Window-Based Neural Tagging for Shallow Discourse Argument Labeling

René Knaebel, Manfred Stede
Applied Computational Linguistics
Department of Linguistics
University of Potsdam
Germany
{rknaebel,stede}@uni-potsdam.de

Sebastian Stober
Artificial Intelligence Lab
Faculty of Computer Science
Otto von Guericke University Magdeburg
Germany
stober@ovgu.de

Abstract

This paper describes a novel approach for the task of end-to-end argument labeling in shallow discourse parsing. Our method describes a decomposition of the overall labeling task into subtasks and a general distance-based aggregation procedure. For learning these subtasks, we train a recurrent neural network and gradually replace existing components of our baseline by our model. The model is trained and evaluated on the Penn Discourse Treebank 2 corpus. While it is not as good as knowledge-intensive approaches, it clearly outperforms other models that are also trained without additional linguistic features.

1 Introduction

Shallow discourse parsing (SDP) is a challenging problem in NLP with the aim to identify local coherence relations in text. Discourse relations are used in text to connect individual segments of text logically. The Penn Discourse Treebank (PDTB) (Prasad et al., 2008a) adopts a non-hierarchical view on discourse relations. As an example from the PDTB, the sentence:

- *We would stop index arbitrage* when **the market is under stress**.

contains an explicit discourse relation which is signaled through the underlined connective (`Conn`) and further consists of two arguments (`Arg1` in italics and `Arg2` in bold). In addition, a sense is assigned to a relation, such as `Condition`.

Discourse analysis, however, does not work on a sentence-level, but takes full documents into account. Often, short paragraphs suffice to show the challenge in extracting overlapping relations. To illustrate the problem, we split the paragraphs shown in Figure 1 into small text chunks. Though our implementation works on the level of individual tokens, we here use these chunks to highlight challenges in discourse argument labeling. The chunks are delimited such that no smaller part would play exactly the same role for the various relations involved.

If you think you have stress-related problems on the job, there's good news and bad news. You're probably right, and you aren't alone.

...

Even the courts are beginning to recognize the link between jobs and stress-related disorders in compensation cases, according to a survey by the National Council on Compensation Insurance. But although 56% of the respondents in the study indicated that mental-health problems were fairly pervasive in the workplace, there is still a social stigma associated with people seeking help.

Figure 1: Excerpt from text WSJ 1582 (PDTB corpus)

Text Chunk	R1	R2	R3
If	`Conn`	`Arg1`	
you think you have stress-related problems on the job,	`Arg2`	`Arg1`	
there's good news and bad news.	`Arg1`	`Arg1`	
You're probably right,		`Arg2`	`Arg1`
and		`Arg2`	`Conn`
you aren't alone		`Arg2`	`Arg2`

Table 1: First sample paragraph split into text chunks.

The first example (cf. Table 1) shows that (1) each relations argument contains an arbitrary number of tokens. Further, (2) chunks may have multiple functions (class labels) referring to different individual relations, e.g. the "if" of the first chunk is the connective of **R1**, while in **R2** it is part of the first argument. As a special case, (3) they can have the same class label but pointing to different relations. Finally, (4) there is no generally fixed linear order for the classes `Arg1`, `Arg2`, `Conn`, although

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 768–777
Hong Kong, China, November 3-4, 2019.

Text Chunk	R4	R5
Even the courts are beginning to recognize the link between jobs and stress-related disorders in compensation cases, according to a survey by the National Council on Compensation Insurance.	Arg1	
But	Conn	Arg1
although		Conn
56% of the respondents in the study indicated that mental-health problems were fairly pervasive in the workplace	Arg2	Arg2
there is still a social stigma associated with people seeking help.	Arg2	Arg1

Table 2: Second sample paragraph split into text chunks.

the connective is always syntactically integrated with the second argument.

In the second example (cf. Table 2), (5) arguments can cover whole sentences as demonstrated by **R4**. (6) Although two connectives are next to each other, they can have different arguments and thus constitute different relations. Finally, (7) arguments can also consist of non-continuous chunks as in **R5**.

SDP consists of the main tasks of identifying connectives, demarcating their arguments, assigning senses to them, and finding the senses of so-called *implicit relations* holding between adjacent text spans, which are not explicitly signaled by a connective. Lin et al. (2014) presented a full end-to-end shallow discourse parser that solves these subtasks with a sequential pipeline architecture, which served as a model for the vast majority of follow-up work. Recently, however, most work has addressed specifically the last-mentioned task of identifying the senses of implicit relations, which has been found to be by far the most challenging one.

The focus of our work, in contrast, is on identifying and delimiting the arguments of relations as well as the disambiguation of connectives. Our aim is to do this without any engineering of linguistic features, so that the approach can be easily applied to new corpora and new languages. With this perspective, we follow in particular the proposals of Wang et al. (2015) and Hooda and Kosseim (2017). The first work applies a recurrent neural network on selected sentences and labels these sentences on a token-level. The second work extends this idea and uses an LSTM on a restricted form of the argument labeling task. The authors show the feasibility of a neural model for explicit argument labeling on pre-extracted argument spans. They prepare a dataset of argument spans (extracted from their context) and train a recurrent neural network to label each token's position in such a span .

In our work, we extend this idea to make it applicable within the full SDP setting, i.e., on running text rather than on previously extracted individual relations. As a baseline approach, we use our reimplementation of the system of Lin et al. (2014). We study different applications of our neural model and substitute corresponding components for argument extraction from the baseline pipeline: First we address extracting the arguments of connectives that are already given; this is a sensible assumption since models for connective classification (Pitler and Nenkova, 2009) work quite well. Then, we extend this approach by removing the dependency on previously identified connectives. This step is not easy, because the connectives serve to identify the number of explicit relations in a document. Because the number of relations is initially not clear when connectives are missing, we adapt a sliding window approach for decomposing the text in overlapping windows. We then develop a process of identical *prediction* steps (one for each possible window within a document) and one final *aggregation* step, which combines the individual results into the final set of predicted relations. As an outlook, we study the capacity of our neural model for the joint prediction of explicit and implicit relation arguments.

The main contributions of this paper are

1. integrating a BiLSTM model into the shallow discourse pipeline architecture, and

2. addressing the problem of jointly predicting connective and arguments with a moving-window approach for handling overlapping relations in running text.

In the following, Section 2 discusses relevant related work, and Section 3 explains our method. The experiments and results are presented in Section 4, followed by a discussion in Section 5 and conclusions in Section 6.

2 Related Work

The task of shallow discourse parsing was initiated by the development of the second version

of the Penn Discourse Treebank (PDTB2) (Prasad et al., 2008b) and further by the shared tasks at CoNLL 2015 and 2016 (Xue et al., 2015, 2016). Successful systems at these competitions were those of Wang et al. (2015); Wang and Lan (2016); Oepen et al. (2016). They followed the pipeline model of (Lin et al., 2014), which consists of successive tasks of connective identification, argument labeling, and sense classification for both explicit and implicit relations. Similar to other approaches used for argument extraction (e.g., (Wang et al., 2015; Laali et al., 2016; Oepen et al., 2016)), these competing systems use standard supervised machine learning models in combination with handcrafted linguistic features, such as syntactic, positional, and lexical features.

For argument labeling (or 'extraction'), the exact boundaries of both arguments must be identified. At the CoNLL shared task 2016, Stepanov and Riccardi (2016) reported the best scores for argument extraction with F-measures of 49.64% for `Arg1` and 76.51% for `Arg2`. In contrast to the systems mentioned before, their approach involves conditional random fields (CRF) (Lafferty et al., 2001), which are generally popular for the task of sequence labeling. Also, Ghosh et al. (2011b,a) formulate argument extraction as a token-level sequence labeling problem and use a pipeline of cascaded conditional random fields to mark up each token in a window. On top of a connective classifier, they first predict `Arg2` due to the closeness to the connective. For the prediction of `Arg1`, they use the same feature set as for the former prediction and additionally take the `Arg2` predictions into account. Similar to us, they use a window around the connective (2 sentences before and after the sentence with the connective). They report scores of about 79% F-measure for `Arg2` and 57% F-measure for `Arg1` with their approach. Our approach differs from theirs in that we use tokens instead of full sentences to create windows. This is necessary because a single sentence might contain multiple connectives and participate in more than one discourse relation (cf. the examples shown in Section 1).

A moving-window approach similar to ours is used, for example, by Graves and Schmidhuber (2005) in their work on phoneme classification. The task is quite different for SDP argument labeling, however, as a word's label highly depends on the word's context and the corresponding relation the word is associated with. Thus, we cannot apply such an approach directly and hence define an aggregation procedure for combining the individual per-window predictions.

Argument labeling with recurrent neural networks was done by Wang et al. (2015) in their DCU parser. In addition to word embeddings, they also used hand-crafted features, such as POS tags, syntactic relations, and lexical features. In contrast, our aim is to explore to what extent the problem can be solved *without* feature engineering. Thus, the main inspiration for our approach is the recent work of Hooda and Kosseim (2017), who use an LSTM network on spans of text for labeling arguments. The authors examine their approach on pre-extracted argument spans where the size of spans is determined by the maximal argument pair distance. This shows the feasibility of neural networks to label discourse arguments in a restricted problem setting. Like in our work, they do not rely on additional data other than word embeddings. In contrast to them, we use the embeddings as they are provided, without further adaption throughout the training. In our experiments, we could not identify gains in performance and thus save computation time by reducing trainable parameters.

3 Method

The main goal of our work is to replace existing components in the general discourse parser pipeline framework with our neural model. Depending on which component we want to substitute, we need different methods for processing the discourse.

Our first approach is to replace the argument extraction module, which operates on the basis of previously identified connectives. After that, we introduce an extended model that jointly predicts connectives and their arguments. As demonstrated in Section 1, the main challenge in this task is that a text contains multiple, potentially overlapping, relations that have to be predicted.

Our approach is to decompose the text (and thereby the SDP task) into a series of smaller texts, viz. into a sequence of overlapping windows. For each window, the statistical model is trained to recognize a possibly partial relation. Afterwards, a final aggregation process combines the individual window predictions and thus realizes the argument extraction for a full text.

We describe the baseline discourse parser in

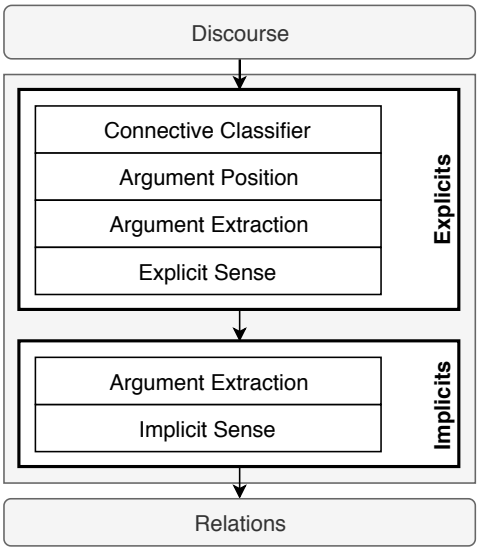

Figure 2: Baseline architecture proposed by Lin et al. (2014) that has been used throughout the experiments.

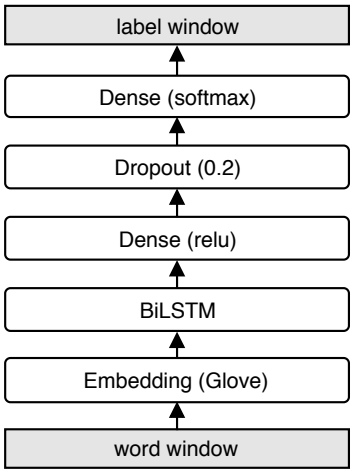

Figure 3: Bidirectional LSTM model architecture. First, tokens in a window are processed sequentially using the recurrent network. Then, each time step is transformed independently using one dense layer for transformation and one dense layer for the final prediction.

Section 3.1. The window model for predicting just arguments is explained in Section 3.2, and its extension to also handle connectives in Section 3.3. Thereafter, we turn to more training and evaluation details in Section 4.

3.1 Baseline: Shallow Discourse Parser

Our baseline discourse parser is inspired by the architecture of the (Lin et al., 2014) parser and consists of several components within a pipeline (see Figure 2). First, explicit relations are identified by classifying connectives, deciding the relative position of the first argument (whether it is contained in the previous sentence or the same sentence as the second argument), then delimiting the arguments, and finally recognizing the sense. In a second phase, implicit relations are classified between adjacent sentences where no explicit relations were found. For the argument extraction component, Lin et al. use a constituent-level approach by iterating over possible subtrees within a sentence and selecting the most likely `Arg1` and `Arg2`.

3.2 Window-based Argument Prediction

For our first approach, we propose a neural network as shown in Figure 3 to predict discourse relations given a sequence of words. Specifically, it operates on a window of words, which we build around a connective that we assume to have been identified by a previous module in the pipeline. Then, for each token the prediction task is defined as a four-way classification problem, i.e., the label is one of `None`, `Arg1`, `Arg2`, or `Conn`[1], as in the work by Hooda and Kosseim (2017) and similar to the window-based approach of Ghosh et al. (2011a).

Each word in the input sequence is embedded into lower-dimensional space. Because of the small size of the PDTB corpus, we use pretrained word embeddings, and these are further processed with a bidirectional Long Short-Term Memory network (BiLSTM). LSTMs have shown better performance compared to simple recurrent neural networks due to their higher capacity for storing important information over longer distances. Further improvements are gained through the bidirectional processing of sequential information (Graves and Schmidhuber, 2005). We keep hidden states for each time step and propagate them to the next layer. For our *BiLSTM* model, the hidden states are processed independently by a dense layer and finally are given to the top output layer (see Figure 3). Between the two dense layers, an additional dropout layer is used for better generalization.

3.3 Joint Prediction of Arguments and Connectives

To apply our model on a text without pre-identified connectives (i.e., to jointly predict connectives and their arguments), we need to adapt training and in-

[1]Notice that while the previously-given connective was used to define the position of the window, we still predict connectives in this model in order to support argument identification (but we will not evaluate the performance on connectives in this model).

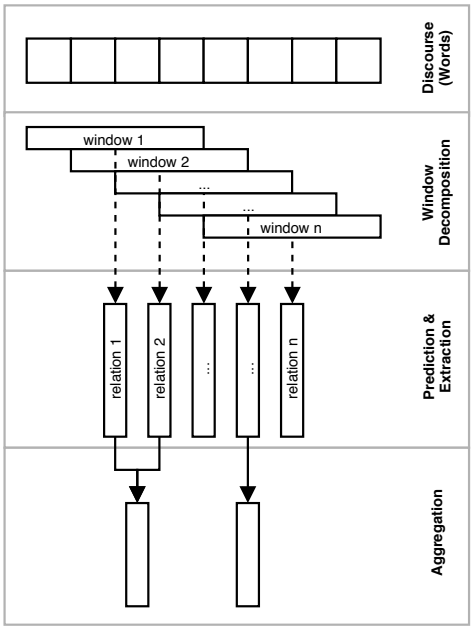

Figure 4: Overview of the proposed method consisting of decomposition, prediction, and aggregation.

ference, as the number of discourse relations and their positions are not known in this scenario. Also, as shown in the introductory example in Table 1, assigning classes to tokens is not a global decision, but has to be decided locally depending on the relation currently predicted. For this reason, we introduce the decomposition of the task into (i) handling multiple overlapping windows and (ii) a subsequent aggregation of the window-level predictions into the final set of predicted relations. The challenge in this aggregation approach is to identify valid predictions and further combine multiple possibly contradictory predictions pointing to the same relation. While we restrict our work here to explicit relations, we will show in our last experiment in Section 4.6 that this approach can also be applied to the case of implicit relations, merely by changing the data that the model is trained on.

3.3.1 Decomposition

The first part of our joint prediction method is to decompose the text into multiple overlapping windows that contain token embeddings. This is done by, first, padding the text at the beginning and at the end such that each token represents the center of one created window.

Then, for each window, we predict whether it contains a (possibly partial) relation or not, and which tokens belong to a particular part of the relation. Thus, for each token our model predicts one of four classes, `None`, `Arg1`, `Arg2`, or `Conn`.

We illustrate the process using the first example from Table 1. We cannot simply split the paragraph into windows and assign classes, because individual tokens have different labels depending on the context. Therefore, we choose the centering of the beginning of `Arg2` as a sufficient criterion (for generality to potentially cover both explicit and implicit relations, we did not choose the connective position as identifier). Thus, we extract three windows around the words "[if][you][think]", "[.][you]['re]", and "[and][you][are]". Because it is possible that a window's center is placed at the beginning or the end of a text, those have to be padded for having the same size as other complete windows.

The main challenge in jointly predicting connectives and their arguments is caused by multiple relations with overlapping arguments. To handle this, we create a unique identifier for each relation by using the position where the second argument of a relation begins. This forces the model to recognize relations only in the case when their second argument is centered in the window, and otherwise ignore them as being not in the focus of the current window.

For training the model, the data is prepared in such a way that it satisfies the properties above. For each relation instance in the PDTB, we use a fixed-size window that is placed on the text such that the relation has `Arg2` centered in this window. To further augment the data, we do not only use perfectly centered relations but additionally move the window one and two words to the left and to the right. Thus, for each relation in a text, we create five training instances. For each window, we keep the words and their argument labels, i.e., whether for a particular relation, a word belongs to one of the arguments, to the connective, or to neither of them. These windows, which correspond to a relation, are collected as positive training samples. Additionally, all windows that have not been extracted so far are gathered as negative training samples. For these negative training samples, although they might identify relations partially, all word labels are set to `None`. In this way, we want to force the model to learn to identify only relations for which `Arg2` is centered within a window.

In short, this process tries to include both arguments of the relation within a single window, but for longer arguments, this is not guaranteed. In this

approach, relations where an argument's span is beyond the window size remain incomplete for the training procedure.

3.3.2 Aggregation

After training a model for individual windows, the remaining challenge is to assemble the final set of relation predictions from all individual window predictions. We can distinguish the following cases: A window can contain

- only `None` labels (and thus no relation is to be extracted at all), or
- an invalid relation, where one or both arguments are empty, or
- a partial relation, where one or both of the arguments is incomplete, or
- a completely predicted relation.

To explain the aggregation step in detail, we define relations more formally and then describe the aggregation of individual relations based on their distance.

A relation is defined as a tuple $\mathrm{R}(a_1, a_2, c)$ of three sets, one for the first argument, the second argument, and the connective. Each of these sets contains the position indices of the included words, and hence these three sets are, by definition, disjoint. We maintain this property throughout the aggregation process of two relations.

Given two relations r_1, r_2, we define the merge of two relations, $r = r_1 \cup r_2$, as follows:

$$r^c = (r_1^c \cup r_2^c) \tag{1}$$

$$r^{a_2} = r_1^{a_2} \cup r_2^{a_2} \setminus r^c \tag{2}$$

$$r^{a_1} = (r_1^{a_1} \cup r_2^{a_1}) \setminus (r^c \cup r^{a_2}) \tag{3}$$

The crucial step is to decide whether two relations that are labeled in two consecutive windows are to be merged, i.e., to decide whether they denote the same relation. To this end, we measure the *distance* between two relations and merge them if the distance is below a certain threshold. Here we make use of the Jaccard distance as a distance metric for sets (but other metrics could be used as well). The Jaccard distance is based on the Jaccard index, which measures the similarity between finite sets by comparing the union and intersection of those sets:

$$J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

	None	Arg1	Arg2	Conn
Exp1	70.68 %	14.43 %	13.79 %	1.10 %
Exp2	85.34 %	7.21 %	6.89 %	0.56 %
Exp3	82.78 %	8.60 %	8.37 %	0.25 %

Table 3: Class label ratio calculated on the extracted training windows.

We define the distance of two relations $J_{rel}(r_1, r_2)$ as the mean of the two Jaccard distances between the relation's arguments.

$$J_{rel}(r_1, r_2) = \frac{J(r_1^{a_1}, r_2^{a_1}) + J(r_1^{a_2}, r_2^{a_2})}{2}$$

Note that for the calculation of the distances of the second arguments, we regard the connective as part of second arguments. This gives some, but not too much, influence to the connectives on the distance measure, compared to the influence of the arguments.

4 Experiments and Results

As described above, we experiment with two tasks of different complexity. The first experiment is a simplification of the general argument labeling problem, where a connective has already been classified. Because the position of the window is thus already determined, no aggregation is necessary for this scenario. For the second and third experiment, the exact positions of relations are not given and thus we follow our sliding-window approach. In all our evaluations, we use precision, recall, and F1 score of exact matches, in order to be comparable with the previous work.

All models are trained for 25 epochs on the corresponding training set, specific for a certain task. We use pretrained word embeddings with 300 dimensions as described later in Section 4.1. Each LSTM has a hidden layer of size 512. The output of each LSTM is concatenated per step, thus, the output of the BiLSTM results in 1024 dimensions per step. The dense layer with ReLU activation function on top of the BiLSTM has 64 dimensions, and the dropout works with a 0.2 probability. Finally, all models are trained using the Adam optimizer (Kingma and Ba, 2015). Because the classes are unbalanced throughout the experiments (compare Table 3) the cross-entropy loss is weighted (King and Zeng, 2001) according to the per-class occurrences in the extracted training windows.

4.1 Word Embedding

After the extraction of a vocabulary from the training corpus, we identify each token by a unique index. Every word that is not listed in the vocabulary is substituted by a special index for unknown words.

Because of the small size of the corpus, it is common to use pretrained embeddings (Mikolov et al., 2013) instead of training them solely on the given training corpus. In our experiments, we use *global vectors* for word representations (GloVe) (Pennington et al., 2014), similar as done by Hooda and Kosseim (2017). In general, GloVe embeddings have yielded promising results for a variety of related tasks. Hence, in the present study, we use GloVe without comparatively evaluating different pretrained embeddings.

Since both models, the word embedding model and ours, are trained on different corpora, the vocabularies also differ. First, we initialize all words with a random embedding. Then, for each word that is found in the set of pretrained embeddings, we replace the random embedding by its pretrained equivalent.

A minor disadvantage of word embeddings is that they are learned on a syntactic level. This means that although two words look similar in terms of characters, they might have different meanings depending on the words in their context. As a consequence, we follow the idea of using LSTMs on top of the word embeddings in order to account for the context in the individual representation.

In contrast to Hooda and Kosseim (2017), we avoid dynamically retraining the word vectors throughout the training process. Our earlier experiments showed that in our setting, training the embeddings has no positive effect on the final result, and thus, by avoiding this step, we also reduce the number of parameters that have to be trained.

4.2 Data Preparation

In our experiments, we follow the CoNLL shared task on shallow discourse parsing and work on their prepared dataset. This dataset differs slightly from the original PDTB2 corpus which is caused by the merge of a few sense labels and the cleaning of the PDTB data. For each relation in the CoNLL corpus, we extract a fixed-size window where the second argument is centered, i.e., where `Arg2` starts in the middle of the window. To produce more training data, we also extract windows that are shifted by small margins (up to two positions) to the left and to the right. Each of these windows constitutes a positive example of a discourse relation the model should learn. Conversely, every window of the same size where `Arg2` is not centered is added as a negative example. Even though a negative example may contain parts of some arguments, the words are labeled with `None` in order to force the model to only recognize relations where `Arg2` starts in the middle of the window. We keep punctuation as part of the vocabulary, as it might capture relevant discourse information.

In our evaluations, all models are trained on a window size of 100 tokens. Based on the span lengths found in the corpus, shown in Table 4, we choose this window size, as it captures a solid majority (over 80%) of the relations.

	Explicit		Implicit	
Perc.	**Span Length**	**Distance**	**Span Length**	**Distance**
min	2	0	2	0
20 %	14	2	20	2
40 %	21	2	29	2
60 %	29	3	38	2
80 %	44	4	49	3
max	1167	987	437	249

Table 4: Statistics calculated from the CoNLL2016 dataset. Overview of span lengths containing both arguments and distances of arguments for specific percentiles.

	Precision	Recall	F1
	Explicits		
Conn	83.42	79.82	81.58
Arg1	28.75	27.51	28.12
Arg2	45.54	43.57	44.54
Arg1+Arg2	27.70	26.51	27.09
	Non-Explicits		
Arg1	67.85	36.05	47.08
Arg2	67.65	35.94	46.94
Arg1+Arg2	59.94	31.84	41.59
	All		
Conn	83.42	79.82	81.58
Arg1	51.32	34.89	41.54
Arg2	58.28	39.62	47.17
Arg1+Arg2	45.38	30.86	36.74

Table 5: Evaluation of our baseline pipeline architecture. [2]

4.3 Baseline

The baseline model is inspired by Lin et al. (2014) who proposed a pipeline approach of several components. For our comparison, we focus on those

Label	Precision	Recall	F1
Arg1	46.69	44.59	45.62
Arg2	68.94	65.83	67.35
Arg1+Arg2	48.16	45.99	47.05

Table 6: Exact match of explicit arguments for a given connective.

components that are responsible for argument extraction: the connective classifier, argument position classifier and argument extractor in the explicit case, as well as the extraction of non-explicit arguments (cf. Figure 2). In contrast to our own model, which works on token level prediction, Lin et al.'s argument extraction model works on subtree level.

The performance reported in Table 5 shows the evaluation scores following the CoNLL shared task for our reimplementation of the pipeline described by Lin et al. Although these numbers are lower than those reported for the original implementation (e.g. predicting both explicit arguments is 10% worse), they may serve as a comparison and indicate that a re-implemenation of this system is not trivial. The values give different perspectives on the data, a full evaluation (**All**), and more detailed views on both parts (**Explicits** and **Non-Explicits**), which correlate with the structure of the architecture.

4.4 Connective Arguments

In the first experiment, we train a model to substitute the components for argument position and argument extraction similar to Ghosh et al. (2011a). The training data contains only positive explicit samples, since the model is never applied to other situations than these. With our procedure described above, we extract 73.610 samples from the corpus. For inference, we use a fixed-size window centered around the previously identified connective. The labeled indices for `Arg1` and `Arg2` are taken without any further processing.

The results in Table 6 show an increase of the values for `Arg1` and `Arg2` compared to our baseline.

4.5 Explicit Argument Extraction

The second experiment generalizes the window-based model and predicts arbitrary explicit relations for a text. This is challenging compared to the former experiment, because the connective is missing and therefore the model does not know the exact position of a relation. For this reason, we introduced the decomposition of a discourse with the sliding window approach and further introduce an aggregation method to get the final set of predicted relations.

Label	Precision	Recall	F1
Conn	69.25	44.56	54.23
Arg1	36.52	23.50	28.59
Arg2	62.96	40.51	49.30
Arg1+Arg2	40.29	25.93	31.55

Table 7: Exact match of explicit arguments for joint prediction of connectives and their arguments.

	Precision	Recall	F1
	Explicits		
Conn	71.35	62.73	66.76
Arg1	33.16	29.15	31.03
Arg2	52.47	46.13	49.09
Arg1+Arg2	37.25	32.75	34.86
	Non-Explicits		
Arg1	41.99	29.26	34.49
Arg2	44.32	30.88	36.40
Arg1+Arg2	40.16	27.99	32.99
	All		
Conn	71.35	62.73	66.76
Arg1	40.95	31.77	35.78
Arg2	51.47	39.94	44.98
Arg1+Arg2	41.83	32.45	36.55

Table 8: Evaluation of the joint extraction of explicit and non-explicit arguments.

The training data additionally contains negative explicit samples in contrast to the first experiment. The same amount of negative instances as positive instances is sampled from all possible negative instances. We extract twice as much training instances as before.

As shown in Table 7, the scores for connective identification are not as high as achieved with a specialized model (as in the baseline). Further, the sores for argument extraction also decrease for `Arg2` slightly and for `Arg1` even more. This is probably caused by the unbalanced data, as `None` labels occur much more often than other labels (see Table 3).

4.6 Explicit/Implicit Arguments Extraction

In the final experiment, we examine the full capacity of our model by using it for jointly predicting both explicit and implicit relation arguments. The training data consists of explicit and implicit relation instances, and for each positive sample, one negative sample is added to the training data. In sum, the model is trained on 325.350 instances.

As expected, the number for identifying non-

explicit relations is lower than the baseline (see Table 8). In contrast to our model, the baseline always selects two adjacent unlabeled sentences, which achieves good results despite its simplicity. Our models seems unable to recognize any sensible underlying pattern.

5 Discussion

Usually, different specialized models are combined to label arguments, either on a constituent-level (Lin et al., 2014; Kong et al., 2014) or on a token-level (Ghosh et al., 2011a,b). As we explained earlier, though, a crucial difference is their usage of highly-engineered linguistic features (e.g., cue words, syntactic classes, word pair relations, production rules), which our system does not use. Instead, it relies solely on word (token) sequences encoded by pretrained embeddings, which thus represent a certain amount of semantic information.

Our comparison with the re-implemented pipeline architecture is quite weak, as our implementation does not perform as good as the original work that relies on manually-engineered rules.

While our approach can obviously not compete with the knowledge-intensive approaches of Oepen et al. (2016) and Stepanov and Riccardi (2016), a fair comparison is that to the similar approach of Wang et al. (2015); here, our system clearly outperforms the earlier result. They report exact match F-measure of 36.32% for `Arg1` and 41.70% for `Arg2`. Compared to that system, our first experiment's approach where the connective was given (Section 4.4) performs much better. In comparison with our second experiment, our system's scores are lower for `Conn` and `Arg1`, but still higher for `Arg2` and both arguments extraction. A big problem in joint connective–argument extraction is the limited amount of data and the unbalanced class labels. We tried to work on the second problem by using weighted losses based on the class occurrences.

6 Conclusions and Future Work

In this work, we integrate a BiLSTM model in the shallow discourse parsing framework. We described tasks of different complexity in argument labeling and explained different ways of applying our neural model. For the general task of argument labeling, we adapt a token-level window-based approach and introduce a novel aggregation method, which is needed for combining individual predictions into the final set of relation arguments. We explored the limits of this approach by studying the joint prediction of the arguments of explicit and implicit relations.

The aggregation of partial relations is done using a distance threshold. For gaining possible improvements on the final results, it may well be worth studying different methods for merging conflicting relation predictions as well as defining different ways of computing the distances of individual predictions.

Because of the simplified nature of our architecture, we think that there are further interesting potentials for future work. Formulating the new window-training problem makes it possible to easily replace our proposed model by other architectures with more capacity. At the same time, we see this way of solving argument extraction as one step toward reducing the complexity and degree of error propagation in the more traditional SDP pipeline architecture.

We release the source code of our system[3] to promote future research.

Acknowledgments

This research was supported by the Federal Ministry of Education and Research (BMBF), Contract 01IS17059.

References

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

Sucheta Ghosh, Sara Tonelli, Giuseppe Riccardi, and Richard Johansson. 2011b. End-to-end discourse parser evaluation. In *Fifth IEEE International Conference on Semantic Computing (ICSC), 2011; September 18-21, 2011; Palo Alto, United States*, pages 169–172.

A. Graves and J. Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052 vol. 4.

Sohail Hooda and Leila Kosseim. 2017. Argument labeling of explicit discourse relations using lstm

[3] www.github.com/rknaebel/discopy

neural networks. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 309–315. INCOMA Ltd.

Gary King and Langche Zeng. 2001. Logistic regression in rare events data. *Political Analysis*, 9:137–163.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Fang Kong, Hwee Tou Ng, and Guodong Zhou. 2014. A constituent-based approach to argument labeling with joint inference in discourse parsing. In *EMNLP*.

Majid Laali, Andre Cianflone, and Leila Kosseim. 2016. The clac discourse parser at conll-2016. In *Proceedings of the CoNLL-16 shared task*, pages 92–99. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Stephan Oepen, Jonathon Read, Tatjana Scheffler, Uladzimir Sidarenka, Manfred Stede, Erik Velldal, and Lilja Øvrelid. 2016. Opt: Oslo–potsdam–teesside. pipelining rules, rankers, and classifier ensembles for shallow discourse parsing. In *Proceedings of the CoNLL-16 shared task*, pages 20–26. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *ACL/IJCNLP*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008a. The penn discourse treebank 2.0. In *LREC 2008*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008b. The penn discourse treebank 2.0. In *In Proceedings of LREC*.

Evgeny Stepanov and Giuseppe Riccardi. 2016. Unitn end-to-end discourse parser for conll 2016 shared task. In *Proceedings of the CoNLL-16 shared task*, pages 85–91. Association for Computational Linguistics.

Jianxiang Wang and Man Lan. 2016. Two end-to-end shallow discourse parsers for english and chinese in conll-2016 shared task. In *Proceedings of the CoNLL-16 shared task*, pages 33–40. Association for Computational Linguistics.

Longyue Wang, Chris Hokamp, Tsuyoshi Okita, Xiaojun Zhang, and Qun Liu. 2015. The dcu discourse parser for connective, argument identification and explicit sense classification. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 89–94. Association for Computational Linguistics.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi Prasad, Christopher Bryant, and Attapol Rutherford. 2015. The conll-2015 shared task on shallow discourse parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 1–16. Association for Computational Linguistics.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Attapol Rutherford, Bonnie Webber, Chuan Wang, and Hongmin Wang. 2016. Conll 2016 shared task on multilingual shallow discourse parsing. In *Proceedings of the CoNLL-16 shared task*, pages 1–19. Association for Computational Linguistics.