Text-level Discourse Dependency Parsing

Sujian Li[1] **Liang Wang**[1] **Ziqiang Cao**[1] **Wenjie Li**[2]
[1] Key Laboratory of Computational Linguistics, Peking University, MOE, China
[2] Department of Computing, The Hong Kong Polytechnic University, HongKong
{lisujian,intfloat,ziqiangyeah}@pku.edu.cn
cswjli@comp.polyu.edu.hk

Abstract

Previous researches on Text-level discourse parsing mainly made use of constituency structure to parse the whole document into one discourse tree. In this paper, we present the limitations of constituency based discourse parsing and first propose to use dependency structure to directly represent the relations between elementary discourse units (EDUs). The state-of-the-art dependency parsing techniques, the Eisner algorithm and maximum spanning tree (MST) algorithm, are adopted to parse an optimal discourse dependency tree based on the arc-factored model and the large-margin learning techniques. Experiments show that our discourse dependency parsers achieve a competitive performance on text-level discourse parsing.

1 Introduction

It is widely agreed that no units of the text can be understood in isolation, but in relation to their context. Researches in discourse parsing aim to acquire such relations in text, which is fundamental to many natural language processing applications such as question answering, automatic summarization and so on.

One important issue behind discourse parsing is the representation of discourse structure. Rhetorical Structure Theory (RST) (Mann and Thompson, 1988), one of the most influential discourse theories, posits a hierarchical generative tree representation, as illustrated in Figure 1. The leaves of a tree correspond to contiguous text spans called Elementary Discourse Units (EDUs)[1]. The adjacent EDUs are combined into the larger text spans by rhetorical relations (e.g., Contrast and Elaboration) and the larger text spans continue to be combined until the whole text constitutes a parse tree. The text spans linked by rhetorical relations are annotated as either *nucleus* or *satellite* depending on how salient they are for interpretation. It is attractive and challenging to parse the whole text into one tree.

Since such a hierarchical discourse tree is analogous to a constituency based syntactic tree except that the constituents in the discourse trees are text spans, previous researches have explored different constituency based syntactic parsing techniques (eg. CKY and chart parsing) and various features (eg. length, position et al.) for discourse parsing (Soricut and Marcu, 2003; Joty et al., 2012; Reitter, 2003; LeThanh et al., 2004; Baldridge and Lascarides, 2005; Subba and Di Eugenio, 2009; Sagae, 2009; Hernault et al., 2010b; Feng and Hirst, 2012). However, the existing approaches suffer from at least one of the following three problems. First, it is difficult to design a set of production rules as in syntactic parsing, since there are no determinate generative rules for the interior text spans. Second, the different levels of discourse units (e.g. EDUs or larger text spans) occurring in the generative process are better represented with different features, and thus a uniform framework for discourse analysis is hard to develop. Third, to reduce the time complexity of the state-of-the-art constituency based parsing techniques, the approximate parsing approaches are prone to trap in local maximum.

In this paper, we propose to adopt the *dependency structure* in discourse representation to overcome the limitations mentioned above. Here is the basic idea: the discourse structure consists of EDUs which are linked by the binary, asymmetrical relations called *dependency relations*. A dependency relation holds between a subordinate EDU called the *dependent*, and another EDU on

[1] EDU segmentation is a relatively trivial step in discourse parsing. Since our work focus here is not EDU segmentation but discourse parsing. We assume EDUs are already known.

which it depends called the *head*, as illustrated in Figure 2. Each EDU has one head. So, the dependency structure can be seen as a set of *head-dependent* links, which are labeled by functional relations. Now, we can analyze the relations between EDUs directly, without worrying about any interior text spans. Since dependency trees contain much fewer nodes and on average they are simpler than constituency based trees, the current dependency parsers can have a relatively low computational complexity. Moreover, concerning linearization, it is well known that dependency structures can deal with non-projective relations, while constituency-based models need the addition of complex mechanisms like transformations, movements and so on. In our work, we adopt the graph based dependency parsing techniques learned from large sets of annotated dependency trees. The Eisner (1996) algorithm and maximum spanning tree (MST) algorithm are used respectively to parse the optimal projective and non-projective dependency trees with the large-margin learning technique (Crammer and Singer, 2003). To the best of our knowledge, we are the first to apply the dependency structure and introduce the dependency parsing techniques into discourse analysis.

The rest of this paper is organized as follows. Section 2 formally defines discourse dependency structure and introduces how to build a discourse dependency treebank from the existing RST corpus. Section 3 presents the discourse parsing approach based on the Eisner and MST algorithms. Section 4 elaborates on the large-margin learning technique as well as the features we use. Section 5 discusses the experimental results. Section 6 introduces the related work and Section 7 concludes the paper.

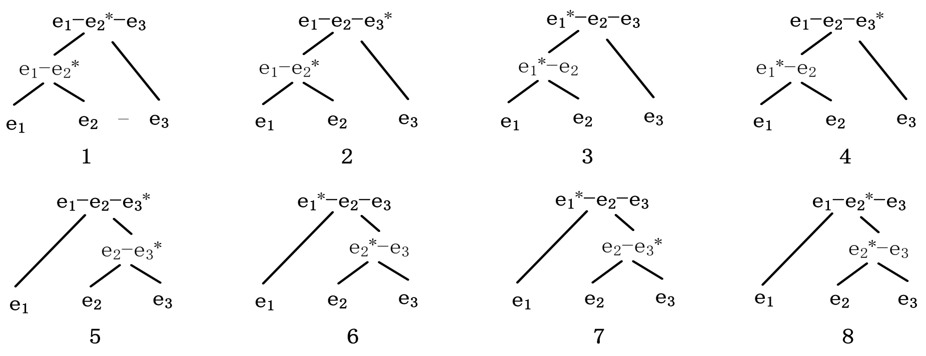

Figure 1: Headed Constituency based Discourse Tree Structure (e_1,e_2 and e_3 denote three EDUs, and * denotes the NUCLEUS constituent)

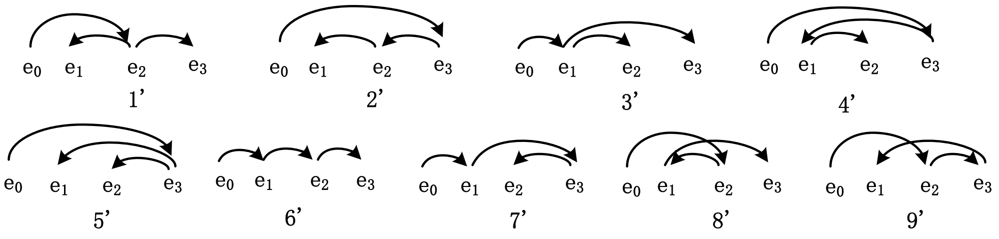

Figure 2: Discourse Dependency Tree Structures (e_1,e_2 and e_3 denote three EDUS, and the directed arcs denote one dependency relations. The artificial e_0 is also displayed here.)

2 Discourse Dependency Structure and Tree Bank

2.1 Discourse Dependency Structure

Similar to the syntactic dependency structure defined by McDonald (2005a, 2005b), we insert an artificial EDU e_0 in the beginning for each document and label the dependency relation linking from e_0 as **ROOT**. This treatment will simplify both formal definitions and computational implementations. Normally, we assume that each EDU should have one and only one head except for e_0. A labeled directed arc is used to represent the dependency relation from one head to its dependent. Then, discourse dependency structure can be formalized as the labeled directed graph, where nodes correspond to EDUs and labeled arcs correspond to labeled dependency relations.

We assume that the text[2] T is composed of n+1 EDUs including the artificial e_0. That is T=e_0 e_1 e_2 … e_n. Let R={r_1,r_2, … ,r_m} denote a finite set of functional relations that hold between two EDUs. Then a discourse dependency graph can be denoted by G=<V, A> where V denotes a set of nodes and A denotes a set of labeled directed arcs, such that for the text T=e_0 e_1 e_2 … e_n and the label set R the following holds:

(1) $V = \{ e_0, e_1, e_2, \ldots e_n \}$

(2) $A \subseteq V\times R \times V$, where $<e_i, r, e_j>\in A$ represents an arc from the head e_i to the dependent e_j labeled with the relation r.

(3) If $<e_i, r, e_j>\in A$ then $<e_k, r', e_j>\notin A$ for all $k\neq i$

(4) If $<e_i, r, e_j>\in A$ then $<e_i, r', e_j>\notin A$ for all $r'\neq r$

The third condition assures that each EDU has one and only one head and the fourth tells that only one kind of dependency relation holds between two EDUs. According to the definition, we illustrate all the 9 possible unlabeled dependency trees for a text containing three EDUs in Figure 2. The dependency trees 1' to 7' are projective while 8' and 9' are non-projective with crossing arcs.

2.2 Our Discourse Dependency Treebank

To automatically conduct discourse dependency parsing, constructing a discourse dependency treebank is fundamental. It is costly to manually construct such a treebank from scratch. Fortunately, RST Discourse Treebank (RST-DT) (Carlson et al., 2001) is an available resource to help with.

A RST tree constitutes a hierarchical structure for one document through rhetorical relations. A total of 110 fine-grained relations (e.g. Elaboration-part-whole and List) were used for tagging RST-DT. They can be categorized into 18 classes (e.g. Elaboration and Joint). All these relations can be hypotactic ("mononuclear") or paratactic ("multi-nuclear"). A hypotactic relation holds between a *nucleus* span and an adjacent *satellite* span, while a paratactic relation connects two or more equally important adjacent *nucleus* spans. For convenience of computation, we convert the n-ary (n>2) RST trees[3] to binary trees through adding a new node for the latter n-1 nodes and assume each relation is connected to only one *nucleus*[4]. This departure from the original theory is not such a major step as it may appear, since any nucleus is known to contribute to the essential meaning. Now, each RST tree can be seen as a headed constituency based binary tree where the nuclei are heads and the children of each node are linearly ordered. Given three EDUs[5], Figure 1 shows the possible 8 headed constituency based trees where the superscript * denotes the heads (nuclei). We use dependency trees to simulate the headed constituency based trees.

Contrasting Figure 1 with Figure 2, we use dependency tree 1' to simulate binary trees 1 and 8, and dependency tress 2'- 7' to simulate binary trees 2-7 correspondingly. The rhetorical relations in RST trees are kept as the functional relations which link the two EDUs in dependency trees. With this kind of conversion, we can get our discourse dependency treebank. It is worth noting that the non-projective trees like 8' and 9' do not exist in our dependency treebank, though they are eligible according to the definition of discourse dependency graph.

3 Discourse Dependency Parsing

3.1 System Overview

As stated above, T=e_0 e_1 …e_n represents an input text (document) where e_i denotes the ith EDU of T. We use V to denote all the EDU nodes and $V\times R\times V_{-0}$ (V_{-0} =V-{e_0}) denote all the possible discourse dependency arcs. The goal of discourse dependency parsing is to parse an optimal spanning tree from $V\times R\times V_{-0}$. Here we follow the arc factored method and define the score of a dependency tree as the sum of the scores of all the arcs in the tree. Thus, the optimal dependency tree for T is a spanning tree with the highest score and obtained through the function $DT(T,\boldsymbol{w})$:

$$\begin{aligned} DT(T,\boldsymbol{w}) &= argmax_{G_T \subseteq V\times R\times V_{-0}} score(T, G_T) \\ &= argmax_{G_T \subseteq V\times R\times V_{-0}} \sum_{<e_i,r,e_j>\in G_T} \lambda(e_i, r, e_j) \\ &= argmax_{G_T \subseteq V\times R\times V_{-0}} \sum_{<e_i,r,e_j>\in G_T} \boldsymbol{w}\cdot \mathbf{f}(e_i, r, e_j) \end{aligned}$$

where G_T means a possible spanning tree with $score(T, G_T)$ and $\lambda(e_i, r, e_j)$ denotes the score of the arc <e_i, r, e_j> which is calculated according to its feature representation $\mathbf{f}(e_i,r,e_j)$ and a weight vector $\boldsymbol{w}$.

Next, two basic problems need to be solved: how to find the dependency tree with the highest

[2] The two terms "text" and "document" are used interchangeably and represent the same meaning.

[3] According to our statistics, there are totally 381 n-ary relations in RST-DT.

[4] We set the first nucleus as the only nucleus.

[5] We can easily get all possible headed binary trees for one more complex text containing more than three EDUs, by extending the 8 possible situations for three EDUs.

score for *T* given all the arc scores (i.e. a parsing problem), and how to learn and compute the scores of arcs according to a set of arc features (i.e. a learning problem).

The following of this section addresses the first problem. Given the text *T*, we first reduce the multi-digraph composed of all possible arcs to the digraph. The digraph keeps only one arc $<e_i, r, e_j>$ between two nodes which satisfies $\lambda(e_i, r, e_j) = max_{r'}\lambda(e_i, r', e_j)$. Thus, we can proceed with a reduction from labeled parsing to unlabeled parsing. Next, two algorithms, i.e. the Eisner algorithm and MST algorithm, are presented to parse the projective and non-projective unlabeled dependency trees respectively.

3.2 Eisner Algorithm

It is well known that projective dependency parsing can be handled with the Eisner algorithm (1996) which is based on the bottom-up dynamic programming techniques with the time complexity of O(n^3). The basic idea of the Eisner algorithm is to parse the left and right dependents of an EDU independently and combine them at a later stage. This reduces the overhead of indexing heads. Only two binary variables, i.e. *c* and *d*, are required to specify whether the heads occur leftmost or rightmost and whether an item is complete.

```
Eisner($T$, $\lambda$)
Input: Text $T=e_0 e_1 \dots e_n$; Arc scores $\lambda(e_i,e_j)$
1   Instantiate $E[i, i, d, c]=0.0$ for all $i, d, c$
2   For $m := 1$ to $n$
3       For $i := 1$ to $n$
4           $j = i + m$
5           if $j > n$ then break;
6           # Create subgraphs with $c=0$ by adding arcs
7           $E[i, j, 0, 0]=\max_{i\le q\le j}(E[i,q,1,1]+E[q+1,j,0,1]+\lambda(e_j,e_i))$
8           $E[i, j, 1, 0]=\max_{i\le q\le j}(E[i,q,1,1]+E[q+1,j,0,1]+\lambda(e_i,e_j))$
9           # Add corresponding left/right subgraphs
10          $E[i, j, 0, 1]=\max_{i\le q\le j}(E[i,q,0,1]+E[q,j,0,0]$
11          $E[i, j, 1, 1]=\max_{i\le q\le j}(E[i,q,1,0]+E[q,j,1,1])$
```

Figure 3: Eisner Algorithm

Figure 3 shows the pseudo-code of the Eisner algorithm. A dynamic programming table E[*i*,*j*,*d*,*c*] is used to represent the highest scored subtree spanning e_i to e_j. *d* indicates whether e_i is the head (*d*=1) or e_j is head (*d*=0). *c* indicates whether the subtree will not take any more dependents (*c*=1) or it needs to be completed (*c*=0). The algorithm begins by initializing all length-one subtrees to a score of 0.0. In the inner loop, the first two steps (Lines 7 and 8) are to construct the new dependency arcs by taking the maximum over all the internal indices ($i \le q \le j$) in the span, and calculating the value of merging the two subtrees and adding one new arc. The last two steps (Lines 10 and 11) attempt to achieve an optimal left/right subtree in the span by adding the corresponding left/right subtree to the arcs that have been added previously. This algorithm considers all the possible subtrees. We can then get the optimal dependency tree with the score $E[0,n,1,1]$.

3.3 Maximum Spanning Tree Algorithm

As the bottom-up Eisner Algorithm must maintain the nested structural constraint, it cannot parse the non-projective dependency trees like 8' and 9' in Figure 2. However, the non-projective dependency does exist in real discourse. For example, the earlier text mainly talks about the topic *A* with mentioning the topic *B*, while the latter text gives a supplementary explanation for the topic *B*. This example can constitute a non-projective tree and its pictorial diagram is exhibited in Figure 4. Following the work of McDonald (2005b), we formalize discourse dependency parsing as searching for a maximum spanning tree (MST) in a directed graph.

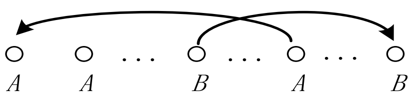

Figure 4: Pictorial Diagram of Non-projective Trees

Chu and Liu (1965) and Edmonds (1967) independently proposed the virtually identical algorithm named the Chu-Liu/Edmonds algorithm, for finding MSTs on directed graphs (McDonald et al. 2005b). Figure 5 shows the details of the Chu-Liu/Edmonds algorithm for discourse parsing. Each node in the graph greedily selects the incoming arc with the highest score. If one tree results, the algorithm ends. Otherwise, there must exist a cycle. The algorithm contracts the identified cycle into a single node and recalculates the scores of the arcs which go in and out of the cycle. Next, the algorithm recursively call itself on the contracted graph. Finally, those arcs which go in or out of one cycle will recover themselves to connect with the original nodes in *V*. Like McDonald et al. (2005b), we adopt an efficient implementation of the Chu-Liu/Edmonds algorithm that is proposed by Tarjan (1997) with O(n^2) time complexity.

```
Chu-Liu-Edmonds($G$, $\lambda$)
Input: Text $T=e_0 e_1 \dots e_n$; Arc scores $\lambda(e_i,e_j)$
1   $A' = \{<e_i, e_j> | e_i = \arg\max \lambda(e_i,e_j); 1 \le j \le |V|\}$
2   $G' = (V, A')$
3   If $G'$ has no cycles, then return $G'$
4   Find an arc set $A_C$ that is a cycle in $G'$
5   $<G_C, ep> = \text{contract}(G, A_C, \lambda)$
6   $G = (V, A) = \text{Chu-Liu-Edmonds}(G_C, \lambda)$
7   For the arc $<e_i,e_C>$ where $ep(e_i,e_C)=e_j$:
8       $A=A \cup A_C \cup \{<e_i,e_j>\} - \{<e_i,e_C>, <a(e_j),e_j>\}$
9   For the arc $<e_C, e_i>$ where $ep(e_C, e_i)=e_j$:
10      $A=A \cup \{<e_j,e_i>\} - \{<e_C,e_i>\}$
11  $V = V$
12  Return $G$
Contract($G=(V,A)$, $A_C$, $\lambda$)
1  Let $G_C$ be the subgraph of $G$ excluding nodes in $C$
2  Add a node $e_C$ to $G_C$ denoting the cycle $C$
3  For $e_j \in V-C$ : $\exists e_i \in C$ $<e_i,e_j> \in A$
4      Add arc $<e_C,e_j>$ to $G_C$ with
           $ep(e_C,e_j)=\arg\max_{e_i \in C} \lambda(e_i,e_j)$
5      $\lambda(e_C,e_j) = \lambda(ep(e_C,e_j),e_j)$
6  For $e_i \in V-C$: $\exists e_j \in C$ $(e_i,e_j) \in A$
7      Add arc $<e_i,e_C>$ to $G_C$ with
           $ep(e_i,e_C)= =\arg\max_{e_i \in C} [\lambda(e_i,e_j)-\lambda(a(e_i),e_j)]$
8      $\lambda(e_i,e_C) = \lambda(e_i,e_j)-\lambda(a(e_i),e_j)+\text{score}(C)$
9  Return $<G_C, ep>$
```

Figure 5: Chu-Liu/Edmonds MST Algorithm

4 Learning

In Section 3, we assume that the arc scores are available. In fact, the score of each arc is calculated as a linear combination of feature weights. Thus, we need to determine the features for arc representation first. With referring to McDonald et al. (2005a; 2005b), we use the Margin Infused Relaxed Algorithm (MIRA) to learn the feature weights based on a training set of documents annotated with dependency structures $\{(T_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ where $\boldsymbol{y}_i$ denotes the correct dependency tree for the text T_i.

4.1 Features

Following (Feng and Hirst, 2012; Lin et al., 2009; Hernault et al., 2010b), we explore the following 6 feature types combined with relations to represent each labeled arc $<e_i, r, e_j>$.

(1) WORD: The first one word, the last one word, and the first bigrams in each EDU, the pair of the two first words and the pair of the two last words in the two EDUs are extracted as features.

(2) POS: The first one and two POS tags in each EDU, and the pair of the two first POS tags in the two EDUs are extracted as features.

(3) Position: These features concern whether the two EDUs are included in the same sentence, and the positions where the two EDUs are located in one sentence, one paragraph, or one document.

(4) Length: The length of each EDU.

(5) Syntactic: POS tags of the dominating nodes as defined in Soricut and Marcu (2003) are extracted as features. We use the syntactic trees from the Penn Treebank to find the dominating nodes,.

(6) Semantic similarity: We compute the semantic relatedness between the two EDUs based on WordNet. The word pairs are extracted from (e_i, e_j) and their similarity is calculated. Then, we can get a weighted complete bipartite graph where words are deemed as nodes and similarity as weights. From this bipartite graph, we get the maximum weighted matching and use the averaged weight of the matches as the similarity between e_i and e_j. In particular, we use path_similarity, wup_similarity, res_similarity, jcn_similarity and lin_similarity provided by the nltk.wordnet.similarity (Bird et. al., 2009) package for calculating word similarity.

As for relations, we experiment two sets of relation labels from RST-DT. One is composed of 19 coarse-grained relations and the other 111 fine-grained relations[6].

4.2 MIRA based Learning

Margin Infused Relaxed Algorithm (MIRA) is an online algorithm for multiclass classification and is extended by Taskar et al. (2003) to cope with structured classification.

```
MIRA   Input: a training set $\{(T_i, \boldsymbol{y}_i)\}_{i=1}^{N}$
1   $\boldsymbol{w}^0 = 0$; $\boldsymbol{v} = 0$; $j = 0$
2   For $iter$ := 1 to $K$
3       For $i$ := 1 to $N$
4           update $\boldsymbol{w}$ according to $(T_i, \boldsymbol{y}_i)$:
              $\min \| \boldsymbol{w}^{j+1} - \boldsymbol{w}^{j} \|$
              s.t. $s(T_i, \boldsymbol{y}_i) - s(T_i, \boldsymbol{y}_i') \ge L(\boldsymbol{y}_i, \boldsymbol{y}_i')$
              where $\boldsymbol{y}_i' = DT(T_i, \boldsymbol{w}^j)$
5             $\boldsymbol{v} = \boldsymbol{v} + \boldsymbol{w}^j$ ;
6             $j = j+1$
7   $\boldsymbol{w} = \boldsymbol{v}/(K*N)$
```

Figure 6: MIRA based Learning

Figure 6 gives the pseudo-code of the MIRA algorithm (McDonld et al., 2005b). This algorithm is designed to update the parameters $\boldsymbol{w}$ using a single training instance $(T_i, \boldsymbol{y}_i)$ in each iteration. On each update, MIRA attempts to keep the norm of the change to the weight vector

[6] 19 relations include the original 18 relation in RST-DT plus one artificial **ROOT** relation. The 111 relations also include the ROOT relation.

as small as possible, which is subject to constructing the correct dependency tree under consideration with a margin at least as large as the loss of the incorrect dependency trees. We define the loss of a discourse dependency tree $\boldsymbol{y}_i'$ (denoted by $L(\boldsymbol{y}_i, \boldsymbol{y}_i')$) as the number of the EDUs that have incorrect heads. Since there are exponentially many possible incorrect dependency trees and thus exponentially many margin constraints, here we relax the optimization and stay with a single best dependency tree $\boldsymbol{y}_i' = DT(T_i, \boldsymbol{w}^j)$ which is parsed under the weight vector $\boldsymbol{w}^j$. In this algorithm, the successive updated values of $\boldsymbol{w}$ are accumulated and averaged to avoid overfitting.

5 Experiments

5.1 Preparation

We test our methods experimentally using the discourse dependency treebank which is built as in Section 2. The training part of the corpus is composed of 342 documents and contains 18,765 EDUs, while the test part consists of 38 documents and 2,346 EDUs. The number of EDUs in each document ranges between 2 and 304. Two sets of relations are adopted. One is composed of 19 relations and Table 1 shows the number of each relation in the training and test corpus. The other is composed of 111 relations. Due to space limitation, Table 2 only lists the 10 highest-distributed relations with regard to their frequency in the training corpus.

The following experiments are conducted: (1) to measure the parsing performance with different relation sets and different feature types; (2) to compare our parsing methods with the state-of-the-art discourse parsing methods.

Relations	**Train**	**Test**	**Relations**	**Train**	**Test**
Elaboration	6879	796	Temporal	426	73
Attribution	2641	343	ROOT	342	38
Joint	1711	212	Compari.	273	29
Same-unit	1230	127	Condition	258	48
Contrast	944	146	Manner.	191	27
Explanation	849	110	Summary	188	32
Background	786	111	Topic-Cha.	187	13
Cause	785	82	Textual	147	9
Evaluation	502	80	TopicCom.	126	24
Enablement	500	46	**Total**	18765	2346

Table 1: Coarse-grained Relation Distribution

Relations	**Train**	**Test**
Elaboration-additional	2912	312
Attribution	2474	329
Elaboration-object-attribute-e	2274	250
List	1690	206
Same-unit	1230	127
Elaboration-additional-e	747	69
Circumstance	545	80
Explanation-argumentative	524	70
Purpose	430	43
Contrast	358	64

Table 2: 10 Highest Distributed Fine-grained Relations

5.2 Feature Influence on Two Relation Sets

So far, researches on discourse parsing avoid adopting too fine-grained relations and the relation sets containing around 20 labels are widely used. In our experiments, we observe that adopting a fine-grained relation set can even be helpful to building the discourse trees. Here, we conduct experiments on two relation sets that contain 19 and 111 labels respectively. At the same time, different feature types are tested their effects on discourse parsing.

Method	Features	Unlabeled Acc.	Labeled Acc.
Eisner	1+2	0.3602	0.2651
	1+2+3	0.7310	0.4855
	1+2+3+4	0.7370	0.4868
	1+2+3+4+5	**0.7447**	**0.4957**
	1+2+3+4+5+6	**0.7455**	**0.4983**
MST	1+2	0.1957	0.1479
	1+2+3	0.7246	0.4783
	1+2+3+4	0.7280	0.4795
	1+2+3+4+5	**0.7340**	**0.4915**
	1+2+3+4+5+6	**0.7331**	**0.4851**

Table 3: Performance Using Coarse-grained Relations.

Method	Feature types	Unlabeled Acc.	Labeled Acc.
Eisner	1+2	0.3743	0.2421
	1+2+3	0.7451	0.4079
	1+2+3+4	0.7472	0.4041
	1+2+3+4+5	**0.7506**	**0.4254**
	1+2+3+4+5+6	**0.7485**	**0.4288**
MST	1+2	0.2080	0.1300
	1+2+3	0.7366	0.4054
	1+2+3+4	0.7468	0.4071
	1+2+3+4+5	**0.7494**	**0.4288**
	1+2+3+4+5+6	**0.7460**	**0.4309**

Table 4: Performance Using Fine-grained Relations.

Based on the MIRA leaning algorithm, the Eisner algorithm and MST algorithm are used to parse the test documents respectively. Referring to the evaluation of syntactic dependency parsing, we use *unlabeled accuracy* to calculate the ratio of EDUs that correctly identify their heads, *labeled accuracy* the ratio of EDUs that have both correct heads and correct relations. Table 3 and Table 4 show the performance on two relation sets. The numbers (1-6) represent the corresponding feature types described in Section 4.1.

From Table 3 and Table 4, we can see that the addition of more feature types, except the 6th feature type (semantic similarity), can promote the performance of relation labeling, whether using the coarse-grained 19 relations and the fine-grained 111 relations. As expected, the first and second types of features (WORD and POS) are the ones which play an important role in building and labeling the discourse dependency trees. These two types of features attain similar performance on two relation sets. The Eisner algorithm can achieve unlabeled accuracy around 0.36 and labeled accuracy around 0.26, while MST algorithm achieves unlabeled accuracy around 0.20 and labeled accuracy around 0.14.

The third feature type (Position) is also very helpful to discourse parsing. With the addition of this feature type, both unlabeled accuracy and labeled accuracy exhibit a marked increase. Especially, when applying MST algorithm on discourse parsing, unlabeled accuracy rises from around 0.20 to around 0.73. This result is consistent with Hernault's work (2010b) whose experiments have exhibited the usefulness of those position-related features. The other two types of features which are related to length and syntactic parsing, only promote the performance slightly.

As we employed the MIRA learning algorithm, it is possible to identify which specific features are useful, by looking at the weights learned to each feature using the training data. Table 5 selects 10 features with the highest weights in absolute value for the parser which uses the coarse-grained relations, while Table 6 selects the top 10 features for the parser using the fine-grained relations. Each row denotes one feature: the left part before the symbol "&" is from one of the 6 feature types and the right part denotes a specific relation. From Table 5 and Table 6, we can see that some features are reasonable. For example, The sixth feature in Table 5 represents that the dependency relation is preferred to be labeled *Explanation* with the fact that "because" is the first word of the dependent EDU. From these two tables, we also observe that most of the heavily weighted features are usually related to those highly distributed relations. When using the coarse-grained relations, the popular relations (eg. Elaboration, Attribution and Joint) are always preferred to be labeled. When using the fine-grained relations, the large relations including List and Elaboration-object-attribute-e are given the precedence of labeling. This phenomenon is mainly caused by the sparseness of the training corpus and the imbalance of relations. To solve this problem, the augment of training corpus is necessary.

	Feature description	**Weight**
1	Last two words in dependent EDU are "appeals court" & Joint	0.475
2	First word in dependent EDU is "racked" & Elaboration	0.445
3	First two words in head EDU are "I 'd" & Attribution	0.324
4	Last word in dependent EDU is "in" & Elaboration	-0.323
5	The res_similarity between two EDUs is 0 & Elaboration	0.322
6	First word in dependent EDU is "because" & Explanation	0.306
7	First POS in head EDU is "DT" & Joint	-0.299
8	First two words in dependent EDU are "that required" & Elaboration	0.287
9	First two words in dependent EDU are "that the" & Elaboration	0.277
10	First word in dependent EDU is "because" & Cause	0.265

Table 5: Top 10 Feature Weights for Coarse-grained Relation Labeling (Eisner Algorithm)

	Features	**Weight**
1	Last two words in dependent EDU are "appeals court" & List	0.576
2	First two words in head EDU are "I 'd" & Attribution	0.385
3	First two words in dependent EDU is "that the" & Elaboration-object-attribute-e	0.348
4	First POS in head EDU is "DT" & List	-0.323
5	Last word in dependent EDU is "in" & List	-0.286
6	First word in dependent EDU is "racked" & Elaboration-object-attribute-e	0.445
7	First two word pairs are <"In an","But even"> & List	-0.252
8	Dependent EDU has a dominating node tagged "CD"& Elaboration-object-attribute-e	-0.244
9	First two words in dependent EDU are "patents disputes" & Purpose	0.231
10	First word in dependent EDU is "to" & Purpose	0.230

Table 6: Top 10 Feature Weights for Coarse-grained Relation Labeling (Eisner Algorithm)

Unlike previous discourse parsing approaches, our methods combine tree building and relation labeling into a uniform framework naturally. This means that relations play a role in building the dependency tree structure. From Table 3 and Table 4, we can see that fine-grained relations are more helpful to building unlabeled discourse trees more than the coarse-grained relations. The best result of unlabeled accuracy using 111 relations is 0.7506, better than the best performance (0.7447) using 19 relations. We can also see that the labeled accuracy using the fine-grained relations can achieve 0.4309, only 0.06 lower than the best labeled accuracy (0.4915) using the coarse-grained relations.

In addition, comparing the MST algorithm with the Eisner algorithm, Table 3 and Table 4 show that their performances are not significantly different from each other. But we think that MST algorithm has more potential in discourse dependency parsing, because our converted discourse dependency treebank contains only projective trees and somewhat suppresses the MST algorithm to exhibit its advantage of parsing non-projective trees. In fact, we observe that some non-projective dependencies produced by the MST algorithm are even reasonable than what they are in the dependency treebank. Thus, it is important to build a manually labeled discourse dependency treebank, which will be our future work.

5.3 Comparison with Other Systems

The state-of-the-art discourse parsing methods normally produce the constituency based discourse trees. To comprehensively evaluate the performance of a labeled constituency tree, the blank tree structure ('S'), the tree structure with nuclearity indication ('N'), and the tree structure with rhetorical relation indication but no nuclearity indication ('R') are evaluated respectively using the *F* measure (Marcu 2000).

To compare our discourse parsers with others, we adopt MIRA and Eisner algorithm to conduct discourse parsing with all the 6 types of features and then convert the produced projective dependency trees to constituency based trees through their correspondence as stated in Section 2. Our parsers using two relation sets are named *Our-coarse* and *Our-fine* respectively. The inputted EDUs of our parsers are from the standard segmentation of RST-DT. Other text-level discourse parsing methods include: (1) *Percep-coarse*: we replace MIRA with the averaged perceptron learning algorithm and the other settings are the same with *Our-coarse*; (2) *HILDA-manual* and *HILDA-seg* are from Hernault (2010b)'s work, and their inputted EDUs are from RST-DT and their own EDU segmenter respectively; (3) *LeThanh* indicates the results given by LeThanh el al. (2004), which built a multi-level rule based parser and used 14 relations evaluated on 21 documents from RST-DT; (4) *Marcu* denotes the results given by Marcu(2000)'s decision-tree based parser which used 15 relations evaluated on unspecified documents.

Table 7 shows the performance comparison for all the parsers mentioned above. *Human* denotes the manual agreement between two human annotators. From this table, we can see that both our parsers perform better than all the other parsers as a whole, though our parsers are not developed directly for constituency based trees. Our parsers do not exhibit obvious advantage than *HILDA-manual* on labeling the blank tree structure, because our parsers and *HILDA-manual* all perform over 94% of *Human* and this performance level somewhat reaches a bottleneck to promote more. However, our parsers outperform the other parsers on both nuclearity and relation labeling. *Our-coarse* achieves 94.2% and 91.8% of the human *F*-scores, on labeling nuclearity and relation respectively, while *Our-fine* achieves 95.2% and 87.6%. We can also see that the averaged perceptron learning algorithm, though simple, can achieve a comparable performance, better than *HILDA-manual*. The parsers *HILDA-seg*, *LeThanh* and *Marcu* use their own automatic EDU segmenters and exhibit a relatively low performance. This means that EDU segmentation is important to a practical discourse parser and worth further investigation.

	S	N	R
Our-coarse	**82.9**	**73.0**	**60.6**
Our-fine	**83.4**	**73.8**	**57.8**
Percep-coarse	82.3	72.6	59.4
HILDA-manual	83.0	68.4	55.3
HILDA-seg	72.3	59.1	47.8
LeThanh	53.7	47.1	39.9
Marcu	44.8	30.9	18.8
Human	**88.1**	**77.5**	**66.0**

Table 7: Full Parser Evaluation

	MAFS	WAFS	Acc
Our-coarse	**0.454**	**0.643**	**66.84**
Percep-coarse	0.438	0.633	65.37
Feng	0.440	0.607	65.30
HILDA-manual	0.428	0.604	64.18
Baseline	-	-	35.82

Table 8: Relation Labeling Performance

To further compare the performance of relation labeling, we follow Hernault el al. (2010a) and use Macro-averaged F-score (MAFS) to evaluate each relation. Due to space limitation, we do not list the *F* scores for each relation. Macro-averaged F-score is not influenced by the number of instances that are contained in each relation. Weight-averaged F-score (WAFS) weights the performance of each relation by the number of its existing instances. Table 8 compares our parser *Our-coarse* with other parsers *HILDA-manual*, *Feng* (Feng and Hirst, 2012) and *Baseline*. *Feng* (Feng and Hirst, 2012) can be seen as a strengthened version of HILDA which adopts more features and conducts feature selection. *Baseline* always picks the most frequent relation (i.e. Elaboration). From the results, we find that *Our-coarse* consistently provides superior performance for most relations over other parsers, and therefore results in higher MAFS and WAFS.

6 Related Work

So far, the existing discourse parsing techniques are mainly based on two well-known treebanks. One is the Penn Discourse TreeBank (PDTB) (Prasad et al., 2007) and the other is RST-DT.

PDTB adopts the predicate-arguments representation by taking an implicit/explicit connective as a predication of two adjacent sentences (arguments). Then the discourse relation between each pair of sentences is annotated independently to characterize its predication. A majority of researches regard discourse parsing as a classification task and mainly focus on exploiting various linguistic features and classifiers when using PDTB (Wellner et al., 2006; Pitler et al., 2009; Wang et al., 2010). However, the predicate-arguments annotation scheme itself has such a limitation that one can only obtain the local discourse relations without knowing the rich context.

In contrast, RST and its treebank enable people to derive a complete representation of the whole discourse. Researches have begun to investigate how to construct a RST tree for the given text. Since the RST tree is similar to the constituency based syntactic tree except that the constituent nodes are different, the syntactic parsing techniques have been borrowed for discourse parsing (Soricut and Marcu, 2003; Baldridge and Lascarides, 2005; Sagae, 2009; Hernault et al., 2010b; Feng and Hirst, 2012). Soricut and Marcu (2003) use a standard bottom-up chart parsing algorithm to determine the discourse structure of sentences. Baldridge and Lascarides (2005) model the process of discourse parsing with the probabilistic head driven parsing techniques. Sagae (2009) apply a transition based constituent parsing approach to construct a RST tree for a document. Hernault et al. (2010b) develop a greedy bottom-up tree building strategy for discourse parsing. The two adjacent text spans with the closest relations are combined in each iteration. As the extension of Hernault's work, Feng and Hirst (2012) further explore various features aiming to achieve better performance. However, as analyzed in Section 1, there exist three limitations with the constituency based discourse representation and parsing. We innovatively adopt the dependency structure, which can be benefited from the existing RST-DT, to represent the discourse. To the best of our knowledge, this work is the first to apply dependency structure and dependency parsing techniques in discourse analysis.

7 Conclusions

In this paper, we present the benefits and feasibility of applying dependency structure in text-level discourse parsing. Through the correspondence between constituency-based trees and dependency trees, we build a discourse dependency treebank by converting the existing RST-DT. Based on dependency structure, we are able to directly analyze the relations between the EDUs without worrying about the additional interior text spans, and apply the existing state-of-the-art dependency parsing techniques which have a relatively low time complexity. In our work, we use the graph based dependency parsing techniques learned from the annotated dependency trees. The Eisner algorithm and the MST algorithm are applied to parse the optimal projective and non-projective dependency trees respectively based on the arc-factored model. To calculate the score for each arc, six types of features are explored to represent the arcs and the feature weights are learned based on the MIRA learning technique. Experimental results exhibit the effectiveness of the proposed approaches. In the future, we will focus on non-projective discourse dependency parsing and explore more effective features.

Acknowledgments

This work was partially supported by National High Technology Research and Development Program of China (No. 2012AA011101), National Key Basic Research Program of China (No. 2014CB340504), National Natural Science Foundation of China (No. 61273278), and National Key Technology R&D Program (No: 2011BAH10B04-03). We also thank the three anonymous reviewers for their helpful comments.

References

Jason Baldridge and Alex Lascarides. 2005. *Probabilistic Head-driven Parsing for Discourse Structure*. In Proceedings of the Ninth Conference on Computational Natural Language Learning, pages 96–103.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python — Analyzing Text with the Natural Language Toolkit*. O'Reilly.

Lynn Carlson, Daniel Marcu, and Mary E. Okurowski. 2001. *Building a Discourse-tagged Corpus in the Framework of Rhetorical Structure Theory*. Proceedings of the Second SIGdial Workshop on Discourse and Dialogue-Volume 16, pages 1–10.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. *On the Shortest Arborescence of a Directed Graph*, Science Sinica, v.14, pp.1396-1400.

Koby Crammer and Yoram Singer. 2003. *Ultraconservative Online Algorithms for Multiclass Problems*. JMLR.

Jack Edmonds. 1967. *Optimum Branchings*, J. Research of the National Bureau of Standards, 71B, pp.233-240.

Jason Eisner. 1996. *Three New Probabilistic Models for Dependency Parsing: An Exploration*. In Proc. COLING.

Vanessa Wei Feng and Graeme Hirst. *Text-level Discourse Parsing with Rich Linguistic Features*, Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 60–68, Jeju, Republic of Korea, 8-14 July 2012.

Hugo Hernault, Danushka Bollegala, and Mitsuru Ishizuka. 2010a. *A Semi-supervised Approach to Improve Classification of Infrequent Discourse Relations Using Feature Vector Extension*. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 399–409, Cambridge, MA, October. Association for Computational Linguistics.

Hugo Hernault, Helmut Prendinger, David A. duVerle, and Mitsuru Ishizuka. 2010b. *HILDA: A Discourse Parser Using Support Vector Machine Classification*. Dialogue and Discourse, 1(3):1–33.

Shafiq Joty, Giuseppe Carenini and Raymond T. Ng. *A Novel Discriminative Framework for Sentence-level Discourse Analysis*. EMNLP-CoNLL '12 Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning Stroudsburg, PA, USA.

Huong LeThanh, Geetha Abeysinghe, and Christian Huyck. 2004. *Generating Discourse Structures for Written Texts*. In Proceedings of the 20th International Conference on Computational Linguistics, pages 329– 335.

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 Method in Natural Language Processing, Vol. 1, EMNLP'09, pages 343-351.

William Mann and Sandra Thompson. 1988. *Rhetorical Structure Theory: Toward a Functional Theory of Text Organization*. Text, 8(3):243–281.

Daniel Marcu. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press, Cambridge, MA, USA.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. *Online Large-Margin Training of Dependency Parsers*, 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005) .

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. *Non-projective Dependency Parsing using Spanning Tree Algorithms*, Proceedings of HLT/EMNLP 2005.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. *Automatic Sense Prediction for Implicit Discourse Relations in Text*, In Proc. of the 47th ACL. pages 683-691.

Rashmi Prasad, Eleni Miltsakaki, Nikhil Dinesh, Alan Lee, Aravind Joshi, Livio Robaldo, and Bonnie Webber. 2007. *The Penn Discourse Treebank 2.0 Annotation Manual*. The PDTB Research Group, December.

David Reitter. 2003. *Simple Signals for Complex Rhetorics: On Rhetorical Analysis with Rich-feature Support Vector Models*. LDV Forum, 18(1/2):38–52.

Kenji Sagae. 2009. *Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing*. In Proceedings of the 11th International Conference on Parsing Technologies, pages 81-84.

Radu Soricut and Daniel Marcu. 2003. *Sentence level discourse parsing using syntactic and lexical information*. In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, Volume 1, pages 149–156.

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, pages 566–574.

Robert Endre Tarjan, 1977. *Finding Optimum Branchings*, *Networks*, v.7, pp.25-35.

Ben Taskar, Carlos Guestrin and Daphne Koller. 2003. *Max-margin Markov Networks*. In Proc. NIPS.

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

Wen Ting Wang, Jian Su and Chew Lim Tan. 2010. *Kernel based Discourse Relation Recognition with Temporal Ordering Information*, In Proc. of ACL'10. pages 710-719.

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 Proc.of the 7th SIGDIAL Workshop on Discourse and Dialogue. pages 117-125.