Natural Language Engineering **1** (1): 1–35. Printed in the United Kingdom

A PDTB-Styled End-to-End Discourse Parser

ZIHENG LIN[1,2], HWEE TOU NG[1] , and MIN-YEN KAN [1]

[1] *Department of Computer Science, National University of Singapore*
13 Computing Drive, Singapore 117417
email: {linzihen, nght, kanmy}@comp.nus.edu.sg
[2] *SAP Research, SAP Asia Pte Ltd*
30 Pasir Panjang Road, Singapore 117440
email: ziheng.lin@sap.com

(*Received 22 September 2011; revised 28 February 2012; revised 12 August 2012*)

Abstract

Since the release of the large discourse-level annotation of the Penn Discourse Treebank (PDTB), research work has been carried out on certain subtasks of this annotation, such as disambiguating discourse connectives and classifying Explicit or Implicit relations. We see a need to construct a full parser on top of these subtasks and propose a way to evaluate the parser. In this work, we have designed and developed an end-to-end discourse parser to parse free texts in the PDTB style in a fully data-driven approach. The parser consists of multiple components joined in a sequential pipeline architecture, which includes a connective classifier, argument labeler, explicit classifier, non-explicit classifier, and attribution span labeler. Our trained parser first identifies all discourse and non-discourse relations, locates and labels their arguments, and then classifies the sense of the relation between each pair of arguments. For the identified relations, the parser also determines the attribution spans, if any, associated with them. We introduce novel approaches to locate and label arguments, and to identify attribution spans. We also significantly improve on the current state-of-the-art connective classifier. We propose and present a comprehensive evaluation from both component-wise and error-cascading perspectives, in which we illustrate how each component performs in isolation, as well as how the pipeline performs with errors propagated forward. The parser gives an overall system F_1 score of 46.80% for partial matching utilizing gold standard parses, and 38.18% with full automation.

1 Introduction

A piece of text is often not to be understood individually, but understood by linking it with other text units from its context. These units can be surrounding clauses, sentences, or even paragraphs. A text span may connect to another span, because there is a causal relation between them. Two text spans may also be connected when they contrast each other. Such semantic relations are termed *rhetorical* or *discourse relations*. A text becomes semantically well-structured and understandable when its individual text units relate to each other to form connections that can be recognized as higher-level prose argumentation and presentation structures.

However even when a text is well-structured, it is not a trivial task to automatically

derive the discourse relations that hold the text together. In natural language processing (NLP), *discourse parsing* is the process of understanding the internal structure of a text and identifying the discourse relations in between its text units. Over the last three decades, researchers have proposed a number of discourse frameworks from different perspectives for the purpose of discourse analysis and parsing (Grosz and Sidner 1986; Mann and Thompson 1988; Polanyi 1988; Hobbs 1990; Lascarides and Asher 1993; Knott and Sanders 1998; Webber 2004; Wolf and Gibson 2005). However, designing and constructing such a discourse parser has been a difficult task, partially attributable to the lack of large-scale annotated data sets.

The Penn Discourse Treebank (PDTB) (Prasad *et al.* 2008) is a recently released, discourse-level annotation aligned with the Penn Treebank (PTB) (Marcus *et al.* 1993), which aims to fill this need. Providing a common platform for discourse researchers, it is the first annotation framework that follows the lexically grounded, predicate-argument approach, as proposed in Webber's framework (2004). The PDTB provides annotations for relations that are explicitly signaled by discourse connectives[1] as well as implied relations, the argument spans of each relation, the sense of each relation, and when present, their attribution spans. The following annotation shows a Condition relation between the italicized and bolded spans that is explicitly signaled by the connective "if"; the text span in the box is an attribution span.

The Treasury said *the U.S. will default on Nov. 9* if **Congress doesn't act by then.**

The predicate of this relation is the explicit discourse connective "if", and the two arguments to the predicate are the spans in italics and in bold. An attribution is a relation between an agent and the abstract object denoted by the discourse relation or an argument of a discourse relation. In this example, "The Treasury said" is the agent of the attribution that covers the entire relation.

Implicit relations capture discourse relations that are not explicitly signaled by discourse connectives. The following example is an Implicit Contrast relation. Here, annotators inferred a connective "however" which can be inserted in between these two sentences to reflect the relation.

She was untrained and, in one botched job killed a client.
Her remorse was shallow and brief.

There are a number of challenges associated with discourse parsing in the PDTB. They include identifying discourse connectives from non-discourse ones, labeling the argument spans of the relations, classifying the relation senses of both Explicit and Implicit relations, and labeling the attribution spans. All of these steps need to be tackled in order to build a fully automatic, end-to-end discourse parser in the PDTB style. Since its release, much research has been carried out on the subtasks of the PDTB, such as identifying discourse connectives and classifying Explicit or Implicit relations. In this work, we:

- design a parsing algorithm that performs discourse parsing in the PDTB representation; and

[1] In this article, *discourse connectives* refer to a subset of *connectives* that signals discourse relations.

- implement an end-to-end system that is based on this algorithm in a fully data-driven approach.

This system includes novel components to locate and label arguments as well as improved components from previous work. We also propose and present a comprehensive evaluation on the parser from both component-wise and error-cascading perspectives. To the best of our knowledge, this is the first parser that performs end-to-end discourse parsing in the PDTB style. The demo and source code of the parser have been released online[2] (Lin *et al.* 2010).

2 Related Work

Many discourse frameworks have been proposed in the literature of discourse modeling. Among them, there are the cohesive devices described by Halliday and Hasan (1976), Hobbs' inventory of coherence relations based on abductive reasoning (Hobbs 1985), the Rhetorical Structure Theory (RST) proposed by Mann and Thompson (1988), Grosz and Sidner (1986)'s models which aim to associate speakers' intentions with their focus of attention in discourse, the Linguistic Discourse Model (LDM) proposed by (Polanyi and Scha 1984; Scha and Polanyi 1988), the Lexicalized Tree Adjoining Grammar for Discourse (D-LTAG) by (Webber and Joshi 1998; Webber 2004; Forbes *et al.* 2003), the Segmented Discourse Representation Theory (SDRT) by Asher and Lascarides (2003) which provides a logically precise dynamic semantic interpretation, and the discourse model that associates discourse relations in a graph structure (Wolf and Gibson 2005). In the following, we will review a number of automatic systems that are based on these discourse frameworks. Readers are referred to (Webber *et al.* 2011) for a more complete review on discourse frameworks, algorithms for discourse structures, and discourse applications in language technology.

Mann and Thompson (1988) proposed Rhetorical Structure Theory (RST) which takes a nucleus-satellite view on rhetorical relations, in which the satellite text span plays a subordinate role to the main nucleus. RST defines a set of rhetorical relations as well as discourse schemas for the structural constituency arrangements of text. As the RST schemas are recursive, they enable the embedding of relations, leading to a tree structure of the text. RST falls into the category that associates discourse relations with text structures.

Marcu (1997) formalized an algorithm to automatically parse an unrestricted text into its discourse tree using the RST framework. He made use of cue phrases to split a sentence into *elementary discourse units* (*edus*), designed algorithms that are able to recognize discourse relations with or without the signals of cue phrases, and proposed four algorithms for determining the valid discourse tree given the relations of adjacent *edus*.

Continuing in this vein, Soricut and Marcu (2003) introduced probabilistic models to segment a sentence into *edus*, and to derive their corresponding sentence-level discourse structure, using lexical and syntactic features. They experimented with their models using the RST Discourse Treebank (RST-DT) corpus (Carlson *et al.* 2001), which is annotated in

[2] `http://wing.comp.nus.edu.sg/~linzihen/parser/`

the RST framework and covers a small subset of the *Wall Street Journal* (WSJ), comprising 385 texts.

Huong *et al.* (2004) divided the discourse parsing process into two steps: first, they used syntactic information and cue phrases to segment sentences into *edus* and to generate discourse structures at sentence-level, and then generated text-level structure from the sentence-level ones in a constrained, bottom-up manner. They experimented with the RST-DT corpus and showed promising system performance on four different component tasks of: 1) sentence-level *edu* segmentation, 2) sentence- and text-level relation connection of text spans, 3) relation orientation (*i.e.*, nucleus vs. satellite) and 4) relation sense classification. In our system experiments, we also perform component-wise evaluations in a similar fashion.

Recently, duVerle and Prendinger (2009) made use of a support vector machine (SVM) approach, using a rich set of shallow lexical, syntactic, and structural features, to train two separate classifiers on identifying the rhetorical structures and labeling the rhetorical roles drawn from the RST-DT.

Wolf and Gibson (2005) proposed to use more complex discourse structures of *chain graphs* to represent discourse relations. They released an annotation of 135 articles in a corpus called Discourse Graphbank, which includes annotations for both Explicit and Implicit relations.

Wellner *et al.* (2006) used multiple knowledge sources to produce syntactic and lexico-semantic features, which were then used to automatically identify and classify Explicit and Implicit discourse relations in the Discourse Graphbank. Their experiments show that discourse connectives and the distance between two text spans have the most impact, and that event-based features also contribute to the performance. As they did not separate the experimental results for Explicit and Implicit relations, it is not possible to draw a conclusion on the performance on classifying Implicit relations.

Baldridge and Lascarides (2005) proposed to represent the discourse structures of SDRT (Asher and Lascarides 2003) in headed trees and model them with probabilistic head-driven parsing techniques. They showed that dialogue-based features can improve the models in both segmentation and relation recognition.

Subba and Di Eugenio (2009) presented a first-order learning approach to determine rhetorical relations between discourse segments and a modified shift-reduce parsing algorithm to build discourse parse trees. They showed improvements by exploiting compositional semantics and segment discourse structure data.

Webber (2004) developed the Discourse Lexicalized Tree Adjoining Grammar (D-LTAG) which associates discourse relations with lexical elements. In D-LTAG, discourse relations are triggered by lexical elements (*i.e.*, explicit or implicit discourse connectives), focusing on low-level discourse structures and semantics of monologues.

The PDTB is the first annotation that follows the lexically grounded, predicate-argument approach in the D-LTAG framework. The scope of the annotation is much larger in comparison with the RST-DT and the Discourse Graphbank, as it covers all *Wall Street Journal* (WSJ) sections in the PTB. With the advent of the PDTB, some recent work has attempted to recognize discourse connectives, relation senses, and argument spans in this newer corpus. In a preliminary study, Miltsakaki *et al.* (2005) used syntactic features and a Maximum Entropy model to classify the relation senses of three explicit connectives – "since",

"while", and "when". Using syntactic features extracted from the parse trees, Pitler and Nenkova (2009) introduced a model that is able to disambiguate the discourse usage of connectives and recognize Explicit relations. They extracted syntactic features from the constituent parses with regard to the connectives, and showed that it significantly outperforms the baselines on a 10-fold cross validation.

Dinesh *et al.* (2005) observed the connection between the syntactic structures and the annotation of argument spans for intra-sentential subordinating explicit connectives, and proposed an automatic algorithm that applies parse tree subtraction to locate such argument spans. Wellner and Pustejovsky (2007) and Wellner (2009) proposed machine learning approaches to identify the *head words* of the two arguments for discourse connectives in the PDTB. They utilized constituent features, dependency features, lexico-syntactic features, as well as the connective and its contextual features. Elwell and Baldridge (2008) followed this work with the use of general and connective-specific rankers and their combinations. Although their method is capable of locating the positions of the arguments, it is not able to label the span of these arguments. Prasad *et al.* (2010) proposed a set of heuristics to locate the position of the Arg1 sentences for Explicit relations in cases that the two arguments are not in the same sentence. Ghosh *et al.* (2011) designed the argument segmentation task for Explicit relations as a token-level sequence labeling task using conditional random fields (CRFs). They assumed gold standard discourse connectives and used a set of syntactic features in two classifiers designed for the two arguments.

Machine learning approaches have been applied on the PDTB to identify Implicit relations (*i.e.*, discourse relations that are not signaled explicitly by discourse connectives such as "because") in Pitler *et al.* (2009) and our previous work (Lin *et al.* 2009). Pitler *et al.* performed classification of Implicit discourse relations using several linguistically informed features, such as word polarity, verb classes, and word pairs, showing an increase in performance over a random classification baseline. Our classifier considers the context of the two arguments, word pair information, as well as the arguments' internal constituent and dependency parses, and the results yield a significant improvement over the majority baseline. Wang *et al.* (2010) subsequently employed the tree kernel and added temporal ordering information to automatically recognize and classify Explicit and Implicit discourse relations. Zhou *et al.* (2010) used a language model to automatically generate implicit connectives and presented two methods to use these connectives for recognition of Implicit relations.

All of these research efforts on the PDTB can be viewed as isolated components of a full parser. Our work differs from these prior efforts in that we design a parsing algorithm that connects all of these subtasks into a single pipeline, and we implement this pipeline into an end-to-end parser in the PDTB style. Our parser attempts to recognize explicit discourse connectives, identify relation senses and argument spans for both explicit and non-explicit relations, and recognize attribution spans for these relations. Component-wise, we introduce two novel approaches to accurately locate and label arguments, and to label attribution spans. We also significantly improve on the current state-of-the-art connective classifier with newly introduced features.

[1]Financial planners often urge investors to diversify *and* to hold a smattering of international securities. [2]*And* many emerging markets have outpaced more mature markets, such as the U.S. *and* Japan. [3]Country funds offer an easy way to get a taste of foreign stocks without the hard research of seeking out individual companies.

[4]*But* it doesn't take much to get burned. [5]Political *and* currency gyrations can whipsaw the funds. [6]Another concern: The funds' share prices tend to swing more than the broader market. [7]*When* the stock market dropped nearly 7% Oct. 13, *for instance*, the Mexico Fund plunged about 18% *and* the Spain Fund fell 16%. [8]*And* most country funds were clobbered more than most stocks *after* the 1987 crash.

Fig. 1. An excerpt from the article wsj_0034. Each sentence is preceded with its superscripted sentence number. All discourse and non-discourse connectives are italicized, with discourse connectives further underlined. All relations annotated in this excerpt are also shown in Examples 1 – 9 in Figure 2.

3 The Penn Discourse Treebank

The Penn Discourse Treebank (PDTB) covers the set of *Wall Street Journal* (WSJ) articles in the PTB – approximately one million words – which is much larger than previous annotations such as the RST-DT. The PDTB adopts a binary predicate-argument view on discourse relations, where a connective acts as a predicate that takes two text spans as its arguments. The span to which the connective is syntactically attached is called Arg2, while the other is called Arg1. The PDTB provides annotation for each discourse connective and its two arguments. Example 1 in Figure 2 shows one Explicit relation extracted from the excerpt in Figure 1, where the connective is underlined, Arg1 is *italicized*, and Arg2 is **bolded**. Figure 1 is an excerpt containing two consecutive paragraphs extracted from wsj_0034, which will be used as a running example throughout this paper.

The PDTB also examined sentence pairs within paragraphs for discourse relations other than Explicit relations. Example 4 shows such an Implicit relation where the annotator inferred an implicit connective "for example" that most intuitively connects Arg1 and Arg2. Some relations in the PDTB are *alternatively lexicalized* by non-connective expressions, *i.e.*, expressions that are not in the pre-defined, closed set of discourse connectives. Example 5 is such an AltLex relation with the non-connective expression "Another concern".

If no Implicit or AltLex relation exists between a sentence pair, annotators then checked whether an entity transition (EntRel) holds. EntRel captures cases where the same entity is realized in both sentences. If no EntRel is found, annotators labeled it as no relation (NoRel). Examples 2 and 10 show an EntRel and a NoRel relation, respectively. Explicit,

(1) *Financial planners often urge investors to diversify and to hold a smattering of international securities.* And **many emerging markets have outpaced more mature markets, such as the U.S. and Japan.**
(Expansion.Conjunction – wsj_0034)

(2) *And many emerging markets have outpaced more mature markets, such as the U.S. and Japan.* **Country funds offer an easy way to get a taste of foreign stocks without the hard research of seeking out individual companies.**
(EntRel – wsj_0034)

(3) *Country funds offer an easy way to get a taste of foreign stocks without the hard research of seeking out individual companies.* But **it doesn't take much to get burned.**
(Comparison.Contrast – wsj_0034)

(4) *But it doesn't take much to get burned.* Implicit = FOR EXAMPLE **Political and currency gyrations can whipsaw the funds.**
(Expansion.Restatement.Specification – wsj_0034)

(5) *Political and currency gyrations can whipsaw the funds.* AltLex [**Another concern**]: **The funds' share prices tend to swing more than the broader market.**
(Expansion.Conjunction – wsj_0034)

(6) When **the stock market dropped nearly 7% Oct. 13,** *for instance, the Mexico Fund plunged about 18% and the Spain Fund fell 16%.*
(Temporal.Synchrony – wsj_0034)

(7) *Another concern: The funds' share prices tend to swing more than the broader market.* **When the stock market dropped nearly 7% Oct. 13**, for instance, **the Mexico Fund plunged about 18% and the Spain Fund fell 16%.**
(Expansion.Instantiation – wsj_0034)

(8) When the stock market dropped nearly 7% Oct. 13, *for instance, the Mexico Fund plunged about 18%* and **the Spain Fund fell 16%**.
(Expansion.Conjunction – wsj_0034)

(9) When the stock market dropped nearly 7% Oct. 13, *for instance, the Mexico Fund plunged about 18% and the Spain Fund fell 16%.* And **most country funds were clobbered more than most stocks after the 1987 crash.**
(Expansion.Conjunction – wsj_0034)

Fig. 2. The nine discourse and non-discourse relations annotated on the excerpt in Figure 1. We underline connectives, and *italicize* Arg1s, and **bold** Arg2s. The last line of each example shows the relation sense and the file in the PDTB where the example is taken from. The relation sense is shown in the format of Level-1-class.Level-2-type.Level-3-subtype.

Table 1. *Level 1 classes and Level 2 types of the discourse relations in the PDTB. Level 3 subtypes are not shown.*

Temporal	Contingency	Comparison	Expansion
Synchrony	Cause	Contrast	Conjunction
Asynchronous	Pragmatic Cause	Pragmatic Contrast	Instantiation
	Condition	Concession	Restatement
	Pragmatic Condition	Pragmatic Concession	Alternative
			Exception
			List

Implicit, and AltLex relations are *discourse* relations, whereas EntRel and NoRel are *non-discourse* relations[3].

(10) *A record date hasn't been set.* **Bell, based in Los Angeles, makes and distributes electronic, computer and building products.**

(NoRel – wsj_0202)

For each discourse relation, the PDTB also provides annotation for the attribution (*i.e.*, the agent that expresses the argument) for Arg1, Arg2, and the relation as a whole. For example, the text span in the box in Example 11 – "declared San Francisco batting coach Dusty Baker after game two" – is the attribution span for the relation.

(11) *"I believe in the law of averages,"* declared San Francisco batting coach Dusty Baker after game two. Implicit = ACCORDINGLY **"I'd rather see a so-so hitter who's hot come up for the other side than a good hitter who's cold."**

(Contingency.Cause.Result – wsj_2202)

Aside from annotating all discourse and non-discourse relations, the annotators of the PDTB provided a three-level hierarchy of relation senses. The first level consists of four major relation *classes*: Temporal, Contingency, Comparison, and Expansion. "Temporal" is used when the events or situations in Arg1 and Arg2 are related temporally. The "Contingency" relation is used to mark when one argument causally influences the other. A "Comparison" relation results when the events in Arg1 and Arg2 are compared to highlight differences. "Expansion" relations result when the semantics/discourse of one argument is continued or expanded in the other argument,

For each class, a second level of 16 *types* is defined to provide finer semantic distinctions. They are listed in Table 1. For example, there are six types defined under the Expansion

[3] Entity transition is also important in capturing textual coherence. Another line of research in centering theory (Grosz *et al.* 1995; Barzilay and Lapata 2008) explores local coherence by modeling entity transition.

class: Conjunction, Instantiation, Restatement, Alternative, Exception, and List. A relation is labeled as "Restatement" when one argument reiterates the semantics of the other argument. In contrast, "Conjunction" is used when Arg2 provides additional information that is related to that in Arg1.

A third level of *subtypes* is defined to specify the semantic contribution of each argument. Nine out of the 16 types are refined into subtypes. For example, the sense labeled for Example 4 is Expansion.Restatement.Specification, meaning that there is a Restatement relation between Arg1 and Arg2, and Arg2 (instead of Arg1) is the argument that provides specific details. In this work, we follow our previous work (Lin *et al.* 2009) and focus on the Level 2 types, as we feel that Level 1 classes are too general and coarse-grained for downstream applications, while Level 3 subtypes are too fine-grained and are only provided for some types.

4 System Overview

We designed our parsing algorithm as a sequential pipeline to mimic the annotation procedure performed by the PDTB annotators. Figure 3 shows the pseudocode. The input to the parser is a free text T, whereas the output is the discourse structure of T in the PDTB style. The algorithm consists of three steps which sequentially label Explicit relations, Non-Explicit relations, and attribution spans. Non-Explicit relations include all relations that are not Explicit – *i.e.*, Implicit, AltLex, EntRel, and NoRel.

We first give a quick overview of the parser's three steps. The first step is to identify discourse connectives, label their Arg1 and Arg2 spans, and recognize their Explicit relation senses. The parser starts by identifying all connective occurrences in T (Line 2 in Figure 3), and labeling them as to whether they function as discourse connectives or not (Lines 3–4). When a connective occurrence C is determined to be a discourse connective, its Arg1 and Arg2 spans are then identified, and the parser classifies the tuple (C, Arg1, Arg2) into one of the Explicit relation senses (Lines 5–7). The second step then examines all adjacent sentence pairs within each paragraph. For each pair (S_i, S_j) that is not identified in any Explicit relation from Step 1, the parser then classifies the pair into EntRel, NoRel, or one of the Implicit/AltLex relation senses (Lines 12–15). Following the PDTB annotation convention, our parser also ignores inter-paragraph relations, *i.e.*, it ignores the adjacent sentence pair in between two paragraphs. In the third step, the parser first splits the text into clauses (Line 21), and for each clause U that appears in any discourse relations (*i.e.*, Explicit, Implicit, and AltLex relations; we term EntRel and NoRel as non-discourse relations), it checks whether U is an attribution span (Lines 22–24). In this step, the parser also follows the PDTB representation to only identify attribution spans appearing in discourse relations. It will not examine spans that are outside the text of the detected discourse relations.

In our work, we adopt a sequential pipeline to parse a text, instead of following a top-down or bottom-up approach that is common in determining the RST discourse trees (Marcu 1997). The reason for this design is twofold. First, the algorithm mimics the annotation procedure performed by the PDTB annotators, in which they labeled argument spans, identified relation senses, and annotated attribution spans in a step-by-step manner. Second, as the PDTB makes no commitments as to what kinds of high-level structures

Input: a text T
Output: a discourse structure of T

```
// Step 1: label Explicit relations
Identify all connective occurrences in T
for each connective occurrence C do
    Label C as disc-conn or non-disc-conn
    if C is disc-conn then
        Label Arg1 span and Arg2 span of C
        Label (C, Arg1, Arg2) as one of the Explicit relations
    end if
end for

// Step 2: label Non-Explicit: Implicit, AltLex, EntRel, and NoRel relations
for each paragraph P in T do
    for each adjacent sentence pair (S_i, S_j) in P do
        if (S_i, S_j) is not labeled as an Explicit relation in Step 1 then
            Label (S_i, S_j) as EntRel, NoRel, or one of the Implicit/AltLex relations
        end if
    end for
end for

// Step 3: label attribution spans
Split T into clauses
for each clause U do
    if U is in some Explicit/Implicit/AltLex relation from Step 1 or 2 then
        Label U as attr-span or non-attr-span
    end if
end for
```

Fig. 3. Pseudocode for the discourse parsing algorithm.

may be built up from the low-level units, we do not presume a tree-like structure or adopt a corresponding tree parsing algorithm. Note that this design allows a relation to be embedded within the argument of another relation, as well as an argument to be shared between two adjacent relations (*i.e.*, the Arg1 of the current relation is the Arg2 of the previous relation). For instance, Example 6 is embedded in the Arg2 of Example 7, and sentence 4 is shared between Example 3 and 4 as a common argument. Lee *et al.* (2006) provides detailed discussion on such types of relation dependencies.

The pipeline of the parser is shown in Figure 4, which consists of the connective classifier, argument labeler, explicit classifier, non-explicit classifier, and attribution span labeler. The first three components correspond to Step 1 in Figure 3, while the last two correspond

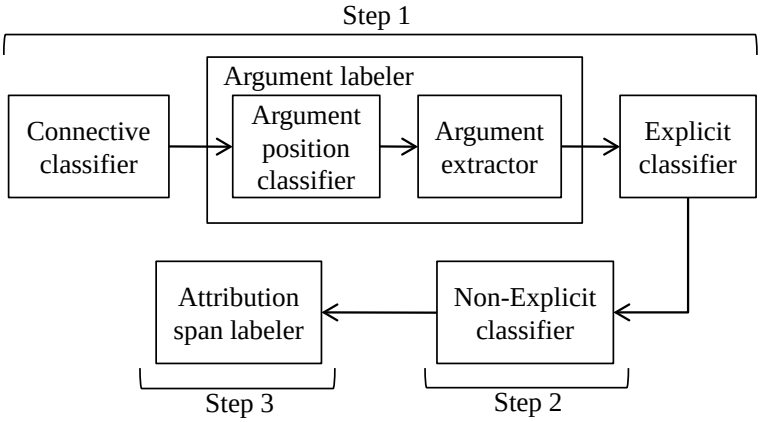

Fig. 4. System pipeline for the discourse parser.

to Steps 2 and 3, respectively. There are two sub-components in the argument labeler: an argument position classifier and an argument extractor. A detailed description of these components follows in the next section.

To illustrate the complete flow of the parsing algorithm, we look at how an ideal parser parses the excerpt of two paragraphs in Figure 1. In the first step, after comparing the text against the list of 100 discourse connectives defined in the PDTB, 10 connective occurrences are identified, which are italicized in Figure 1. The connective classifier then checks these occurrences and labels six of them as discourse connectives, as indicated by the underlines in Figure 1. The argument labeler follows by labeling the Arg1 and Arg2 spans for each discourse connective. In Example 6, the Arg1 and Arg2 spans for the connective "when" labeled by the argument labeler are "for ... 16%" and "the ... 13". The tuple (when, "for ... 16%", "the ... 13") is next propagated to the explicit classifier, which classifies the relation sense as Synchrony. In the second step, the parser examines all adjacent sentence pairs within paragraphs 1 and 2 separately, *i.e.*, the inter-paragraph sentence pair (S_3, S_4) is exempt from checking. Since there are Explicit relations already assigned to the pairs (S_1, S_2) in Paragraph 1 and (S_6, S_7) and (S_7, S_8) in Paragraph 2 (see Example 1, 7, and 9), they are also exempt from further classification. The non-explicit classifier then classifies the remaining pairs (S_2, S_3), (S_4, S_5) and (S_5, S_6) as EntRel, Implicit, and AltLex relations, respectively, as shown in Example 2, 4, and 5. In the last step, the attribution span labeler will examine all discourse relations to label their attribution spans. As there are no such spans in Figure 1, the reader is referred to Example 11 for a sentence with an attribution span that would be labeled.

We now refine this overview by detailing the individual components' structure and processing workflow.

5 Components

Our system takes a fully data-driven, supervised learning approach. As such, the annotated data is processed into binary feature vectors that are suitable input to a maximum entropy learning model. In the following descriptions, we describe the component designs as well as the derived feature classes.

5.1 Connective Classifier

There are 100 types of discourse connectives defined in the PDTB. Given a connective occurrence such as "when", the parser needs to decide whether it is functioning as a discourse connective. To illustrate, compare the use of the connective "and" in Example 1 (*i.e.* Sentence 2) and in Sentence 5 of Figure 1. In Example 1, "And" is functioning as a discourse connective to join two discourse events in Arg1 and Arg2, thus the annotators labeled the sense as Expansion.Conjunction. On the other hand, the "and" in Sentence 5 is used to link "Political" and "currency" in a noun phrase, which is not an example of discourse use. In the entire PDTB corpus, words and phrases annotated as discourse connectives (100 types) constitute only 29.65% of all their occurrences, with the remaining 70% not functioning as discourse connectives. Thus, it is crucial to disambiguate the connectives before sending them down the pipeline to label their argument spans and relation senses.

Pitler and Nenkova (2009) showed that syntactic features extracted from constituent parse trees are very useful in disambiguating discourse connectives. Beside the connective itself as a feature, they applied other syntactic features: the highest node in the tree that covers only the connective words (which they termed *self category*), the parent, left and right siblings of the self category, and two binary features that check whether the right sibling contains a VP and/or a trace. The best feature set they demonstrated also included pairwise interaction features between the connective and each syntactic feature, and interaction features between pairs of syntactic features.

In addition to the above, we observed that a connective's context and part-of-speech (POS) give a very strong indication of its discourse usage. For example, the connective "after" usually functions as a discourse connective when it is followed by a present participle, as in "after rising 3.9%". The syntactic parse path from the connective to the root of the tree models how it is syntactically connected to the sentence as a whole, reflecting its functionality within the sentence. Based on these observations, we propose a set of lexico-syntactic and path feature classes for a connective C with its previous word $prev$ and next word $next$:

- C POS
- $prev$ + C
- $prev$ POS
- $prev$ POS + C POS
- C + $next$
- $next$ POS
- C POS + $next$ POS
- path of C's parent $\rightarrow$ root
- compressed path of C's parent $\rightarrow$ root

Input: a discourse connective C and the text T
Output: Arg1 and Arg2 spans of C

```
1: // Argument position classifier
2: Classify the relative position of Arg1 as SS or PS
3:
4: // Argument extractor
5: if the relative position of Arg1 is SS then
6:     Identify the Arg1 and Arg2 subtree nodes within the sentence parse tree
7:     Apply tree subtraction to extract the Arg1 and Arg2 spans
8: else // the relative position of Arg1 is PS
9:     Label the sentence containing C as Arg2
10:    Identify and label the Arg1 sentence from all previous sentences of Arg2
11: end if
```

Fig. 5. Pseudocode for the argument labeler, which corresponds to Line 6 in Figure 3.

Each of the above lines represents a feature class. The first seven feature classes model the connective's context and POS, while the last two are the path from C to the root and the compressed path where adjacent identical tags are combined (*e.g.*, -VP-VP- is combined into -VP-). Our path feature class is novel in the way it models the syntactic relation between the connective under consideration and the syntactic root. Appendix A.1 lists the features to disambiguate the connective "after" in Example 20, whose constituent parse tree is shown in Figure 10. Appendix A uses the Explicit relation in Example 20 to illustrate the features extracted for the classifiers in Step 1.

5.2 Argument Labeler

The parser now labels the Arg1 and Arg2 spans of every connective labeled in the previous step as a discourse connective, in two steps: (1) identifying the locations of Arg1 and Arg2, and (2) labeling the spans. We note that Arg2 is the argument with which the connective is syntactically associated, and thus its position is fixed once we locate the connective. The remaining problem of the first step is in identifying the location of Arg1. We implement this as a classification task to recognize the *relative position* of Arg1, with respect to the connective (Line 2 in Figure 5). According to the different relative positions of Arg1, the argument extractor then attempts to extract the Arg1 and Arg2 spans in the second step (Line 5 – 11 in Figure 5). Figure 5 gives the pseudocode for the argument labeler, which corresponds to Line 6 in Figure 3 and is further discussed in the following.

5.2.1 Argument Position Classifier

Prasad *et al.* (2008) described the breakdown of the positions of Arg1 in their study of the PDTB annotations. They showed that Arg1 can be located within the same sentence

as the connective (SS), in some previous sentence of the connective (PS), or in some sentence following the sentence containing the connective (FS). When Arg1 is located in some previous sentence, it can either be in the immediately previous sentence of the connective (IPS), or in some non-adjacent previous sentence of the connective (NAPS). Example 8 is a relation where the arguments and connective appear in the same sentence, while Example 1 shows a case in which Arg2 immediately follows Arg1. The distribution from Prasad *et al.* (2008) shows that 60.9% of the Explicit relations are SS, 39.1% are PS, and less than 0.1% are FS (only 8 instances).

Motivated by this observation, we design an argument position classifier to identify the relative position of Arg1 as SS or PS. We ignore FS since there are too few training instances. We notice that the connective string itself is a very good feature. For example, when the connective token is "And" (*i.e.*, "and" with its first letter capitalized, as in Example 1), it is a continuation from the previous sentence and thus Arg1 is likely in PS; whereas when the connective token is lowercase "and", Arg1 is likely the clause at the left hand side of "and" and thus it is in SS (Example 8). Furthermore, some connectives always take a particular position. For example, "when" always indicates an SS case, whereas "additionally" always indicates PS.

Aside from the connective string, we also use the contextual feature classes in the classifier for the connective C with its first and second previous words $prev_1$ and $prev_2$. The list below gives the feature classes used in our supervised classifier.

- C string
- position of 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

After the relative position of Arg1 is identified, the result is propagated to the argument extractor, which employs different strategies to extract the Arg1 and Arg2 spans, depending on whether the result is SS or PS.

5.2.2 Argument Extractor

When the relative position of Arg1 is classified as PS from the previous stage, Arg1 is located in one of the previous sentences of the connective, while Arg2 is in the same sentence as the connective. A majority classifier labels the immediately previous sentence as Arg1, which already gives an F_1 of 76.90% under the gold standard setting on the entire PDTB. In this paper, we focus on extracting the argument spans for the SS case and do not focus on identifying the Arg1 sentences for the PS case. As such, we employ the majority

classifier as our classifier for the PS case. Next, we describe our approach to extract the arguments for the SS case in detail.

When Arg1 is classified as in the same sentence (SS), this means that Arg1, Arg2, and the connective itself are in the same sentence. This can be further divided into four situations depending on the overlap and position of the two arguments in the sentence:

1. Arg1 precedes Arg2,
2. Arg2 precedes Arg1,
3. Arg2 is embedded within Arg1, or
4. Arg1 is embedded within Arg2.

These four situations are illustrated by Examples 8, 6, 12, and 13, respectively. One possible approach is to split the sentence into clauses before deciding which clause is Arg1 or Arg2. The problem with this approach is that it is not able to recognize the last two cases, where one argument divides the other into two parts. Another challenge is to exclude the text spans that are not in the relation, such as the span "It's the ... American" in Example 12.

(12) It's the petulant complaint of an impudent American *whom Sony hosted for a year* while **he was on a Luce Fellowship in Tokyo** – *to the regret of both parties.*
(Temporal.Synchrony – wsj_0037)

(13) **The prime minister,** *whose hair is thinning and gray and whose face has a perpetual pallor,* nonetheless **continues to display an energy, a precision of thought and a willingness to say publicly what most other Asian leaders dare say only privately.**
(Comparison.Concession.Contra-expectation – wsj_0296)

Dinesh *et al.* (2005) showed that Arg1 and Arg2 in the same sentence for subordinating connectives are always syntactically related as shown in Figure 6(a), where Arg1 and Arg2 nodes are the lowest nodes that cover the respective spans. They demonstrated that a rule-based algorithm is capable of extracting Arg1 and Arg2 in such cases for subordinating connectives. By using tree subtraction, the third case mentioned above can be easily recognized and the text spans that are not in the relation can be excluded. In Figure 6(a), Span 3 is labeled as Arg2 that divides Arg1 into two non-continuous Spans 2 and 4. The out-of-relation spans (Spans 1 and 5) are also excluded by subtracting the subtree root at the Arg1 node from the entire tree starting from the Root.

However, dealing with only the subordinating connectives is not sufficient. Subordinating connectives only take up 40.93% for the SS cases; the percentages of coordinating connectives and discourse adverbials in the whole PDTB for SS cases are 37.50% and 21.57%, respectively. We observe that coordinating connectives ("and", "or", "but", etc.) usually constrain Arg1 and Arg2 to be syntactically related in one of two ways as shown in Figure 6(b)-(c), where CC is the connective POS. Example 14 and Figure 7 give an example to illustrate Figure 6(c). Discourse adverbials do not demonstrate such syntactic constraints as strongly as subordinating and coordinating connectives do, but their Arg1 and Arg2 are also syntactically bound by the positions of and path between the two argument nodes. For example, Figure 8 shows the syntactic relation of the Arg1 and Arg2 nodes for the discourse adverbial "still" in Example 15. Furthermore, the rule-based algorithm in (Dinesh *et al.* 2005) does not recognize the fourth case where the Arg1 span

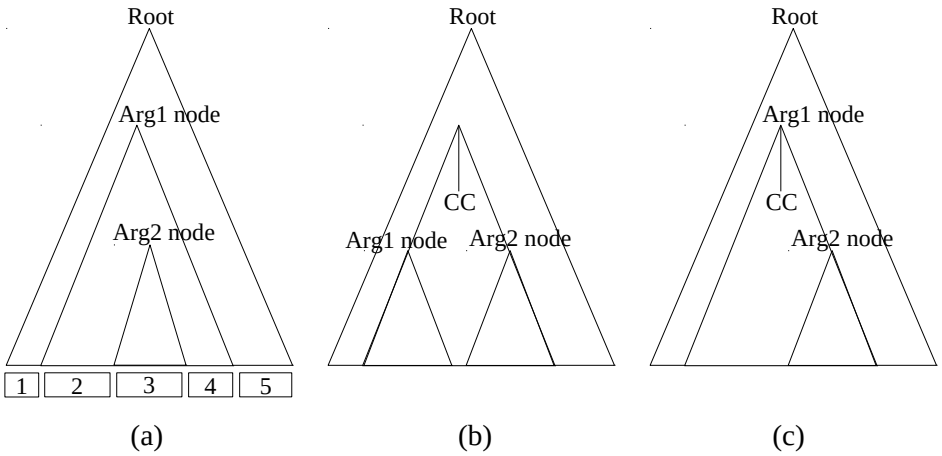

Fig. 6. Syntactic relations of Arg1 and Arg2 subtree nodes in the parse tree. (a): Arg2 contains span 3 that divides Arg1 into two spans 2 and 4. (b)-(c): two syntactic relations of Arg1 and Arg2 for coordinating connectives.

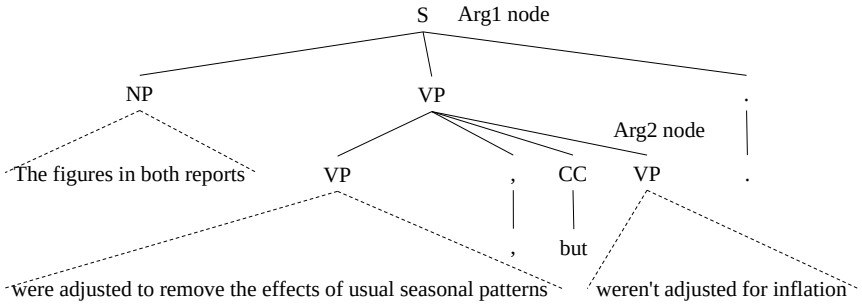

Fig. 7. The parse tree for Example 14 to illustrate Figure 6(c).

is embedded within Arg2. The ratio of third case to fourth case occurrences in the entire PDTB corpus is approximately 1:1. Thus we believe that the fourth case also needs to be taken care of.

(14) *The figures in both reports were adjusted to remove the effects of usual seasonal patterns,* but **weren't adjusted for inflation.**

(Comparison.Contrast – wsj_0036)

(15) The ultimate result came in Hymowitz v. Lilly, where the highest New York court expanded the market-share approach for the first time to say *that drug makers that could prove Mindy Hymowitz's mother didn't use their pill* **must** still **pay their share of any damages**.

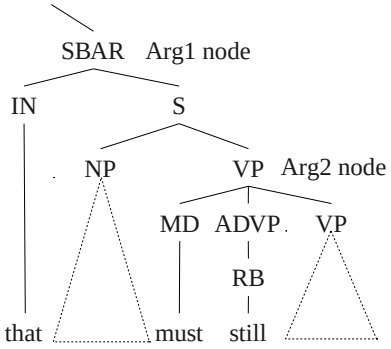

Fig. 8. Part of the parse tree for Example 15 with Arg1 and Arg2 nodes labeled.

(Comparison.Concession.Contra-expectation – wsj_0130)

Given these observations, we design an automatic argument node identifier to first identify the Arg1 and Arg2 subtree nodes within the sentence parse tree for all subordinating connectives, coordinating connectives, and discourse adverbials; and then apply tree subtraction to extract the Arg1 and Arg2 spans. The argument node identifier labels each internal node (except the part-of-speech node) of the tree with three probabilities: functioning as Arg1-node, Arg2-node, and None. The internal node with the highest Arg1-node probability is chosen as the Arg1 node, and likewise for the Arg2 node. If the Arg1 node is the ancestor of the Arg2 node, the subtree under the Arg2 node is then subtracted from the Arg1 subtree to obtain the Arg1 span, and conversely when the Arg2 node is the ancestor of the Arg1 node. Motivated by the syntactic properties observed, we use the following feature classes for the node N under consideration with regard to the connective C:

- C string
- C's syntactic category: subordinating, coordinating, or discourse adverbial
- number of left siblings of C
- number of right siblings of C
- the path P of C's parent $\rightarrow N$
- the path P and whether the number of C's left sibling is greater than one
- the relative position of N to C: left, middle, or right

The syntactic category (subordinating, coordinating, or discourse adverbial) of the connective is a useful clue of the locations of the Arg1 and Arg2 nodes. We obtain the corresponding categories for the connectives from the list provided in (Knott 1996). Appendix C shows the list of discourse connectives and their syntactic categories from Knott's thesis. The path from C's parent node to the node N under consideration is also an informative feature, as it reflects how N is related to C syntactically. The following are two paths for the actual Arg2 node and the MD node in Figure 8:

RB $\uparrow$ ADVP $\uparrow$ VP
RB $\uparrow$ ADVP $\uparrow$ VP $\downarrow$ MD

The relative position of N to C is medial when N is on the path of C to root; it can also be left or right depending on whether it is located on the left- or right-hand side of this path. This feature also models the syntactic relation of C and N to some extent. To label each internal node with three probabilities, we adopt a maximum entropy classifier, as it is capable of estimating class probabilities.

To illustrate how the argument position classifier and argument extractor work together to label the arguments, let us look at Example 12. After examining the features for the connective "while", the argument position classifier will decide that Arg1 and Arg2 are in the same sentence (SS), and pass it to the argument extractor. Since the class is SS, the argument extractor invokes the argument node identifier to locate the internal node that covers the Arg1 span (*i.e.*, "whom Sony ... both parties") and that covers Arg2 and the connective (*i.e.*, "while he ... in Tokyo"). Finally, tree subtraction is applied to clean up and remove the Arg2 span from the Arg1 span.

5.3 Explicit Classifier

After identifying a discourse connective and its two arguments, the next step is to decide what Explicit relation it conveys. It is important to disambiguate the relation sense of the connective, as the same connective may carry different semantics under different contexts. For example, the connective "and" has different senses of Expansion.Conjunction and Expansion.List in Example 8 and 16, respectively.

(16) *Microsoft added 2 1/8 to 81 3/4* and **Oracle Systems rose 1 1/2 to 23 1/4**.
(Expansion.List – wsj_0327)

Prasad *et al.* (2008) reported a human agreement of 94% on Level 1 classes and 84% on Level 2 types for Explicit relations over the whole PDTB corpus. The connective itself is a very good feature, as only a few connectives are ambiguous as pointed out in (Pitler *et al.* 2008), and the distribution of the majority of the ambiguous connectives is highly skewed toward certain senses. We train an explicit classifier using three types of feature classes of the connective C and its previous word $prev$:

- C string
- C's POS
- C + $prev$

We follow our previous work (Lin *et al.* 2009) to train and test on the 16 Level 2 types.

5.4 Non-Explicit Classifier

The PDTB also provides annotation for Implicit relations, AltLex relations, entity transition (EntRel), and otherwise no relation (NoRel), which are lumped together as Non-Explicit relations. The Non-Explicit relations are annotated for all adjacent sentence pairs within paragraphs. If there is already an Explicit relation from the previous step between two adjacent sentences, they are exempt from further examination.

Similar to the explicit classifier, we adopt the Level 2 types for the Implicit and AltLex

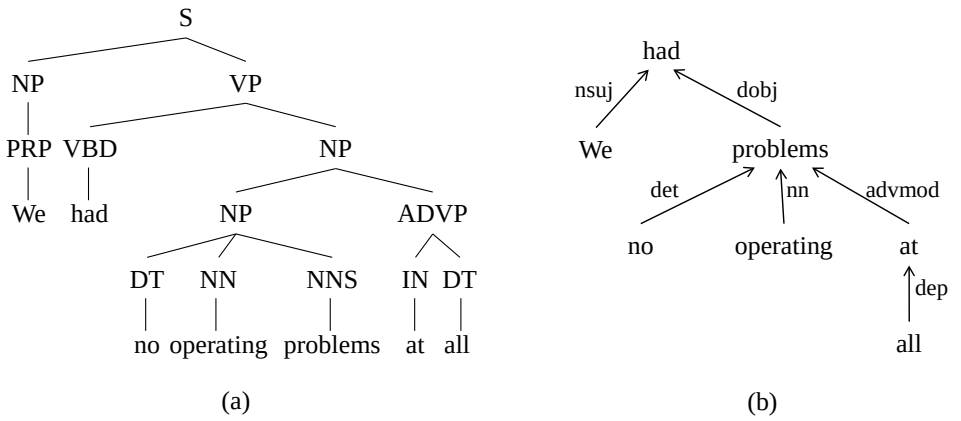

Fig. 9. A constituent subtree (a) and a dependency subtree (b) for Arg1 of an implicit relation from wsj_2224.

relations. As there are too few training instances for Condition, Pragmatic Condition, Pragmatic Contrast, Pragmatic Concession, and Exception relations (in total only 9 training instances), these five types are removed, resulting in 11 Level 2 types. Thus, our Non-Explicit classifier assigns candidate sentence pairs to one of 13 types (11 Level 2 types plus EntRel and NoRel). We apply the three feature classes from our previous work (Lin *et al.* 2009):

- constituent parse features
- dependency parse features
- word-pair features

Constituent parse and dependency parse features include production rules and dependency rules from the parse trees of the arguments. From our observation of the PDTB relations, the syntactic structure within one argument may constrain the relation sense and the syntactic structure of the other argument. For Arg1 and Arg2 of each relation, we extract the corresponding constituent and dependency parses. As an argument can be a single sentence or a clause, this results in a whole constituent/dependency tree or parts of a tree. From these parses, we extract all possible production rules and dependency rules, and represent each rule as three binary features to check whether it appears in Arg1, Arg2, or both arguments. For instance, the production rules for the subtree in Figure 9(a) are S $\rightarrow$ NP VP, NP $\rightarrow$ PRP, PRP $\rightarrow$ "We", etc.; the dependency rules for the dependency subtree in Figure 9(b) are "had" $\leftarrow$ nsubj dobj, "problems" $\leftarrow$ det nn advmod, "at" $\leftarrow$ dep. Word-pair features are word pairs in which one word is extracted from Arg1 and the other from Arg2, i.e., all (w_i, wj) where w_i is a word from Arg1 and w_j a word from Arg2.

AltLex relations are very similar to their counterpart Explicit relations, except that they are *alternatively lexicalized* by some non-connective expressions, instead of being expressed by one of the 100 PDTB pre-defined discourse connectives. Such non-connective

expressions are usually attached to the beginning of Arg2 (*e.g.*, such as "Another concern" in Arg2 of Example 5). To distinguish AltLex relations, we use three feature classes that represent the first three stemmed terms of Arg2. For the example above, the features that are turned on will be $term_1$=*another*, $term_2$=*concern*, and $term_3$=*the*.

5.5 Attribution Span Labeler

For each discourse relation (*i.e.*, Explicit, Implicit, or AltLex relation), the PDTB annotators labeled the attribution spans and annotated four dimensions for Arg1, Arg2, and the relation: their sources, types, scopal polarities, and determinacy. For the current parser, we label the attribution spans without labeling the four attribution dimensions and direction (Arg1, Arg2, or the relation) it is associated with. The reason is that our focus of study is the attribution location and span, and recognizing these four dimensions and attribution direction will lead to building another set of classifiers which are outside of our study. Note that we label attribution spans that appear within discourse relations.

The attribution span labeler consists of two steps: splitting the text into clauses, and deciding which clauses are attribution spans. In the first step, we employ a lightweight clause splitter that we have developed which uses a syntactically motivated approach similar to (Skadhauge and Hardt 2005). This clause splitter makes use of punctuation symbols and syntactic structures of SBAR complements.

The attribution span labeler then classifies each clause into attr-span or non-attr-span. Words (especially verbs) in the clause are a very good clue to decide whether it is an attribution. Examples are the verbs "declared" and "say" in Example 11 and 15. Another useful clue is by looking at the end of the previous clause and the start of the next one. In Example 11, which is partially replicated below, the previous clause ends with a comma and a closing quotation mark, and the next clause starts with an opening quotation mark, which suggest that the previous and next clauses are in the same speech act and the current clause is probably the attribution of the speech.

. . . averages," declared San Francisco batting coach Dusty Baker after game two. "I'd . . .

Based on these observations, we propose the following feature classes which are extracted from the current, previous, and next clauses ($curr$, $prev$, and $next$):

- lowercased and lemmatized verbs in $curr$
- the first and last terms 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 $next$
- the position of $curr$ in the sentence: start, middle, end, or whole sentence
- production rules extracted from $curr$

Appendix B shows features extracted for the above example. Some clauses that belong to single attribution spans may be incorrectly split into more than one clause by the clause splitter. For example, "said C. Bruce Johnstone, who runs Fidelity Investments' $5 billion

Equity-Income Fund." is annotated as a single attribution span in the PDTB. It is (mistakenly) split into two clauses "said C. Bruce Johnstone," and "who runs Fidelity Investments' $5 billion Equity-Income Fund." by the clause splitter, and then both classified as attr-span. To correct such mistakes, adjacent attribution clauses within a sentence are combined to form a single attribution span after classification.

6 Evaluation

In all of our experiments, we follow the recommendation from (PDTB-Group 2007) to use Sections 02–21 in the PDTB for training, Section 22 for development, and Section 23 for testing. All classifiers are trained with the OpenNLP maximum entropy package[4] without smoothing and with 100 iterations.

For each component, the experiments are carried out when there is no error propagated from the previous components (*i.e.*, using gold standard annotation for the previous components), and also when there is error propagation. As the PDTB was aligned with the PTB, we can either directly use the gold standard parse trees and sentence boundaries from the PTB files, or we can apply an automatic parser and sentence splitter. The experiments are carried out under three settings for each component:

1. GS + no EP: using gold standard (GS) parses and sentence boundaries without error propagation (EP)
2. GS + EP: using GS with EP
3. Auto + EP: using both automatic parsing and sentence splitting (Auto) with EP.

Thus, GS + no EP corresponds to a clean, per-component evaluation, whereas the Auto + EP setting assesses end-to-end fully automated performance (as would be expected on new, unseen text input). We use the NIST text segmenter[5] to insert sentence boundaries and the Charniak parser[6] to parse the sentences in the Auto setting. As there are no gold standard dependency parses for the PTB files, we employ the Stanford dependency parser[7] in both GS and Auto settings.

The main focus of this work is designing an end-to-end discourse parser joined by all components and not on improving a specific component. As such, we only reimplement (Pitler and Nenkova 2009)'s system to compare with our connective classifier. For the other components, we do not reimplement other systems mentioned in the related work section.

6.1 Results for Connective Classifier

On the connective classifier task, Pitler and Nenkova (2009) (hereafter, P&N) reported an accuracy of 96.26% and F_1 of 94.19% with a 10-fold cross validation (CV) on Sections 02–22. To compare with P&N, we also run a 10-fold CV on Sections 02–22 using their features and obtain replicated accuracy of 96.09% and replicated F_1 of 93.57%. Adding in

[4] `http://maxent.sourceforge.net/`
[5] `http://duc.nist.gov/duc2004/software/duc2003.breakSent.tar.gz`
[6] `ftp://ftp.cs.brown.edu/pub/nlparser/`
[7] `http://nlp.stanford.edu/software/lex-parser.shtml`

Table 2. *Results for the connective classifier. No EP as this is the first component in the pipeline.*

	P&N		+new	
	Acc.	F_1	Acc.	F_1
GS	95.30	92.75	97.34	95.76
Auto	94.21	91.00	96.02	93.62

our lexico-syntactic and path features, the performance is increased to 97.25% accuracy and 95.36% F_1, yielding improvements of 0.99% and 1.17% over the reported results and 1.16% and 1.79% over the replicated results. A paired t-test shows that the improvements over our replication of P&N's results are significant with $p < 0.001$[8].

In Table 2, we report results from the connective classifiers trained on Sections 02–21 and tested on Section 23. As there is no error propagated into the connective classifier since it is the first component, we report results for just the GS and Auto settings. The second and third columns show the accuracy and F_1 using the features of P&N, whereas the last two columns show the results when we add in the lexico-syntactic and path features (+new). Introducing the new features significantly (all with $p < 0.001$) increases the accuracy and F_1 by 2.04% and 3.01% under the GS setting, and 1.81% and 2.62% under Auto. This confirms the usefulness of integrating the contextual and syntactic information. As the connective classifier is the first component in the pipeline, good performance is crucial to mitigate the effect of cascaded errors downstream.

When we look into the incorrectly labeled connectives, we find that the connective with the highest number of incorrect labels is "and" (8 false negatives and 4 false positives for the GS setting). This is not surprising, as "and" is always regarded as an ambiguous connective. A solution to this problem is to separately train one model for each highly ambiguous connective and train another generic model to identify the remaining connectives.

6.2 Results for Argument Labeler

We next perform evaluation on the argument position classifier, and report micro precision, recall, and F_1, as well as the per-class F_1, in Table 3. The GS + no EP setting gives a high F_1 of 97.94%, which drops 3.59% and another 2.26% when error propagation and full automation are added in. The per-class F_1 shows that the performance degradation is mostly due to the SS class (Arg1 and Arg2 in the Same Sentence): the drops for SS are 5.36% and 3.35%, compared to 1.07% and 0.68% for PS. When we look into the contingency table

[8] It is not possible to conduct paired t-test on the reported results for P&N as we do not have their predictions.

Table 3. *Results for the argument position classifier.*

	Prec.	Recall	F_1	Per-class F_1 SS	Per-class F_1 PS
GS + no EP	97.94	97.94	97.94	98.26	97.49
GS + EP	94.66	94.04	94.35	92.90	96.42
Auto + EP	92.75	91.44	92.09	89.55	95.74

Table 4. *Results for identifying the Arg1 and Arg2 subtree nodes for the SS case under the GS + no EP setting for the three categories.*

	Arg1 F_1	Arg2 F_1	Arg1&Arg2 F_1
Subordinating	88.46	97.93	86.98
Coordinating	90.34	90.34	82.39
Discourse adverbial	46.88	62.50	37.50
All	86.63	93.41	82.60

for the GS + EP setting, we notice that out of the 36 false positives propagated from the connective classifier, 30 of them are classified as SS; for the Auto + EP setting, there are 46 out of 52 classified as SS. This shows that the difference in the performance drops for SS and PS is largely due to error propagation from the connective classifier, and not to the classes themselves. Although not strictly comparable, these results are consistent with the results in (Prasad *et al.* 2010), which reported an accuracy of 93% on classifying discourse adverbials into intra- and inter-sentential. Three feature classes were used in (Prasad *et al.* 2010): connective head, connective position, and syntactic path from the connective to the root of the sentence.

We next evaluate the performance of the argument extractor. Table 4 illustrates the results of identifying the Arg1 and Arg2 subtree nodes for the SS case for the three connective categories. The last column shows the Arg1&Arg2 F_1 which requires both Arg1 and Arg2 nodes to be identified correctly. We only show the results for the GS + no EP setting. As expected, Arg1 and Arg2 nodes for subordinating connectives are the easiest ones to identify and give a high Arg2 F_1 of 97.93% and a Arg1&Arg2 level F_1 of 86.98%. We note that the Arg1 F_1 and Arg2 F_1 for coordinating connectives are the same, which is unexpected, as we expect Arg2 nodes to be easier to classify since Arg2 and the connective are syntactically associated. Error analysis shows that Arg2 spans for coordinating connectives tend to include extra text that causes the Arg2 nodes to move lower down in the parse tree.

Table 5. *Overall results for the argument extractor.*

		Arg1 F_1	Arg2 F_1	Arg1&Arg2 F_1
Partial	GS + no EP	86.67	99.13	86.24
	GS + EP	83.62	94.98	83.52
	Auto + EP	81.72	92.64	80.96
Exact	GS + no EP	59.15	82.23	53.85
	GS + EP	57.64	79.80	52.29
	Auto + EP	47.68	70.27	40.37

For example, "... and Mr. Simpson said he resigned in 1988" contains the extra span "Mr. Simpson said" which causes the Arg2 node (which covers "he resigned in 1988") moving two levels down the tree. The system erroneously labels "Mr. Simpson ... 1988" as Arg2.

Also as we discussed, discourse adverbials are difficult to identify as their Arg1 and Arg2 nodes are not strongly bound in the parse trees. However, as they do not take up a large percentage in the test data (only 5.38% of the test data is for identifying Arg1 and Arg2 nodes for discourse adverbials under the GS + no EP setting), they do not lead to a large degradation as shown in the last row of the overall performance of the three categories.

Miltsakaki *et al.* (2004) reported human agreements on both exact and partial matches to be 90.2% and 94.5%, respectively. We follow this work and report both exact and partial matches. When checking exact match, we require two spans to match identically, excluding any leading and ending punctuation symbols. A partial match is credited if there is any overlap between the verbs and nouns of the two spans. The results for the overall performance for both SS and PS cases are shown in Table 5. The GS + no EP setting gives a satisfactory F_1 of 86.24% for partial matching on Arg1&Arg2 F_1. On the other hand, the results for exact matching are much lower than the human agreement. In Miltsakaki *et al.*'s work, most disagreements for exact match were reported to come from partial overlaps which do not show significant semantic difference. Similarly, in our analysis, we observed that most misses are due to small portions of text being deleted from or added to the spans by the annotators to follow the *minimality principle*. The minimality principle states that the annotation should include in the argument the minimal span of text that is sufficient for the interpretation of the relation. This requires deep semantic analysis and poses difficulties for machines to follow.

6.3 Results for Explicit Classifier

Following the pipeline, we then evaluate the explicit classifier, with its performance shown in Table 6. Recall that human agreement on Level 2 types is 84.00% and a baseline clas-

Table 6. *Results for the explicit classifier.*

	Precision	Recall	F_1
GS + no EP	86.77	86.77	86.77
GS + EP	83.19	82.65	82.92
Auto + EP	81.19	80.04	80.61

Table 7. *Results for the non-explicit classifier.*

	Precision	Recall	F_1	Baseline F_1
GS + no EP	39.63	39.63	39.63	21.34
GS + EP	26.21	27.63	26.90	20.30
Auto + EP	24.54	26.45	25.46	19.31

sifier that uses only the connectives as features already yields an F_1 of 86.00% under the GS + no EP setting on Section 23. Adding our new features improves F_1 to 86.77% (but which is not a statistically significant improvement). With full automation and error propagation, we obtain an F_1 of 80.61%. Pitler and Nenkova (2009) show that using the same syntactic features as their connective classifier is able to improve the explicit classifier on a 10-fold cross validation on Sections 02-22. We have trained the classifier on Sections 02-21 using their features and tested on Section 23, but it actually performs worse than the baseline. Therefore we do not include their features in the explicit classifier.

6.4 Results for Non-Explicit Classifier

For the non-explicit classifier, a majority class baseline that labels all instances as EntRel yields an F_1 in the low 20s, as shown in the last column of Table 7. The percentage of EntRel is slightly higher than the most frequent implicit Cause relations (21.34% vs. 21.24% in the implicit relations). A single component evaluation (GS + no EP) shows a micro F_1 of 39.63%. Although the F_1 scores for the GS + EP and Auto + EP settings are unsatisfactory, they still significantly ($p < 0.01$) outperform the majority class baseline by about 6%. This performance is in line with the difficulties of classifying Implicit relations discussed in detail in our previous work (Lin *et al.* 2009).

Table 8. *Results for the attribution span labeler.*

		Precision	Recall	F_1
Partial	GS + no EP	79.40	79.96	79.68
	GS + EP	65.93	79.96	72.27
	Auto + EP	64.40	51.68	57.34
Exact	GS + no EP	65.72	66.19	65.95
	GS + EP	54.57	66.19	59.82
	Auto + EP	47.83	38.39	42.59

Table 9. *Overall performance for both Explicit and Non-Explicit relations. GS + no EP setting is not included, as this is not a component-wise evaluation.*

		F_1
Partial	GS + EP	46.80
	Auto + EP	38.18
Exact	GS + EP	33.00
	Auto + EP	20.64

6.5 Results for Attribution Span Labeler

The final component, the attribution span labeler, is evaluated under both partial and exact match, similar to the argument extractor. From Table 8, we see that the GS + no EP setting achieves F_1 scores of 79.68% and 65.95% for partial and exact match, respectively. When error propagation is introduced, the degradation of F_1 is largely due to the drop in precision. This is not surprising as at this point, the test data contains a number of false positives propagated from the previous components. This has an effect on the precision calculation but not recall (the recall scores do not change). When full automation is further added, the degradation is largely due to the drop in recall. This is because the automatic parser introduces noise that causes errors in the clause splitting step.

6.6 Overall Performance

To evaluate the whole pipeline, we look at the Explicit and Non-Explicit relations that are correctly identified. We define a relation as correct if its relation sense is classified

correctly, and both its Arg1 and Arg2 are partially or exactly matched. The overall performance is shown in Table 9. Under partial matching, the GS + EP setting gives an overall system F_1 of 46.80%, while under exact matching, it achieves an F_1 of 33.00%. Auto + EP gives 38.18% F_1 for partial match and 20.64% F_1 for exact match. A large portion of the misses come from the Non-Explicit relations, as they are more difficult to classify in comparison to the Explicit relations. The GS + EP F_1 is close to the system F_1 of 44.3% of an RST parser reported in (duVerle and Prendinger 2009).

7 Discussion and Future Work

The overall performance of the whole pipeline shows that the non-explicit classifier generates a large portion of the errors, which suggests that there is still much room for improvement in that component. In our previous work (Lin *et al.* 2009) on classifying Implicit relations, we have shown that the difficulties of this task are mostly attributed to four types of challenges: the ambiguity among the relation senses, the need for using inference and a knowledge base, the analysis of the contextual information in understanding the arguments, and access to world knowledge. We plan to tackle some of these challenges in the non-explicit classifier in our future work. For example, we may extract information from external resources such as WordNet (Miller 1995) and Wikipedia to incorporate world knowledge into the component.

In our explicit classifier, although the tuple (C, Arg1, Arg2) is passed into the classifier, the current approach does not make use of information from Arg1 and Arg2. One future direction is to extract informative features from these two arguments for the explicit classifier. The current approach also does not deal with identifying Arg1 from all previous sentences for the PS case. Although about 77% of Arg1s can be located in the immediately previous sentence of Arg2s in this case, it is important to take the rest into consideration to make this component complete. Furthermore, this task will not be easy, as there is no restriction on the distance between Arg1 and Arg2. Example 17 shows a situation where Arg1 is located four sentences away from Arg2. Our next step is to design a PS identifier and integrate it into the current pipeline. One possibility is to follow (Prasad *et al.* 2010) to use a set of filters and heuristics to locate the positions of Arg1 spans.

(17) *GOODY PRODUCTS Inc. cut its quarterly dividend to five cents a share from 11.5 cents a share.* The reduced dividend is payable Jan. 2 to stock of record Dec. 15. The Kearny, N.J.-based maker of hair accessories and other cosmetic products said it cut the dividend due to its third-quarter loss of $992,000, or 15 cents a share. In the year-ago quarter, the company reported net income of $1.9 million, or 29 cents a share. **The company** <u>also</u> **adopted an anti-takeover plan.**

(Expansion.Conjunction – wsj_0068)

The PDTB provides annotations for the direction of the attributions to indicate whether an attribution is pointing to Arg1, Arg2, or the relation as a whole. In Example 18, the attribution "traders said" points to Arg1, while the second attribution "Disney ... said" points to the whole relation.

(18) *But then it shot upward 7 1/2 as Goldman, Sachs & Co. stepped in and bought*,

traders said. However, Disney specialist Robert Fagenson said: **"I would be surprised if Goldman represented 4% of the opening volume."**

(Comparison – wsj_2232)

Such information can also be incorporated into the parser, as this provides finer grained information on the opinions and the opinion holders, which is useful for downstream subjectivity analysis. The current attribution span labeler only considers clauses within the relation, which may result in missing clauses that are attribution spans of the relation but reside outside the relation. For instance, in Example 19, the attribution span "said David ... Sunday's go" for the relation "*I'm for ...* **lost yesterday**" resides outside the relation itself, thus it will not be examined by our system. One possible approach to solve this problem is to use a window of sentences to check previous and following sentences for attributions that are pointing to this relation.

(19) "*I'm for the Giants today*, **but only** because **they lost yesterday**.
I love 'em both. The only thing I'm rooting for is for the Series to go seven games,"
said David Williams, a Sacramento septuagenarian, at the Coliseum before
Sunday's go.

(Contingency.Cause.Reason – wsj_2202)

Wellner (2009) pointed out that verbs from the attribution spans are useful features in identifying the argument head words. In his work, Wellner checked whether the argument verb (as only argument verbs, not argument spans, are identified) is a potentially attribution-denoting verb. This suggests that we can feed the results from the attribution span labeler back into the argument labeler. In fact, we can feed all of the results from the end of the pipeline back into the start, to construct a joint learning model (imagine an arrow being drawn from the attribution span labeler back to the connective classifier in Figure 4).

We believe that our discourse parser is very useful in downstream applications, such as text summarization and question answering (QA). For example, a text summarization system may utilize the contrast and restatement relations to recognize updates and redundancy, whereas causal relations can be used in a QA system to answer *why*-questions. The attribution spans from the parser are also very useful for applications on opinion mining and subjectivity analysis to locate the opinion holders.

Discourse structure can also be used in analysis and understanding of the coherence of text. Given two texts and their respective discourse structures, one can analyze and compare these two structures. Discourse patterns extracted from the structures may suggest which text is more coherent than the other. In (Lin *et al.* 2011), we propose a coherence model which applies the output from the discourse parser that we have developed in this work, and demonstrate that this model is capable of distinguishing a coherent text from an incoherent one. We further demonstrate in (Lin *et al.* 2012) that this model can be used to rank a list of machine-generated summaries with regard to their readability. This illustrates the applicability of our discourse parser in other NLP applications.

8 Conclusion

In this work, we have designed a parsing algorithm that performs discourse parsing in the PDTB representation, and implemented it into an end-to-end system in a fully, data-

driven approach. This is the first end-to-end discourse parser that can parse any unrestricted English text into its discourse structure in the PDTB style. We have proposed automatic approaches to locate the relative positions of arguments and label the argument spans when they appear in the same sentence. The performance of the connective classifier is also significantly improved from previous work. We have implemented a component to label the attribution spans for the relations. We evaluated the system both component-wise, as well as in an end-to-end fashion with cascaded errors. We reported overall system F_1 scores of 46.80% for partial matching utilizing gold standard parses, and 38.18% with full automation. Many downstream NLP applications, such as coherence assessment, summarization, and question answering, need to analyze relations beyond sentence-level. We believe that these applications will be able to make use of the output from our discourse parser to improve their performance.

References

Nicholas Asher and Alex Lascarides. 2003. *Logics of Conversation*. Cambridge University Press, Cambridge, England.

Jason Baldridge and Alex Lascarides. 2005. Probabilistic head-driven parsing for discourse structure. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CONLL 2005)*, pages 96–103, Ann Arbor, Michigan, USA.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34:1–34, March.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*, Morristown, NJ, USA.

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 ACL Workshop on Frontiers in Corpus Annotation II: Pie in the Sky*, Ann Arbor, MI, USA.

David duVerle and Helmut Prendinger. 2009. A novel discourse parser based on Support Vector Machine classification. 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 (ACL-IJCNLP 2009)*, Singapore.

Robert Elwell and Jason Baldridge. 2008. Discourse connective argument identification with connective specific rankers. In *Proceedings of the IEEE International Conference on Semantic Computing (ICSC 2008)*, Santa Clara, CA, USA.

Katherine Forbes, Eleni Miltsakaki, Rashmi Prasad, Anoop Sarkar, Aravind Joshi, and Bonnie Webber. 2003. D-LTAG system: Discourse parsing with a lexicalized tree-adjoining grammar. *Journal of Logic, Language and Information*, 12(3):261–279.

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 (IJCNLP 2011)*, pages 1071–1079, Chiang Mai, Thailand, November.

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, July.

Barbara J. Grosz, Scott Weinstein, and Aravind K. Joshi. 1995. Centering: a framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225, June.

Michael A.K Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman, London.

Jerry R. Hobbs. 1985. On the coherence and structure of discourse. Technical Report CSLI-85-37, Center for the Study of Language and Information, Stanford University.

Jerry R. Hobbs. 1990. Literature and cognition. In *CSLI Lecture Notes Number 21*. CSLI Publications.

Le Thanh Huong, Geetha Abeysinghe, and Christian Huyck. 2004. Generating discourse structures for written texts. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, Morristown, NJ, USA.

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

Alistair Knott and Ted Sanders. 1998. The classification of coherence relations and their linguistic markers: An exploration of two languages. *Journal of Pragmatics*, 30(2):135–175.

Alex Lascarides and Nicholas Asher. 1993. Temporal interpretation, discourse relations and commonsense entailment. *Linguistics and Philosophy*, 16(5):437–493.

Alan Lee, Rashmi Prasad, Aravind Joshi, Nikhil Dinesh, and Bonnie Webber. 2006. Complexity of dependencies in discourse: Are dependencies in discourse more complex than in syntax? In *Proceedings of the 5th International Workshop on Treebanks and Linguistic Theories*, Prague, Czech Republic.

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 (EMNLP 2009)*, Singapore.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2010. A PDTB-styled end-to-end discourse parser. Technical Report TRB8/10, School of Computing, National University of Singapore, August.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2011)*, pages 997–1006, Portland, Oregon, USA, June.

Ziheng Lin, Chang Liu, Hwee Tou Ng, and Min-Yen Kan. 2012. Combining coherence models and machine translation evaluation metrics for summarization evaluation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, Jeju, Korea, July.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Daniel Marcu. 1997. *The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts*. Ph.D. thesis, University of Toronto.

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

George A. Miller. 1995. Wordnet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2004. The Penn Discourse Treebank. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, Lisbon, Portugal.

Eleni Miltsakaki, Nikhil Dinesh, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2005. Experiments on sense annotations and sense disambiguation of discourse connectives. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT2005)*, Barcelona, Spain, December.

PDTB-Group, 2007. *The Penn Discourse Treebank 2.0 Annotation Manual*. The PDTB Research Group.

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*, Singapore.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind Joshi. 2008. Easily identifiable discourse relations. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008) Short Papers*, Manchester, UK.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP 2009)*, Singapore.

Livia Polanyi. 1988. A formal model of the structure of discourse. *Journal of Pragmatics*, 12(5–6):601–638.

Livia Polanyi and Remko Scha. 1984. A syntactic approach to discourse semantics. In *Proceedings of the 10th International Conference on Computational Linguistics (COLING 1984)*, pages 413–419. Association for Computational Linguistics.

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 6th International Conference on Language Resources and Evaluation (LREC 2008)*.

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 (LREC-2010)*, pages 2076–2083, Valletta, Malta, May.

Remko Scha and Livia Polanyi. 1988. An augmented context free grammar for discourse. In *Proceedings of the 12th Conference on Computational Linguistics*, pages 573–577. Association for Computational Linguistics.

Peter Rossen Skadhauge and Daniel Hardt. 2005. Syntactic identification of attribution in the RST Treebank. In *Proceedings of the Recent Advances in Natural Language Processing (RANLP 2005)*, Borovets, Bulgaria.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003)*, Edmonton, Canada.

Rajen Subba and Barbara Di Eugenio. 2009. An effective discourse parser that uses rich linguistic information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2009)*, Boulder, Colorado, June. Association for Computational Linguistics.

WenTing Wang, Jian Su, and Chew Lim Tan. 2010. Kernel based discourse relation recognition with temporal ordering information. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, Uppsala, Sweden, July.

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

Bonnie Webber and Aravind Joshi. 1998. Anchoring a lexicalized tree-adjoining grammar for discourse. In *Coling/ACL Workshop on Discourse Relations and Discourse Markers*, pages 86–92.

Bonnie Webber, Markus Egg, and Valia Kordoni. 2011. Discourse structure and language technology. *Natural Language Engineering*, pages 1–54.

Ben Wellner. 2009. *Sequence Models and Ranking Methods for Discourse Parsing*. Ph.D. thesis, Brandeis University.

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 (EMNLP-CoNLL 2007)*, Prague, Czech Republic.

Ben Wellner, James Pustejovsky, Catherine Havasi, Anna Rumshisky, and Roser Sauri. 2006. Classification of discourse coherence relations: An exploratory study using multiple knowledge sources. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, Sydney, Australia.

Florian Wolf and Edward Gibson. 2005. Representing discourse coherence: a corpus-based analysis. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, Morristown, NJ, USA.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010. Predicting discourse connectives for implicit discourse relation recognition. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 1507–1514, Beijing, China.

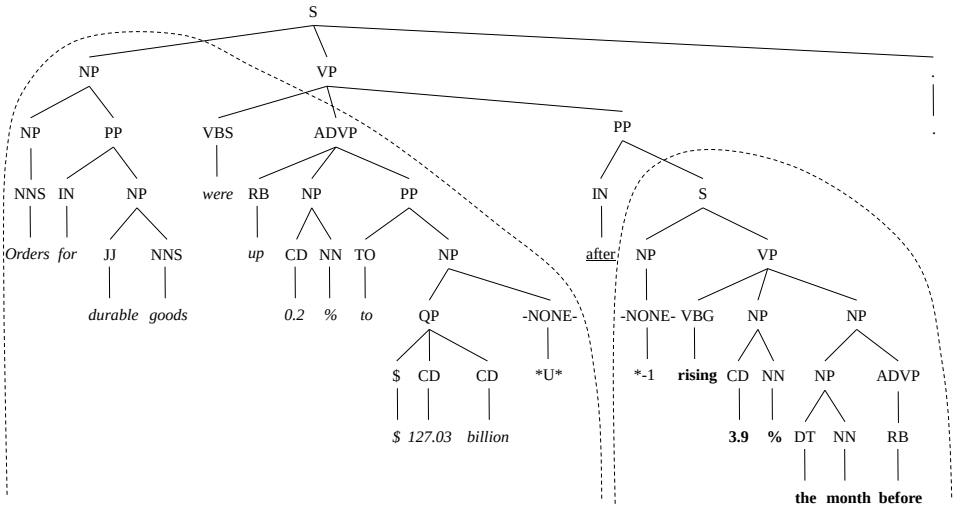

Fig. 10. The constituent parse tree for Example 20.

A Features for the Classifiers in Step 1

Here are the features extracted from the Explicit relation in Example 20 for the classifiers in Step 1 of the parser. The constituent parse of Example 20 is shown in Figure 10.

(20) *Orders for durable goods were up 0.2% to $127.03 billion* after **rising 3.9% the month before.**

(Temporal.Asynchronous – wsj_0036)

A.1 Features for the Connective Classifier

- C POS = IN
- $prev$ + C = billion after
- $prev$ POS = CD
- $prev$ POS + C POS = CD IN
- C + $next$ = after rising
- $next$ POS = VBG
- C POS + $next$ POS = IN VBG
- path of C's parent → root = IN ↑ PP ↑ VP ↑ S
- compressed path of C's parent → root = IN ↑ PP ↑ VP ↑ S

A.2 Features for the Argument Position Classifier

- C string = after

- position of C in the sentence = middle
- C POS = IN
- $prev_1$ = billion
- $prev_1$ POS = CD
- $prev_1$ + C = billion after
- $prev_1$ POS + C POS = CD IN
- $prev_2$ = 127.03
- $prev_2$ POS = CD
- $prev_2$ + C = 127.03 after
- $prev_2$ POS + C POS = CD IN

A.3 Features for the Argument Node Identifier

In the parse tree (Figure 10) for Example 20, we need to identify the Arg1 and Arg2 nodes from the 18 internal nodes (except POS nodes). Here we list the features used to label the S node that covers the Arg2 span.

- C string = after
- C's syntactic category = subordinating
- numbers of left siblings of C = 0
- numbers of right siblings of C = 1
- the path P of C's parent $\rightarrow N$ = IN $\uparrow$ PP $\downarrow$ S
- the path P and whether the number of C's left sibling is greater than one = IN $\uparrow$ PP $\downarrow$ S and no
- the relative position of N to C = right

A.4 Features for the Explicit Classifier

- C string = after
- C's POS = IN
- C + $prev$ = billion after

B Features for the Attribution Span Labeler in Step 3

The following shows features extracted from Example 21 for the attribution span labeler. The constituent parse of Example 21 is shown in Figure 11.

(21) ... averages," declared San Francisco batting coach Dusty Baker after game two. "I'd ...

- lowercased verb in $curr$ = declared
- lemmatized verb in $curr$ = declare
- the first term of $curr$ = declared
- the last term of $curr$ = .
- the last term of $prev$ = "
- the first term of $next$ = "

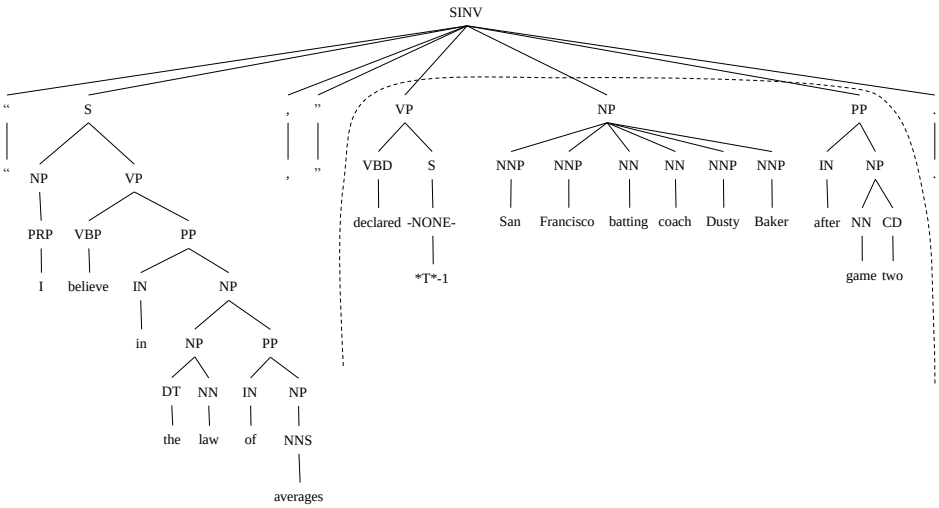

Fig. 11. The constituent parse tree for Example 21.

– the last term of $prev$ + the first term of $curr$ = ” declared
– the last term of $curr$ + the first term of $next$ = . “
– the position of $curr$ in the sentence = middle
– VP → VBD S
– VBD → declared
– NP → NNP NNP NN NN NNP NNP
– NNP → San
– NNP → Francisco
– NN → batting
– NN → coach
– NNP → Dusty
– NNP → Baker
– PP → IN NP
– IN → after
– NP → NN CD
– NN → game
– CD → two

Table 10. *Syntactic categories from Knott (1996) for 100 discourse connectives in PDTB.*

Syntactic category	Discourse connectives
Discourse adverbial	accordingly, additionally, afterwards, also, alternatively, as a result, as an alternative, as well, besides, by comparison, by contrast, by then, consequently, conversely, earlier, either..or, except, finally, for example, for instance, further, furthermore, hence, in addition, in contrast, in fact, in other words, in particular, in short, in sum, in the end, in turn, indeed, instead, later, likewise, meantime, meanwhile, moreover, nevertheless, next, nonetheless, on the contrary, on the other hand, otherwise, overall, previously, rather, regardless, separately, similarly, simultaneously, specifically, still, thereafter, thereby, therefore, thus, ultimately, whereas
Coordinator	and, but, else, if..then, neither..nor, nor, on the one hand..on the other hand, or, plus, then, yet
Subordinator	after, although, as, as if, as long as, as soon as, as though, because, before, before and after, for, however, if, if and when, insofar as, lest, much as, now that, once, since, so, so that, though, till, unless, until, when, when and if, while

C List of discourse connectives and their syntactic categories

Table 10 shows the list of all discourse connectives in the PDTB and their corresponding syntactic categories from Knott (1996).