Discourse Relation Prediction: Revisiting Word Pairs with Convolutional Networks

Siddharth Varia
Dept. of Computer Science
Columbia University
sv2504@columbia.edu

Chris Hidey
Dept. of Computer Science
Columbia University
ch3085@columbia.edu

Tuhin Chakrabarty
Dept. of Computer Science
Columbia University
tc2896@columbia.edu

Abstract

Word pairs across argument spans have been shown to be effective for predicting the discourse relation between them. We propose an approach to distill knowledge from word pairs for discourse relation classification with convolutional neural networks by incorporating joint learning of implicit and explicit relations. Our novel approach of representing the input as word pairs achieves state-of-the-art results on four-way classification of both implicit and explicit relations as well as one of the binary classification tasks. For explicit relation prediction, we achieve around 20% error reduction on the four-way task. At the same time, compared to a two-layered Bi-LSTM-CRF model, our model is able to achieve these results with half the number of learnable parameters and approximately half the amount of training time.

1 Introduction

Implicit discourse relation identification is the task of recognizing the relationship between text segments without the use of an explicit connective indicating the relationship. For instance, while a connective such as "because" may indicate a causal relationship when present between sentences, it is not necessary for causality (as in Example 1). Without the explicit connective, automatically identifying the relationship is much more difficult. Improvement in identifying implicit discourse relations will also improve performance in downstream tasks such as question answering, textual inference (for determining relationships between text segments), machine translation and other multi-lingual tasks (for transferring discourse information between languages).

The Penn Discourse Tree Bank (PDTB) theory of discourse relations (Prasad et al., 2008) defines a shallow discourse representation between adjacent or nearby segments. As a result, the span of the arguments participating in the discourse relation is often the most important input to a classifier.

Initial approaches used linguistically informed features derived from the arguments as inputs to traditional machine learning methods (Pitler et al., 2008). More recently, the application of neural methods has resulted in the best performance on this task, modeling the relationship between words in the arguments in context (Ji et al., 2015; Dai and Huang, 2018).

A common approach in prior work is to use pairs of words from across the arguments as features (Marcu and Echihabi, 2002; Blair-Goldensohn et al., 2007; Pitler et al., 2009). Consider the example:

> I am **late** for the meeting because the train was **delayed**. (1)

The words "late" and "delayed" are semantically related and (absent the connective) one might hypothesize that their presence is what triggers a causal relation. Therefore, pairs of words across discourse arguments should be useful features for identifying discourse relations. However, learning these specific word pairings requires leveraging large text corpora to observe them in the relevant discourse context (Biran and McKeown, 2013). Furthermore, as the number of possible word pairs grows quadratically with the size of the vocabulary, representing word pairs discretely results in very sparse feature sets. Since a continuous representation of the word pairs allows for better generalization to unseen pairs, we thus use a Convolutional Neural Network (CNN) to embed word pairs from the arguments in a dense vector representation. We also extend this idea of word pairs beyond a single pair of words by using larger filter sizes.

Our results show that these word pairs provide

Proceedings of the SIGDial 2019 Conference, pages 442–452
Stockholm, Sweden, 11-13 September 2019.

improved performance in transferring knowledge from explicit relations, indicating less sensitivity to word ordering. Finally, an additional advantage is that our architecture based on convolution layers allows for additional improvement in the speed of training through parallel processing unlike sequential models based on LSTMs.

Our primary contributions are as follows:

- A novel application of convolutional neural networks to model word pairs in the arguments in a discourse relation
- A demonstration that joint learning of implicit and explicit relations with both word pairs and n-grams improves performance over learning implicit relations only
- State-of-the-art results on four-way classification for both implicit and explicit relations, reducing the error by 20% in the latter case
- A model with half the number of learnable parameters compared to a state-of-the-art two-layered Bi-LSTM-CRF model along with approximately half the training time

2 Related Work

Previous work on discourse relations found success using word pairs as features. In the earliest work using word pairs, Marcu and Echihabi (2002) used unambiguous explicit markers such as "but" to create a corpus of discourse relations. They used a Naive Bayes approach by taking the cross-product of words on either side of the connective. Blair-Goldensohn et al. (2007) used word pairs for discourse relations as well. Later work (Pitler et al., 2009) applied this approach to the PDTB but found that the top word pairs were discourse connectives, which is counter-intuitive as connectives were removed to obtain word pairs. These earlier approaches use word pairs directly as features, which creates a large sparse feature space. In more recent work, Biran and McKeown (2013) address the sparsity issue by using features based on word pairs in the context of an explicit connective in the Gigaword corpus. Even though this approach addresses the sparsity issue by using a much larger corpus, it is still impractical to account for *every* possible word pair. In comparison to these previous word pair methods, our model takes advantage of the continuous representation of word embeddings to model similarity between word pairs.

In other work, researchers have found that approaches using neural networks have helped increase performance on this task, as neural models are better at dealing with sparsity. Some work has focused on using novel representations. Ji et al. (2015) model the arguments with recursive neural networks (modeling the tree structure of each argument). Lei et al. (2017) model interaction between words in arguments by learning linear and quadratic relations. Liu and Li (2016) develop a method for repeated reading over the discourse context by using an external memory. Other researchers have found success by modeling the sequence of words using recurrent neural networks (Chen et al., 2016) with a gating mechanism to combine contextual word pairs while some approaches have used convolutional neural networks over each argument (Qin et al., 2016). Most recent work has focused on joint learning, as the PDTB is a relatively small dataset for neural methods. Liu et al. (2016) and Lan et al. (2017) propose a multi-task learning approach across PDTB and other corpora. Qin et al. (2017) have demonstrated the effectiveness of an adversarial approach, forcing one model without connective information to be similar to a model with connective. Rönnqvist et al. (2017) developed the first attention-based BiLSTM network for Chinese implicit discourse relations. Dai and Huang (2018) show that incorporating additional document context at the paragraph level and jointly predict both implicit and explicit relations. Finally, Bai and Zhao (2018) propose a deep model using contextual ELMo embeddings, multiple CNN layers and Bi-attention. Unlike these approaches, we represent the input in a novel way as a set of word pairs, while using a much simpler architecture, and distill knowledge between explicit and implicit relations.

3 Methods

Our architecture consists of two primary components. The first component learns complex interactions from word pairs and the second component learns n-gram features from individual arguments. The features from word pair convolutions and individual argument convolutions are then combined using a gating mechanism. Finally, we jointly learn representations for implicit and explicit relations.

	the	train	was	delayed
late	late, the	late, train	late, was	late, delayed
for	for, the	for, train	for, was	for, delayed
the	the, the	the, train	the, was	the, delayed
meeting	meeting, the	meeting, train	meeting, was	meeting, delayed

Table 1: $Arg1$ is along the rows, $Arg2$ is across the columns. Cell (i, j) corresponds to the word pair composed from i^{th} word of $Arg1$ and j^{th} word of $Arg2$.

3.1 Product of Arguments

For the first component, we use convolution operations over the Cartesian product of words from the two arguments.

Word/Word Pairs Initially, we consider the interaction between all pairs of individual words in the arguments. Table 1 illustrates the use of word pairs from Example 1 in Section 1, where Arg_1 = "I am late for the meeting" and Arg_2 = "the train was delayed." The sequence of word pairs starts at the first row and moves on to successive rows. In this example, we remove the connective to illustrate an implicit relation. For explicit relations, we include the connective as part of the second argument and create word pairs for the connective (e.g. "because") as well. When computing the Cartesian product, we drop very short ($length < 3$) functional words to limit the number of word pairs (hence the absence of the words "I" and "am").

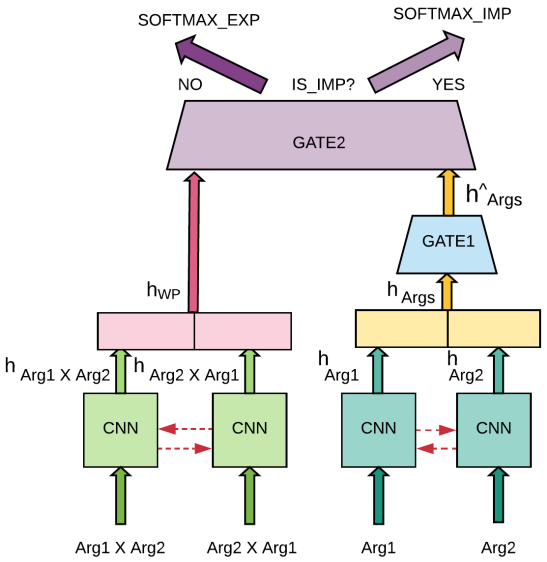

Figure 1: Architecture of our proposed network. Dashed arrows indicate that weights are shared among CNNs

Word/N-Gram Pairs We also use larger filters to capture relations between word pairs. A filter of size $2k$ will capture k word pairs. Henceforth we use the notation ***WP-k*** to indicate a sequence of k word pairs (where *WP-1* refers to a single pair of words). We use the following notation to describe concrete examples of word pair features: ("late" : "delayed") is an example of *WP-1*. Similarly ("late" : "the train was delayed") is an example of *WP-4* and corresponds to the following sequence of word pairs (as in row 1 of Table 1): "late the late train late was late delayed". In other words, it corresponds to the Cartesian product of "late" with the 4-gram "the train was delayed." Thus, another interpretation of *WP-k* is a mapping of a word in Arg_1 to a k-gram in Arg_2 and vice-versa. This interpretation of *WP-k* is not true at word transitions. For instance, in the above example, a filter of size 8 will also capture "late was late delayed for the for train" as one of the *WP-4*. By learning *WP-k* features (for $k > 1$) we are able to capture more complex interaction between the arguments. This is novel in comparison to the common practice of just using *WP-1* as features. We use filters of even length and a stride of two to ensure the filter will always end at word pair boundaries.

Mathematically, the input to the CNN (where $[\cdot]$ means concatenation) is:

$$\mathbf{v}_{Arg_1 \times Arg_2} = [x_1^1 \cdot x_2^1 \cdot x_1^1 \cdot x_2^2 \cdot x_1^1 \cdot x_2^3 \cdots]$$

where $\mathbf{x}_1^i$ is the concatenation of word and POS embeddings corresponding to the i^{th} word of Arg_1 and $\mathbf{x}_2^j$ is the concatenation of word and POS embeddings corresponding to the j^{th} word of Arg_2. We also include the representation obtained from $(Arg_2 \times Arg_1)$, as our preliminary experiments showed that this approach was complementary to the representation from $(Arg_1 \times Arg_2)$:

$$\mathbf{v}_{Arg_2 \times Arg_1} = [x_2^1 \cdot x_1^1 \cdot x_2^1 \cdot x_1^2 \cdot x_2^1 \cdot x_1^3 \cdots]$$

Following convolution, in line with the common practice, we apply max pooling along the length

of the sequence to pick the most prominent feature per feature map. Next we concatenate the max-pooled features from different filter sizes to obtain a hidden representation. This hidden representation from the CNN has dimensionality equal to the number of feature maps $\times$ the number of filters. Thus, for each discourse relation comprised of Arg_1 and Arg_2, we obtain two hidden representations $\mathbf{h}_{Arg_1 \times Arg_2}$ and $\mathbf{h}_{Arg_2 \times Arg_1}$. We concatenate these representations to obtain a vector $\mathbf{h}_{WP}$. The weights of the convolution layers are shared between $Arg_1 \times Arg_2$ and $Arg_2 \times Arg_1$ to allow the model to learn from both types of interaction. The left side of the Figure 1 depicts this component of our combined architecture.

3.2 Individual Arguments

For the second component, we use a CNN over *individual* arguments Arg_1 and Arg_2 (illustrated on the right side of Figure 1). As discussed in Rönnqvist et al. (2017), arguments provided without context may contain elements indicative of a discourse relation (Asr and Demberg, 2015), e.g. implicit causality verbs (Rohde and Horton, 2010). We thus hypothesize that the hidden representation obtained from individual arguments will complement the representation obtained from the word pairs. To learn representations from the individual arguments we use filters of odd and even length and stride equal to one.

As with word pairs, the weights of the convolution layers are shared between Arg_1 and Arg_2 to allow the model to learn representations from both sides independent of the order of the arguments.

3.3 Combination of Argument Representations

In order to combine the representations $\mathbf{h}_{Arg_1}$ and $\mathbf{h}_{Arg_2}$, we incorporate a method for the model to learn to weight the interaction between the argument features. We employ $Gate_1$ as shown on the right side of Figure 1. This gate is defined as follows:

$$\begin{aligned} c &= Relu(W_1 \cdot \mathbf{h}_{Args} + b_1) \\ g_a &= \sigma(W_2 \cdot \mathbf{h}_{Args} + b_2) \\ \hat{\mathbf{h}}_{Args} &= c \odot g_a \end{aligned} \tag{2}$$

where $\mathbf{h}_{Args}$ is the concatenation of $\mathbf{h}_{Arg_1}$ and $\mathbf{h}_{Arg_2}$.

We subsequently join the word pair representations and the individual argument representations with a similar mechanism. The two components are combined using $Gate_2$ which is defined as in Equation 2 but instead takes as input the concatenation of $\hat{\mathbf{h}}_{Args}$ and $\mathbf{h}_{WP}$

To predict the discourse relation, the output of $Gate_2$ is then input to a separate dense layer with softmax non-linearity for either explicit or implicit relation classification as shown in Figure 1.

3.4 Joint Learning of Implicit and Explicit Relations

Finally, to fully take advantage of the labeled data in the PDTB, we jointly learn implicit and explicit relations. For explicit relations, we add the connective to the beginning of Arg_2. As shown in Figure 1 and similar to (Dai and Huang, 2018), we use separate classification layers for explicit and implicit relations. To jointly learn both types of relations, we randomize the order of implicit or explicit relations rather than training each mini-batch separately.

4 Experiments

4.1 Data

We run experiments on three different tasks (binary, four-way and fifteen-way). For binary and four-way tasks, we train and test on the class level relations defined in the PDTB: Comparison, Contingency, Temporal, and Expansion. We use a common partition of the data: sections 2-20 for training, 0-1 for validation, and 21-22 for testing. For this partition, there are 1046 implicit relation instances and 1285 explicit relation instances in the test set. For fifteen-way task, we precisely follow the setup of CoNLL 2016 shared task on shallow discourse parsing and evaluate our approach on their test and blind test sets. A small fraction (around 4%) of the relations in PDTB have multiple gold labels. During training, we replicate a relation once for each gold label and for evaluation we deem the prediction to be correct if the predicted label matches any of the gold labels. We use this scheme for all the tasks in this paper.

4.2 Experimental setup

We use Spacy to tokenize and annotate POS tags for the individual arguments. To learn *WP-k* features, we limit the Cartesian product to a maximum of 500 word pairs per relation. For n-gram features, we limit the arguments to a maximum of 100 words. For word embeddings, we used

Model		Implicit					
		Macro F1	Acc	Com	Con	Exp	Tem
LSTM	(Lei et al., 2017)	46.46	-	-	-	-	-
	(Lan et al., 2017)	47.80	57.39	-	-	-	-
	(Dai and Huang, 2018)	- (48.82)	- (58.20)	- (37.72)	- (49.39)	- (68.86)	- **(40.70)**
CNN	(Liu et al., 2016)	44.98	57.27	-	-	-	-
	(Bai and Zhao, 2018)	51.06	-	-	-	-	-
Ours	WP-[1-4], Args, Implicit Only	50.77 (49.2)	59.46 (56.11)	45.82 **(42.1)**	54.39 (51.1)	70.48 (64.77)	43.04 (38.8)
	Args Joint Learning	49.47 (48.1)	59.66 (57.50)	42.68 (35.50)	54.82 **(52.5)**	70.30 (67.07)	41.82 (37.47)
	WP-1, Args, Joint Learning	50.71 (48.73)	59.18 (57.36)	45.91 (37.33)	**55.87** (52.27)	69.04 (66.61)	42.96 (38.70)
	WP-[1-4], Args, Joint Learning	**51.84 (50.2)**	**60.52 (59.13)**	**46.84** (41.94)	53.74 (49.81)	**72.42 (69.27)**	**43.97** (39.77)

Table 2: Results of four-way classification experiments on implicit relations. The numbers in the parenthesis correspond to average of 10 runs

Model	Explicit	
	Macro F1	Acc
(Dai and Huang, 2018)	- (93.70)	- (94.46)
Args, JL	**95.48** (94.81)	**96.2** (95.63)
WP-1, Args, JL	95.13 **(94.83)**	95.95 **(95.67)**
WP-[1-4], Args, JL	95.0 (94.50)	95.72 (95.33)

Table 3: Results of four-way classification experiments on explicit relations. JL : Joint Learning

word2vec pre-trained embeddings. However, for words not found in word2vec, we back-off to embeddings trained on the raw WSJ articles. We fix the word embeddings during training. We also concatenate one hot POS embeddings to the fixed word embeddings. We use 100 and 50 feature maps per filter size for learning *WP-k* and n-grams respectively. For *WP-k*, we use filters of size 2, 4, 6 and 8. For n-grams, we use filters of size 2, 3, 4 and 5. For all dense layers and gate layers, we set the output dimension of the weight matrices to 300. For regularization, we use dropout (Srivastava et al., 2014) of 0.5 after convolution operations and before the softmax layers. We also use L2 regularization with a coefficient of 0.0001 and early stopping to prevent over-fitting. For training, we minimize multi-class cross-entropy loss using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.0005 and batch size of 200. Our architecture is implemented in Theano deep learning framework.[1]

5 Results

We compare our results to previous work along two dimensions: the architecture of the model (CNN or LSTM) and whether the model employs a joint learning component.
Our work and the work of (Dai and Huang, 2018) and (Lan et al., 2017) involves jointly training on explicit relations. (Lan et al., 2017) and (Liu et al., 2016) also train on *BLLIP* and *RST*, respectively.

5.1 Four-way classification

Tables 2 & 3 show the results of four-way experiments on implicit and explicit relations respectively.[2] Please note that these results are from the same joint learning experiments wherever applicable and they are presented in different tables for the sake of better presentation. We compare the performance of our models under different configurations. We gradually add *WP-k* features to study their contribution. First we add *WP-1* (fil-

[1] https://github.com/siddharthvaria/WordPair-CNN
[2] (Qin et al., 2017) reported only one-versus-all binary classification so their results are only included in Table 4

Model		Com	Con	Exp	Tem
LSTM	(Lei et al., 2017)	40.47	55.36	69.50	35.34
	(Lan et al., 2017)	40.73	**58.96**	72.47	38.50
	(Dai and Huang, 2018)	- (**46.79**)	- (**57.09**)	- (70.41)	- (**45.61**)
CNN	(Liu et al., 2016)	37.91	55.88	69.97	37.17
	(Qin et al., 2017)	40.87	54.56	72.38	36.20
	(Bai and Zhao, 2018)	**47.85**	54.47	70.60	36.87
Ours	WP-[1-4], Args Joint Learning	45.03 (44.1)	56.53 (56.02)	**73.5** (**72.11**)	**46.15** (44.41)

Table 4: Results of binary classification experiments. The numbers in the parenthesis correspond to average of 10 runs

ters of size 2) and then we add *WP-k* features to illustrate the contribution of more complex interactions for $k > 1$. Additionally, we compare joint learning of implicit and explicit relations (*Joint Learning*) against learning implicit relations only (*Implicit only*). *Args* refers to n-grams from individual arguments. For all experiments, we report both the maximum and average (in parenthesis) of 10 runs for fair comparison with all prior work. It is not surprising to see that gradually adding word pair features improves performance on implicit relations. When using joint learning and *WP-[1-4]* we obtain an improvement in Macro F1-Score and Accuracy for implicit relations over previous state of the art works (Dai and Huang, 2018) and (Bai and Zhao, 2018). We also observe improvement for the expansion class in the joint learning setting, likely due to its prevalence in both implicit and explicit relations. In these cases, we observe that joint learning improves over training on just implicit relations (with a 3 point improvement in overall accuracy primarily due to a 5 point improvement in classification of the expansion class). On the other hand, we find that in just *Args* setting, we obtain state-of-the-art performance compared to prior work (Dai and Huang, 2018) for explicit relation F1-Score and accuracy, achieving 20% reduction in error rate. We don't get any benefit by combining it with word pairs (for *WP-1*), and the extra complexity for $k > 1$ makes it more difficult for the model to distinguish the effective features. This may occur because the connective itself is a very strong baseline.

5.2 Four-way Ensemble results

As the experiments described in 4.2 were conducted with 10 random initializations, we also present the results of an ensemble created out of these 10 runs via majority voting in Table 5. Compared to (Dai and Huang, 2018), while our ensemble achieves marginal improvement of 0.59 F1 on implicit relations, it improves by around 1 point on explicit relations for both metrics, around a 20% error reduction.

5.3 Binary classification

In Table 4, we report our results on four binary classification tasks. From the results, we see that our model does better on all classes in comparison to other CNN-based architectures. Our averaged results are directly comparable to those of (Dai and Huang, 2018) and we observe improvement for the expansion class. Our model may not generalize as well on other three classes because they account for 14, 26.4, and 6% of the test set, respectively, leading to high variance across multiple runs.

5.4 Fifteen-way classification

CoNLL organized a multilingual shallow discourse parsing shared task in 2016. In this shared task (Xue et al., 2016), they consider second level types and release test and blind test sets for fifteen-way classification of explicit and implicit relations, including EntRel and AltLex relations as implicit relations. We compare our architecture against the systems that participated in that task, with results presented in Table 6. Our architecture produces very similar results in line with the results reported by various neural network based systems that participated in the task. However, we also observe that using word pair features does not lead to further improvement over using just n-gram features. One possible explanation for this

Model	Implicit		Explicit	
	Macro F1	Acc	Macro F1	Acc
Dai et al. (2018)	51.84	59.85	94.17	94.82
WP-[1-4], Args, IO	51.63	58.03	-	-
Args, JL	49.54	58.70	94.81	95.64
WP-1, Args, JL	51.90	59.94	**95.16**	**95.95**
WP-[1-4], Args, JL	**52.53**	**61.28**	94.38	95.25

Table 5: Ensemble results of four-way classification experiments. JL : "Joint Learning" and IO : "Implicit Only"

Model	F1 score			
	Implicit		Explicit	
	PDTB	Blind	PDTB	Blind
Xue et al., (2016)	**40.91**	37.67	**90.22**	**78.56**
Lan et al., (2017)	39.40	**40.12**	-	-
Args, JL	39.68	38.74	89.91	76.98
WP-[1-4], Args, JL	39.39	39.36	89.48	77.00

Table 6: Results of fifteen-way task on CoNLL 2016 test and blind test sets

trend is that word pair features capture enough semantic information to discriminate the top-level classes however it fails to separate the second level of types. Comparing against (Lan et al., 2017), we see that our model is competitive with their LSTM-based architecture in spite of the fact that they used external data to achieve these results. This also possibly indicates that it is hard to get further improvements on this task without data augmentation due to lack of enough training data for second level types in PDTB.

6 Discussion

6.1 Comparison of Model Complexity

In Table 7 we present the number of parameters of our model in the first two columns. We have con-

Model	Parameters
Ours	
Conv 2,3,4,5 50 per size	242.2k
Conv 2,4,6,8 100 per size	692k
Gate_1	240k
Gate_2	660k
Total	1834.2k
LSTM Model	
Bi-LSTM Layer 1	1550.4k
Bi-LSTM Layer 2	2160k
Total	3710.4k

Table 7: Comparison of model complexity. $Gate_1$ & $Gate_2$ have output of size 300. LSTMs have hidden state of size 300.

volution layers to learn n-gram features and *WP-k* features. Apart from these layers, we have two gate layers: $Gate_1$ and $Gate_2$ in the table. Our model has approximately 1.8 million parameters. The input embeddings to our model have dimensionality of 346 (300 (word) + 46 (POS)). Assuming the same input to the two-layered Bi-LSTM model with a hidden state of size 300, this model will have approximately 3.7 million parameters. For this comparison, we have assumed the number of parameters of the LSTM given input vectors of size m and giving output vectors of size n is $4(nm + n^2)$. Both models have dense layers for implicit and explicit relation prediction so they are ignored for these calculations.

6.2 Comparison of Training Time

We also compare the running time of our model to the model of (Dai and Huang, 2018). We compare the wall clock training time per epoch of both systems, using their released code as well as our own. For a fair comparison, we re-implemented our architecture in Pytorch to match their usage. Furthermore, the models were run on the same GPU (Tesla K80) on the same machine. We ran each model three times for five epochs. The training time of our model was 109.6 seconds on average compared to 206.17 seconds for their model.

6.3 Qualitative Analysis

We conduct a qualitative analysis in an attempt to understand the most important *WP-k* and n-grams learned by our architecture. We modified our architecture to get rid of all non-linear layers after the convolutional layers, which allows us to examine the effect of the word pairs and n-grams

Implicit Relations and Top Features
Arg1: Alliant said it plans to use the microprocessor in future products **Arg2**: It declined to discuss its plans for upgrading its current product line **Class**: Comparison ***WP-k***: **(said : declined)**, (Alliant : product line), (declined : Alliant said), (upgrading : microprocessor future products), **(plans : declined discuss its plans)**, (discuss : use the microprocessor future) ***Arg1 n-grams***: (Alliant said), (microprocessor in future products), (plans to use the microprocessor) ***Arg2 n-grams***: (product line), (It declined to discuss), (for upgrading its current product)
Arg1: I ca n't see why there would be a conflict of interest **Arg2**: Estimates are based on the previous price of similar works sold at auction and current market conditions, and are not affected by any knowledge of who the potential buyer could be **Class**: Contingency ***WP-k***: (n't : affected), (not : conflict), (why : not affected), (not : n't see), (see : works sold auction and), **(affected : why there would conflict)** ***Arg1 n-grams***: (a conflict of interest), (ca n't see why there) ***Arg2 n-grams***: (Estimates are based on), (works sold at auction), (are not affected by any)
Arg1: And it allows Mr. Van de Kamp to get around campaign spending limits **Arg2**: He can spend the legal maximum for his campaign **Class**: Expansion ***WP-k***: **(And : can)**, (limits : spend), (allows : spend), (And : He can), **(maximum : spending limits)** ***Arg1 n-grams***: (And it allows), (spending limits) ***Arg2 n-grams***: (He can spend), (legal maximum), (his campaign)
Arg1: As the market dropped Friday , Robertson Stephens slashed the value of the offering by 7% **Arg2**: Yesterday , when similar securities rebounded , it bumped the valuation up again **Class**: Temporal ***WP-k***: (As : when), **(bumped : slashed)**, (As : Yesterday when), (Yesterday : As the market), (when : As the market dropped) ***Arg1 n-grams***: (market dropped), (Robertson Stephens slashed the value) ***Arg2 n-grams***: (similar securities), (Yesterday , when), (bumped the valuation up again)
Arg1: the fact that seven patents were infringed suggests that infringement was willful **Arg2**: It 's difficult to be that consistently wrong **Class**: Contingency ***WP-k***: (willful : consistently), **(willful : wrong)**, (suggests : difficult that consistently wrong), (consistently : infringed suggests that infringement) ***Arg1 n-grams***: (the fact), (suggests that infringement was willful) ***Arg2 n-grams***: (consistently wrong), ('s difficult to be that)
Arg1: and special consultants are springing up to exploit the new tool **Arg2**: Blair Entertainment has just formed a subsidiary – 900 Blair – to apply the technology to television **Class**: Expansion ***WP-k***: **(springing : formed)**, (exploit : formed), (springing : subsidiary Blair), (formed : springing exploit), (springing : has just formed subsidiary), **(Blair : and special consultants)** ***Arg1 n-grams***: (special consultants are springing up) ***Arg2 n-grams***: (Blair Entertainment has), (subsidiary – 900 Blair –)

Table 8: Implicit examples along with top features selected from across three runs. Note that we drop very short words in the cartesian product only and not in the individual arguments.

directly on the output. Dropping the gate layers caused the F1 score averaged across the first three runs to drop from 50.9 to 50.1, indicating both that the gate layers help incorporate interactions between the model components and that our approach here is a reasonable approximation to what the model is learning. Instead of the gates, we concatenate the output of all the convolutional layers and use a final classification layer (different for implicit and explicit relations as in our full model) to train this simplified architecture. In the absence of non-linearity, we are able to map the features selected by max pooling back to the *WP-k* and n-grams associated with their embeddings (i.e. argmax pooling rather than max pooling and mapping the selected indices back to the input). As each filter is associated with multiple feature maps ($k = 100$ for word pairs and $k = 50$ for n-grams as described in Section 4.2), we count the number of times each *WP-k* and n-gram was selected during pooling and select the most prominent features according to their frequency.

We present six implicit examples,[3] in Table 8 and the corresponding top *WP-k* and n-gram features. We selected these examples by running the simplified architecture three times and selecting implicit examples which were classified correctly during all the runs in the *Joint Learning* setting.

We find the following general properties in the examples we studied:

- We consistently observed that smaller filters learn either verb-to-verb mappings or adjective-to-adjective mappings. In examples one, four and five, (said : declined), (bumped : slashed) and (willful : wrong) are selected respectively. The first two pairs capture antonymy and last one maps adjectives.

- Larger filters tend to align important words (verbs and nouns) in either argument to phrases in the other argument. In the third example, (maximum: spending limits) is selected, along with (affected : why there would conflict) in the second example, and (plans : declined discuss its plans) from the first, among others.

- For the third example, while the true class is *Expansion*, which the *Joint Learning* model classifies correctly, the *Implicit Only* model labels it as *Contingency*. The *Joint Learning* model selects functional word pair interactions such as (And : can), which may be more indicative of an Expansion relation due to the presence of the connective "And" at the start of the first argument. We also observe the word pair (Kamp : spend) is not selected as a top feature in the *Joint Learning* setting, while it is selected in the *Implicit Only* scenario. As it includes a proper noun, it is unlikely to generalize as a useful feature. Finally, *Joint Learning* identifies the semantically coherent *WP-2* (maximum : spending limits). This pair does not appear in the *Implicit Only* case.

[3] Although we learn implicit and explicit relations jointly, we focus on only implicit relations due to space constraints.

7 Conclusion

We proposed an approach to learn implicit relations by incorporating word pair features as a novel way to capture the interaction between the arguments, a distinct approach compared to the popular attention-based approaches used with Bi-LSTM based models. We also show that joint learning of implicit and explicit relations is beneficial to implicit relations. Our results show that our model is able to surpass or match the performance of a Bi-LSTM based model using paragraph level context.

For future work, we plan to explore data augmentation techniques. As our best performance is on the expansion class, which is also the largest, if we are able to obtain more data we might improve our performance on smaller classes as well. We will thus investigate extending our joint learning method to include resources beyond the PDTB.

Another possible avenue is to replace word embeddings with contextualized embeddings to study the efficacy of the latter with our architecture. Pre-trained language models like BERT (Jacob et al., 2018) have been recently used to achieve state-of-the-art results on sentence pair classification tasks. As a future step we will experiment with our model on top of these contextual representations, which would likely enhance the performance while still maintaining the interpretability.

Acknowledgments

We would like to thank Kathleen McKeown, Dimitris Alikaniotis and anonymous reviewers for their constructive feedback

References

Fatemeh Torabi Asr and Vera Demberg. 2015. Uniform information density at the level of discourse relations: Negation markers and discourse connective omission. In *IWCS 2015*, page 118.

Hongxiao Bai and Hai Zhao. 2018. Deep enhanced representation for implicit discourse relation recognition. *CoRR*, abs/1807.05154.

Or Biran and Kathleen McKeown. 2013. Aggregated word pair features for implicit discourse relation disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 69–73, Sofia, Bulgaria. Association for Computational Linguistics.

Sasha Blair-Goldensohn, Kathleen McKeown, and Owen Rambow. 2007. Building and refining rhetorical-semantic relation models. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 428–435, Rochester, New York. Association for Computational Linguistics.

Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Implicit discourse relation detection via a deep architecture with gated relevance network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1726–1735, Berlin, Germany. Association for Computational Linguistics.

Zeyu Dai and Ruihong Huang. 2018. Improving implicit discourse relation classification by modeling inter-dependencies of discourse units in a paragraph. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 141–151. Association for Computational Linguistics.

Devlin Jacob, Chang Ming-Wei, Lee Kenton, and Toutanova Kristina. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Arxiv*. Google Research.

Yangfeng Ji, Gongbo Zhang, and Jacob Eisenstein. 2015. Closing the gap: Domain adaptation from explicit to implicit discourse relations. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2219–2224, Lisbon, Portugal. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Man Lan, Jianxiang Wang, Yuanbin Wu, Zheng-Yu Niu, and Haifeng Wang. 2017. Multi-task attention-based neural networks for implicit discourse relationship representation and identification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1299–1308. Association for Computational Linguistics.

Wenqiang Lei, Xuancong Wang, Meichun Liu, Ilija Ilievski, Xiangnan He, and Min-Yen Kan. 2017. Swim: A simple word interaction model for implicit discourse relation recognition. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4026–4032.

Yang Liu and Sujian Li. 2016. Recognizing implicit discourse relations via repeated reading: Neural networks with multi-level attention. *CoRR*, abs/1609.06380.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2750–2756. AAAI Press.

Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 368–375, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 683–691, Suntec, Singapore. Association for Computational Linguistics.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind Joshi. 2008. Easily identifiable discourse relations. In *Coling 2008: Companion volume: Posters*, pages 87–90, Manchester, UK. Coling 2008 Organizing Committee.

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

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016. A stacking gated neural architecture for implicit discourse relation classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2263–2270. Association for Computational Linguistics.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1006–1017, Vancouver, Canada. Association for Computational Linguistics.

Hannah Rohde and William S. Horton. 2010. Why or what next? eye movements reveal expectations about discourse direction.

Samuel Rönnqvist, Niko Schenk, and Christian Chiarcos. 2017. A recurrent neural model with attention for the recognition of Chinese implicit discourse relations. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 256–262, Vancouver, Canada. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

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, Berlin, Germany. Association for Computational Linguistics.