Constraints

As the key to the success of ILP-based joint inference, the following constraints are employed:

Constraint 1: The arguments corresponding to a connective cannot overlap with the connective. Let $c_1, c_2..., c_k$ be the argument candidates that correspond to the same connective and overlap with the connective in a sentence.[1] Then this constraint ensures that none of them will be assigned as Arg1 or Arg2.

$$\sum_{i=1}^{k} Z^{NULL}_{c_i,p} = k \tag{3}$$

Constraint 2: There are no overlapping or embedding arguments. Let $c_1, c_2..., c_k$ be the argument candidates that correspond to the same connective and cover the same word in a sentence.[2]

[1] Only when the target connective node does not cover the connective exactly and our pruning algorithm collects all the children of the target connective node as part of constituent candidates, such overlap can be introduced.

[2] This constraint only works in system combination of Section 5.4, where additional phantom candidates may introduce such overlap.

Then this constraint ensures that at most one of the constituents can be assigned as Arg1 or Arg2. That is, at least $k-1$ constituents should be assigned the special label NULL.

$$\sum_{i=1}^{k} Z_{c_i,p}^{NULL} \geq k-1 \quad (4)$$

Constraint 3: For a connective, there is at least one constituent candidate assigned as Arg2.

$$\sum_{c} Z_{c,p}^{Arg2} \geq 1 \quad (5)$$

Constraint 4: Since we view the previous complete sentence as a special Arg1 constituent candidate, denoted as m, there is at least one candidate assigned as Arg1 for every connective.

$$\sum_{c} Z_{c,p}^{Arg1} + Z_{m,p}^{Arg1} \geq 1 \quad (6)$$

Constraint 5: The number of discontinuous constituents assigned as Arg1 or Arg2 should be at most 2. That is, if argument candidates $c_1, c_2..., c_k$ corresponding to the same connective are discontinuous, this constraint ensures that at most two of the constituents can be assigned the same label Arg1 or Arg2.

$$\sum_{i=1}^{k} Z_{c_i,p}^{Arg1} \leq 2, and \sum_{i=1}^{k} Z_{c_i,p}^{Arg2} \leq 2 \quad (7)$$

5.4 System Combination

Previous work shows that the performance of argument labeling heavily depends on the quality of the syntactic parser. It is natural that combining different argument labeling systems on different parse trees can potentially improve the overall performance of argument labeling.

To explore this potential, we build two argument labeling systems — one using the Berkeley parser (Petrov et al., 2006) and the other the Charniak parser (Charniak, 2000). Previous studies show that these two syntactic parsers tend to produce different parse trees for the same sentence (Zhang et al., 2009). For example, our preliminary experiment shows that applying the pruning algorithm on the output of the Charniak parser produces a list of candidates with recall of 80.59% and 89.87% for Arg1 and Arg2 respectively, while achieving recall of 78.6% and 91.1% for Arg1 and Arg2 respectively on the output of the Berkeley parser. It also shows that combining these two candidate lists significantly improves recall to 85.7% and 93.0% for Arg1 and Arg2, respectively.

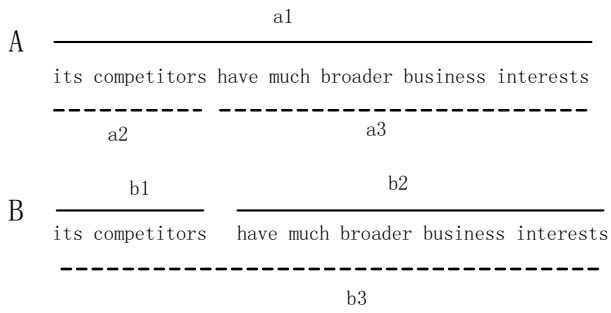

Figure 2: An example on unifying different candidates.

In subsection 5.2, we only consider the confidence scores returned by an argument classifier. Here, we proceed to combine the probabilities produced by two argument classifiers. There are two remaining problems to resolve:

- How do we unify the two candidate lists? In principle, constituents spanning the same sequence of words should be viewed as the same candidate. That is, for different candidates, we can unify them by adding phantom candidates. This is similar to the approach proposed by Punyakanok et al. (2008) for the semantic role labeling task. For example, Figure 2 shows the candidate lists generated by our pruning algorithm based on two different parse trees given the segment "its competitors have much broader business interests". Dashed lines are used for phantom candidates and solid lines for true candidates. Here, system A produces one candidate a1, with two phantom candidates a2 and a3 added. Analogously, phantom candidate b3 is added to the candidate list output by System B. In this way, we can get the unified candidate list: "its competitors have much broader business interests", "its competitors", "have much broader business interests".

- How do we compute the confidence score for every decision? For every candidate in the unified list, we first determine whether it is a true candidate based on the specific parse tree. Then, for a true candidate, we extract the features from the corresponding parse

tree. On this basis, we can determine the confidence score using our argument classifier. For a phantom candidate, we set the same prior distribution as the confidence score. In particular, the probability of the "NULL" class is set to 0.55, following (Punyakanok et al., 2008), and the probabilities of Arg1 and Arg2 are set to their occurrence frequencies in the training data. For the example shown in Figure 2, since System A returns "its competitors have much broader business interests" as a true candidate, we can obtain its confidence score using our argument classifier. For the two phantom candidates — "its competitors" and "have much broader business interests" — we use the prior distributions directly. This applies to the candidates for System B. Finally, we simply average the estimated probabilities to determine the final probability estimate for every candidate in the unified list.

6 Experiments

In this section, we systematically evaluate our constituent-based approach with a joint inference mechanism to argument labeling on the PDTB corpus.

6.1 Experimental settings

All our classifiers are trained using the OpenNLP maximum entropy package[3] with the default parameters (i.e. without smoothing and with 100 iterations). As the PDTB corpus is aligned with the PTB corpus, the gold parse trees and sentence boundaries are obtained from PTB. Under the automatic setting, the NIST sentence segmenter[4] and the Charniak parser[5] are used to segment and parse the sentences, respectively. lp_solve[6] is used for our joint inference.

This paper focuses on automatically labeling the full argument spans of discourse connectives. For a fair comparison with start-of-the-art systems, we use the NUS PDTB-style end-to-end discourse parser[7] to perform other subtasks of discourse parsing except argument labeling, which includes connective identification, non-explicit discourse relation identification and classification.

Finally, we evaluate our system on two aspects: (1) the dependence on the parse trees (GS/Auto, using gold standard or automatic parse trees and sentence boundaries); and (2) the impact of errors propagated from previous components (noEP/EP, using gold annotation or automatic results from previous components). In combination, we have four different settings: GS+noEP, GS+EP, Auto+noEP and Auto+EP. Same as Lin et al. (2014), we report exact match results under these four settings. Here, exact match means two spans match identically, except beginning or ending punctuation symbols.

6.2 Experimental results

We first evaluate the effectiveness of our constituent-based approach by comparing our system with the state-of-the-art systems, ignoring the joint inference mechanism. Then, the contribution of the joint inference mechanism to our constituent-based approach, and finally the contribution of our argument labeling system to the end-to-end discourse parser are presented.

Effectiveness of our constituent-based approach

By comparing with two state-of-the-art argument labeling approaches, we determine the effectiveness of our constituent-based approach.

Comparison with the linear tagging approach

As a representative linear tagging approach, Ghosh et al. (2011; 2012; 2012) only reported the exact match results for Arg1 and Arg2 using the evaluation script for chunking evaluation[8] under GS+noEP setting with Section 02–22 of the PDTB corpus for training, Section 23–24 for testing, and Section 00–01 for development. It is also worth mentioning that an argument span can contain multiple discontinuous segments (i.e., chunks), so chunking evaluation only shows the exact match of every argument segment but not the exact match of every argument span. In order to fairly compare our system with theirs, we evaluate our system using both the exact metric and the chunking evaluation. Table 2 compares the results of our system without joint inference and the results reported by Ghosh et al. (2012) on the same data split. We can find that our system performs much bet-

[3] http://maxent.sourceforge.net/

[4] http://duc.nist.gov/duc2004/software/duc2003.breakSent.tar.gz

[5] ftp://ftp.cs.brown.edu/pub/nlparser/

[6] http://lpsolve.sourceforge.net/

[7] http://wing.comp.nus.edu.sg/ linzihen/parser/

[8] http://www.cnts.ua.ac.be/conll2000/chunking/conlleval.txt

ter than Ghosh's on both Arg1 and Arg2, even on much stricter metrics.

Systems	Arg1	Arg2
ours using exact match	65.68	84.50
ours using chunking evaluation	67.48	88.08
reported by Ghosh et al. (2012)	59.39	79.48

Table 2: Performance (F1) comparison of our argument labeling approach with the linear tagging approach as adopted in Ghosh et al. (2012)

Comparison with the subtree extracting approach

For a fair comparison, we also conduct our experiments on the same data split of Lin et al. (2014) with Section 02 to 21 for training, Section 22 for development, and Section 23 for testing. Table 3 compares our labeling system without joint inference with Lin et al. (2014), a representative subtree extracting approach. From the results, we find that the performance of our argument labeling system significantly improves under all settings. This is because Lin et al. (2014) considered all the internal nodes of the parse trees, whereas the pruning algorithm in our approach can effectively filter out those unlikely constituents when determining Arg1 and Arg2.

	Setting	Arg1	Arg2	Arg1&2
ours	GS+noEP	62.84	84.07	55.69
	GS+EP	61.46	81.30	54.31
	Auto+EP	56.04	76.53	48.89
Lin's	GS+noEP	59.15	82.23	53.85
	GS+EP	57.64	79.80	52.29
	Auto+EP	47.68	70.27	40.37

Table 3: Performance (F1) comparison of our argument labeling approach with the subtree extraction approach as adopted in Lin et al. (2014)

As justified above, by integrating the advantages of both linear tagging and subtree extraction, our constituent-based approach can capture both rich syntactic information from parse trees and local sequential dependency between tokens. The results show that our constituent-based approach indeed significantly improves the performance of argument labeling, compared to both linear tagging and subtree extracting approaches.

Contribution of Joint Inference

Same as Lin et al. (2014), we conduct our experiments using Section 02 to 21 for training, Section 22 for development, and Section 23 for testing. Table 4 lists the performance of our argument labeling system without and with ILP inference under four different settings, while Table 5 reports the contribution of system combination. It shows the following:

- On the performance comparison of Arg1 and Arg2, the performance on Arg2 is much better than that on Arg1 with the performance gap up to 8% under different settings. This is due to the fact that the relationship between Arg2 and the connective is much closer. This result is also consistent with previous studies on argument labeling.

- On the impact of error propagation from connective identification, the errors propagated from connective identification reduce the performance of argument labeling by less than 2% in both Arg1 and Arg2 F-measure under different settings.

- On the impact of parse trees, using automatic parse trees reduces the performance of argument labeling by about 5.5% in both Arg1 and Arg2 F-measure under different settings. In comparison with the impact of error propagation, parse trees have much more impact on argument labeling.

- On the impact of joint inference, it improves the performance of argument labeling, especially on automatic parse trees by about 2%.[9]

- On the impact of system combination, the performance is improved by about 1.5%.

	Setting	Arg1	Arg2	Arg1&2
without Joint Inference	GS+noEP	62.84	84.07	55.69
	GS+EP	61.46	81.30	54.31
	Auto+noEP	57.75	79.85	50.27
	Auto+EP	56.04	76.53	48.89
with Joint Inference	GS+noEP	65.76	83.86	58.18
	GS+EP	63.96	81.19	56.37
	Auto+noEP	60.24	79.74	52.55
	Auto+EP	58.10	76.53	50.73

Table 4: Performance (F1) of our argument labeling approach.

Contribution to the end-to-end discourse parser

[9]Unless otherwise specified, all the improvements in this paper are significant with $p < 0.001$.

Systems	Setting	Arg1	Arg2	Arg1&2
Charniak	noEP	60.24	79.74	52.55
	EP	58.10	76.53	50.73
Berkeley	noEP	60.78	80.07	52.98
	EP	58.80	77.21	51.43
Combined	noEP	61.97	80.61	54.50
	EP	59.72	77.55	52.52

Table 5: Contribution of System Combination in Joint Inference.

Lastly, we focus on the contribution of our argument labeling approach to the overall performance of the end-to-end discourse parser. This is done by replacing the argument labeling model of the NUS PDTB-style end-to-end discourse parser with our argument labeling model. Table 6 shows the results using gold parse trees and automatic parse trees, respectively.[10] From the results, we find that using gold parse trees, our argument labeling approach significantly improves the performance of the end-to-end system by about 1.8% in F-measure, while using automatic parse trees, the improvement significantly enlarges to 6.7% in F-measure.

Setting	New d-parser	Lin et al.'s (2014)
GS	34.80	33.00
Auto	27.39	20.64

Table 6: Performance (F1) of the end-to-end discourse parser.

7 Conclusion

In this paper, we focus on the problem of automatically labeling the full argument spans of discourse connectives. In particular, we propose a constituent-based approach to integrate the advantages of both subtree extraction and linear tagging approaches. Moreover, our proposed approach integrates inter- and intra-sentence argument labeling by viewing the immediately preceding sentence as a special constituent. Finally, a joint inference mechanism is introduced to incorporate global information across arguments into our constituent-based approach via integer linear programming.

[10]Further analysis found that the error propagated from sentence segmentation can reduce the performance of the end-to-end discourse parser. Retraining the NIST sentence segmenter using Section 02 to 21 of the PDTB corpus, the original NUS PDTB-style end-to-end discourse parser can achieve the performance of 25.25% in F-measure, while the new version (i.e. replace the argument labeling model with our argument labeling model) can achieve the performance of 30.06% in F-measure.

Acknowledgments

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office. This research is also partially supported by Key project 61333018 and 61331011 under the National Natural Science Foundation of China.

References

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139.

Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2005. Attribution and the (non-)alignment of syntactic and discourse arguments of connectives. In *Proceedings of the Workshop on Frontiers in Corpus Annotation II: Pie in the Sky*, pages 29–36.

Robert Elwell and Jason Baldridge. 2008. Discourse connective argument identification with connective specific rankers. In *Second IEEE International Conference on Semantic Computing*, pages 198–205.

Sucheta Ghosh, Richard Johansson, Giuseppe Riccardi, and Sara Tonelli. 2011. Shallow discourse parsing with conditional random fields. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 1071–1079.

Sucheta Ghosh, Giuseppe Riccardi, and Richard Johansson. 2012. Global features for shallow discourse parsing. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 150–159.

Sucheta Ghosh. 2012. *End-to-End Discourse Parsing using Cascaded Structured Prediction*. Ph.D. thesis, University of Trento.

Alistair Knott. 1996. *A Data-Driven Methodology for Motivating a Set of Coherence Relations*. Ph.D. thesis, University of Edinburgh.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A PDTB-styled end-to-end discourse parser. *Natural Language Engineering*, 20(2):151–184.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2004. The Penn Discourse Treebank. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, pages 2237–2240.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of the LREC 2008 Conference*, pages 2961–2968.

Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2010. Exploiting scope for shallow discourse parsing. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The important of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Bonnie Webber. 2004. D-LTAG: extending lexicalized TAG to discourse. *Cognitive Science*, 28(5):751–779.

Ben Wellner and James Pustejovsky. 2007. Automatically identifying the arguments of discourse connectives. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 92–101.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of 2004 Conference on Empirical Methods in Natural Language Processing*, pages 88–94.

Hui Zhang, Min Zhang, Chew Lim Tan, and Haizhou Li. 2009. K-best combination of syntactic parsers. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1552–1560.

Addendum to Paper
approved by SIGDAT board in October 2015

This addendum concerns EMNLP 2014 paper
Fang Kong; Hwee Tou Ng; Guodong Zhou. A Constituent-Based Approach to Argument Labeling with Joint Inference in Discourse 68 - 77.

Due to experimental setting and evaluation problems in the code written by the first author for the paper, the reported results can not be replicated and the conclusion that joint inference significantly improves the performance of argument labeling fails to be supported.