Joint Syntacto-Discourse Parsing and the Syntacto-Discourse Treebank

Abstract

Discourse parsing has long been treated as a stand-alone problem independent from constituency or dependency parsing. Most attempts at this problem are pipelined rather than end-to-end, sophisticated, and not self-contained: they assume gold-standard text segmentations (Elementary Discourse Units), and use external parsers for syntactic features. In this paper we propose the first end-to-end discourse parser that jointly parses in both syntax and discourse levels, as well as the first syntacto-discourse treebank by integrating the Penn Treebank with the RST Treebank. Built upon our recent span-based constituency parser, this joint syntacto-discourse parser requires no preprocessing whatsoever (such as segmentation or fea- ture extraction), achieves the state-of-the- art end-to-end discourse parsing accuracy.

1 Introduction

Distinguishing the semantic relations between segments in a document can be greatly beneficial to many high-level NLP tasks, such as summarization, sentiment analysis, question answering, and textual quality evaluation.

There has been a variety of research on discourse parsing. But most of them suffer from the following limitations:

1. pipelined rather than end-to-end: they assume pre-segmented discourse, and worse yet, use gold-standard segmentations
2. not self-contained: they rely on external syntactic parsers and pretrained word vectors;
3. complicated: they design sophisticated features, including those from parse-trees.

We argue for the first time that discourse parsing should be viewed as an extension of, and be performed in conjunction with, constituency parsing. We propose the first joint syntacto-discourse tree- bank, by unifying constituency and discourse tree representations. Based on this, we propose the first end-to-end incremental parser that jointly parses at both constituency and discourse levels. Our algo- rithm builds up on the span-based parser; it employs the strong general- ization power of bi-directional LSTMs, and parses efficiently and robustly with an extremely simple span-based feature set that does not use any tree structure information.

We make the following contributions:

1. We develop a combined representation of constituency and discourse trees to facilitate parsing at both levels without explicit conver- sion mechanism. Using this representation, we build and release a joint treebank based on the Penn Treebank and RST Treebank.
2. We propose a novel joint parser that parses at both constituency and discourse levels.
3. Even though it simultaneously performs con- stituency parsing, our parser does not use any explicit syntactic feature, nor does it need any binarization of discourse trees, thanks to the powerful span-based framework.

4. Empirically, our end-to-end parser outperforms the existing pipelined discourse pars- ing efforts. When the gold EDUs are pro- vided, our parser is also competitive to other existing approaches with sophisticated fea- tures.

2 Combined Representation & Treebank

We first briefly review the discourse structures in Rhetorical Structure Theory, and then discuss how to unify discourse and constituency trees, which gives rise to our syntacto-discourse treebank PTB-RST.

2.1 Review: RST Discourse Structures

In an RST discourse tree, there are two types of branchings. Most of the internal tree nodes are binary branching, with one nucleus child containing the core semantic meaning of the current node, and one satellite child semantically decorating the nucleus. Like dependency labels, there is a relation annotated between each satellite-nucleus pair, such as "Background" or "Purpose". There are also non- binary-branching internal nodes whose children are conjunctions, e.g., a "List" of semantically similar EDUs (which are all nucleus nodes).

2.2 Syntacto-Discourse Representation

It is widely recognized that lower-level lexical and syntactic information can greatly help determining both the boundaries of the EDUs (i.e., dis- course segmentation) as well as the semantic relations between EDUs. While these previous approaches rely on pre-trained tools to provide both EDU segmentation and intra-EDU syntactic parse trees, we in- stead propose to directly determine the low-level segmentations, the syntactic parses, and the high- level discourse parses using a single joint parser. This parser is trained on the combined trees of constituency and discourse structures.

We first convert an RST tree to a format similar to those constituency trees in the Penn Treebank. For each binary branching node with a nucleus child and a satellite child, we use the relation as the label of the converted parent node. The nucleus/satellite relation, along with the direction (either $\leftarrow$ or $\rightarrow$, pointing from satellite to nucleus) is then used as the label. For a conjunctive branch (e.g. "List"), we simply use the relation as the label of the converted node.

After converting an RST tree into the constituency tree format, we then replace each leaf node (i.e., EDU) with the corresponding syntactic (sub)tree from PTB. Given that the sentences in the RST Treebank is a subset of that of PTB, we can always find the corresponding constituency subtrees for each EDU leaf node. In most cases, each EDU corresponds to one sin- gle (sub)tree in PTB, since the discourse bound- aries generally do not conflict with constituencies. In other cases, one EDU node may correspond to multiple subtrees in PTB, and for these EDUs we use the lowest common ancestor of those subtrees in the PTB as the label of that EDU in the con- verted tree. E.g., if C–D is one EDU in the PTB tree A it might be converted to Purpose$\rightarrow$DCB A based on the Penn Treebank and RST Treebank. This PTB-RST treebank is released as a set of tools to generate the joint trees given Penn Tree- bank and RST Treebank data. During the align- ment between the RST trees and the PTB trees, we only keep the common parts of the two trees.

We follow the standard training/testing split of the RST Treebank. In the training set, there are 347 joint trees with a total of 17,837 tokens, and the lengths of the discourses range from 30 to 2,199 tokens. In the test set, there are 38 joint trees with a total of 4,819 tokens, and the lengths vary from 45 to 2,607. Figure 3 shows the distribution of the discourse lengths over the whole dataset, which on average is about 2x of PTB sen- tence length, but longest ones are about 10x the longest lengths in the Treebank.

3 Joint Syntacto-Discourse Parsing

Given the combined syntacto-discourse treebank, we now propose a joint parser that can perform end-to-end discourse segmentation and parsing.

3.1 Extending Span-based Parsing

As mentioned above, the input sequences are sub- stantially longer than PTB parsing, so we choose linear-time parsing, by adapting a popular greedy constituency parser, the span-based constituency parser.

3.2 Joint PTB-RST Treebank

Using the conversion strategy described above we build the first joint syntacto-discourse treebank.

As in span-based parsing, at each step, we main- tain a a stack of spans. Notice that in conventional incremental parsing, the stack stores the subtrees constructed so far, but in span-based constituency parsing, the stack only stores the boundaries of subtrees, which are just a list of indices ...i k j. In other words, quite shockingly, no tree structure is represented anywhere in the parser.

Similar to span-based constituency parsing, we alternate between structural (either shift or combine) and label (labelX or nolabel) actions in an odd-even fashion. But different from previous work, after a structural action, we choose to keep the last branching point k, i.e., i k j (mostly for combine, but also trivially for shift). This is because in our parsing mechanism, the dis- course relation between two EDUs is actually de- termined after the previous combine action. We need to keep the splitting point to clearly find the spans of the two EDUs to determine their relations. This midpoint k disappears after a label ac- tion; therefore we can use the shape of the last span on the stack (whether it contains the split point, i.e., i k j or i j) to determine the par- ity of the step and thus no longer need to carry the step z in the state .

The nolabel action makes the binarization of the discourse/constituency tree unnecessary, because nolabel actually combines the top two spans on the stack into one span, but without annotating the new span a label. This greatly simplifies the pre- processing and post-processing efforts needed.

	Prec.	Recall	F1
Constituency	87.6	86.9	87.2
Discourse	46.5	40.2	43.0
Overall	83.5	81.6	82.5

Table 1: Accuracies on PTB-RST at constituency and discourse levels.

3.3 Recurrent Neural Models and Training

The scoring functions in the deductive system are calculated by an underlying neu- ral model, which is similar to the bi-directional LSTM model that evaluates based on span boundary features. Again, it is important to note that no discourse or syntactic tree structures are represented in the features.

During the decoding time, a document is first passed into a two-layer bi-directional LSTM model, then the outputs at each text position of the two layers of the bi-directional LSTMs are con- catenated as the positional features. The spans at each parsing step can be represented as the fea- ture vectors at the boundaries. The span features are then passed into fully connected networks with softmax to calculate the likelihood of performing the corresponding action or marking the cor- responding label.

We use the "training with exploration" strategy and the dynamic oracle mechanism to make sure the model can handle unseen parsing configurations properly.

4 Experiments

We use the treebank described in Section 2 for empirical evaluation. We randomly choose 30 documents from the training set as the development set.

We tune the hyperparameters of the neural model on the development set. For most of the hyperpa- rameters we settle with the same values sug- gested previously. To alleviate the overfitting problem for training on the relative small RST Treebank, we use a dropout of 0.5.

One particular hyperparameter is that we use a value to balance the chances between training following the exploration (i.e., the best action cho- sen by the neural model) and following the correct path provided by the dynamic oracle. We find that = 0.8, i.e., following the dynamic oracle with a probability of 0.8, achieves the best performance. One explanation for this high chance to follow the oracle is that, since our combined trees are significantly larger than the constituency trees in Penn Treebank, lower makes the parsing easily divert into wrong trails that are difficult to learn from.

Since our parser essentially performs both constituency parsing task and discourse parsing task. We also evaluate the performances on sentence constituency level and discourse level separately. The result is shown in Table 1. Note that in constituency level, the accuracy is not directly comparable with the accuracy reported previously, since: a) our parser is trained on a much smaller dataset (RST Treebank is about 1/6 of Penn Treebank); b) the parser is trained to optimize the discourse-level accuracy.

Table 2 shows that, in the perspective of end- to-end discourse parsing, our parser first outper- forms the state-of-the-art segmentator, and furthermore, in end-to-end pars- ing, the superiority of our parser is more pronounced comparing to the previously best parser.

On the other hand, the majority of the conven- tional discourse parsers are not end-to-end: they rely on gold EDU segmentations and pre-trained tools like Stanford parsers to generate features. We perform an experiment to compare the per- formance of our parser with them given the gold EDU segments (Table 3). Note that most of these parsers do not handle multi-branching discourse nodes and are trained and evaluated on binarized discourse trees, so their performances are actually not directly comparable to the results we reported.

	description	syntactic feats.	segmentation	structure	+nuclearity	+rela
	segmentation only	Stanford	95.1	-	-	
	end-to-end pipeline	Penn Treebank	94.0	72.3	59.1	47
joint syntactic & discourse parsing	-	95.4	78.8	65.0	52.2	

Table 2: F1 scores of end-to-end systems. "+nuclearity" indicates scoring of tree structures with nucle- arity included. "+relation" has both nuclearity and relation included (e.g., ←Elaboration).

		syntactic feats	structure	+nuclearity	+relation
human	annotation	-	88.7	77.7	65.8
6*sparse		Penn Treebank	83.0	68.4	54.8
		Charniak (retrained)	82.7	68.4	55.7
		Charniak (retrained)	-	-	57.3
		Stanford	85.7	71.0	58.2
		ZPar (retraied)	83.5	68.1	55.1
		Stanford	86.0	72.4	59.7
5*neural			82.4	69.2	56.8
	+ sparse features	Stanford	84.0	70.8	58.6
		MALT	80.5	68.6	58.3
	+ sparse features	MALT	81.6	71.1	61.8
	span-based discourse parsing	-	84.2	67.7	56.0

Table 3: Experiments using gold segmentations. The column of "syntactic feats" shows how the syntactic features are calculated in the corresponding systems. Note that our parser predicts solely based on the span features from bi-directionaly LSTM, instead of any explicitly designed syntactic features.

5 Conclusion

We have presented a neural-based incremental parser that can jointly parse at both constituency and discourse levels. To our best knowledge, this is the first end-to-end parser for discourse parsing task.

Our parser achieves the state-of-the-art per- formance in end-to-end parsing, and unlike previ- ous approaches, needs little pre-processing effort.