Neural Network Approaches to Implicit Discourse Relation Recognition

Andre Cianflone

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science at

Concordia University

Montréal, Québec, Canada

December 2017

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Andre Cianflone**

Entitled: **Neural Network Approaches to Implicit**

Discourse Relation Recognition

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Charalambos Charis Poullis

_____ Examiner
Dr. Adam Krzyzak

_____ Examiner
Dr. Tien D. Bui

_____ Supervisor
Dr. Leila Kosseim

Approved by _____
Dr. Sudhir Mudur, Chair
Department of Computer Science and Software Engineering

_____ 2017

_____
Dr. Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

Neural Network Approaches to Implicit
Discourse Relation Recognition

Andre Cianflone

In order to understand a coherent text, humans infer semantic or logical relations between textual units. For example, in "I am hungry. I did not have lunch today." the reader infers a "causality" relation even if it is not explicitly stated via a term such as "because". The linguistic device used to link textual units without the use of such explicit terms is called an "implicit discourse relation". Recognising implicit relations automatically is a much more challenging task than in the explicit case. Previous methods to address this problem relied heavily on conventional machine learning techniques such as CRFs and SVMs which require many hand-engineered features.

In this thesis, we investigate the use of various convolutional neural networks and sequence-to-sequence models to address the automatic recognition of implicit discourse relations. We demonstrate how our sequence-to-sequence model can achieve state-of-the-art performance with the use of an attention mechanism. In addition, we investigate the automatic representation learning of discourse relations in high capacity neural networks and show that for certain discourse relations such a network does learn discourse relations in only a few neurons.

Acknowledgments

I would first like to thank Dr. Leila Kosseim. In 2014 I decided to return to university to pursue studies in computer science. As part of my Graduate Diploma, I attended Dr. Kosseim's Artificial Intelligence course. It was a life-changing event. Her course, with the help of her amazing teaching abilities, sparked my passion for AI. I was truly lucky as the following year Dr. Kosseim accepted to supervise me for my Master of Computer Science. Her guidance, her knowledge, her attention to detail and her magnificent personality have made working with her over the past few years a great privilege, and a lot of fun.

I would like to thank my lab colleagues Majid Laali and Elnaz Davoodi with whom I had many wonderful discussions over the years on varied academic topics and many non-thesis related but equally fascinating fields. I would also like to thank my colleague Sohail Hooda who first introduced me to deep learning early on in my studies.

Lastly, I dedicate this thesis to three very important people: my partner Au Ca and my parents. Since I first made the decision to leave a lucrative career in finance, Au Ca showed nothing but love, support and encouragement. She motivated me to follow my dreams and continues to inspire me to be a better person, every day. My parents were also instrumental for their support and unconditional love not only over the last few years but, well, since birth. They have been a huge source of inspiration, wisdom, and great examples to follow. Words cannot express my gratitude to all three of them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Understanding a discourse involves recognizing how its textual units are linked together to form a coherent text. Recognizing such links automatically is known as *discourse relation recognition* (DRR). Take for example the following sentence:

(Ex. 1) *Get your facts first,* and then **you can distort 'em as much as you please**.[1]

The underlined expression connects the first discourse argument (in italic), referred to as `Arg1`, with the second discourse argument (in bold), referred to as `Arg2`, via a TEMPORAL discourse relation. Understanding discourse relations has been shown to be beneficial to many tasks, including sentiment analysis (Somasundaran et al., 2009), summarization (Louis et al., 2010), coherence evaluation (Lin et al., 2011) and question answering (Jansen et al., 2014).

The underlined expression in (Ex. 1) is known as a *discourse connective*, and constitutes a strong signal in the identification of discourse relations. When such a connective is present, the relation is referrred to as `Explicit`. Today, the automatic recognition of `Explicit` discourse relation is performed with near-human performance. In fact, (Pitler et al., 2008) showed that by solely considering the `Explicit`'s discourse connective and ignoring the words in `Arg1` and `Arg2`, one can correctly predict the discourse relation sense with 93% accuracy. On the other hand, when the

[1] Quote by Mark Twain while interviewed by Rudyard Kipling. Page 180 of (Rudyard, 1899).

discourse connective is missing, the discourse relation is referred to as `Implicit`. For example:

(Ex. 2) *After two years of talks, plans for the venture are sufficiently advanced for the companies to seek French and British government clearance.* **The companies hope for a final agreement by year-end** (2362)[2]

The connective *accordingly* is understood by the context and could have been inserted in (Ex. 2). However, since the discourse connective is missing, the reader infers a causal relation in the discourse solely by the context.

In the case of `Implicit` discourse relation recognition (IDR), it is extremely challenging to automatically predict these discourse relation senses due to the lack of a discourse connective. The relation between the two arguments must be deduced by extracting the semantics. (Pitler et al., 2009) presents the first work that addresses this task exclusively. Given the highly unbalanced distribution of senses (see Chapter 2), the authors formulate the task as one-vs-all classification, previously shown to be an appropriate approach to multiclass classification (Rifkin and Klautau, 2004). The best combination of features achieves an F1-score on IDR of 21.96% for COMPARISON, 47.13% for CONTINGENCY, 76.42% for EXPANSION and 16.76% for TEMPORAL, all very far from the average F1-score of 93% in the `Explicit` case.

1.2 Goal of this Thesis

Past methods used for IDR have relied heavily on feature engineering (Pitler et al., 2009; Xue et al., 2015) (see Chapter 2 for details). Motivated by recent research on the use of neural networks for natural language processing, the goal of this thesis is to develop neural network architectures for IDR. Instead of hand-crafting features, we want to develop architectures that not only automatically learn such features, but are also better sense predictors. Furthermore, we hypothesize that a neural network which takes into consideration discourse structure theory in its architecture can perform better at IDR than a generic neural network.

[2] All examples in this thesis ending with a number in parenthesis come from the Penn Discourse Treebank (Prasad et al., 2008), discussed in detail in Chapter 2. The number refers to the document ID.

1.3 Contributions

This thesis presents a number of theoretical and practical contributions:

- the implementation of various convolutional neural networks for IDR, including our work at CoNLL-2016 (Laali et al., 2016) (see Chapter 3)
- insights relating discourse theory and convolutional neural network architectures for IDR, and the settings under which they work best (see Chapter 3)
- a novel architecture, based on sequence-to-sequence models, for IDR which achieves state-of-the-art performance, to appear in (Cianflone and Kosseim, 2018) (see Chapter 4)
- insights into the possible automatic learning of discourse relation representation in unsupervised learning (see Chapter 5)

1.4 Thesis Structure

This chapter briefly motivated the importance of discourse relations and the difficulty of IDR with previous conventional machine learning approaches. Recent research inspired us to investigate neural networks for IDR in this research. The rest of this thesis is structured as follows. Chapter 2 reviews the major discourse theories in use, the datasets used in our experiments, previous work on IDR, and background theory on convolutional neural networks and recurrent neural networks on which later chapters are based. Chapter 3 presents two variations of shallow CNNs, a system with hierarchical CNNs as well as a CNN augmented with a recurrent layer. Chapter 4 presents our work on sequence-to-sequence models for IDR, including a model achieving state-of-the-art performance with the help of an attention mechanism. Chapter 5 examines unsupervised learning of discourse relation sense. Finally, Chapter 6 summarizes the work in the thesis and proposes some future experiments based on these findings.

Chapter 2

Related Work

Since this thesis focuses on implicit discourse relation recognition (IDR), we first briefly present the main frameworks used in the study of computational discourse analysis. Discourse structures represent the way humans organize or structure text to communicate. There are two popular ways of viewing discourse structure in NLP: as a tree structure or as a linear structure. The tree structure is the view taken by Rhetorical Structure Theory (RST) (Mann and Thompson, 1988), whereas the linear structure is the view taken by the Penn Discourse Treebank (PDTB) (Prasad et al., 2008). In Section 2.1 we give a brief overview of RST and give a more detailed description of the PDTB in Sections 2.2 and 2.3 as we use this dataset throughout this thesis. We then describe previous conventional machine learning and preliminary neural network approaches to IDR in Section 2.6. In Section 2.7 we give the necessary background on convolutional neural networks and recurrent neural networks to better appreciate our work in Chapters 2.7 and 2.8.

2.1 Rhetorical Structure Theory

The Rhetorical Structure Theory framework (Mann and Thompson, 1988) describes text in terms of a tree structure, where each leaf is a textual unit, known as an *Elementary Discourse Unit* (EDU). An EDU is the minimal unit of discourse. EDUs are linked to one another to form nodes corresponding to contiguous text spans. An extensive sample tree is illustrated in Figure 2.1. The tree describes how each node, or EDU, is related to another via a labelled arc, where the label

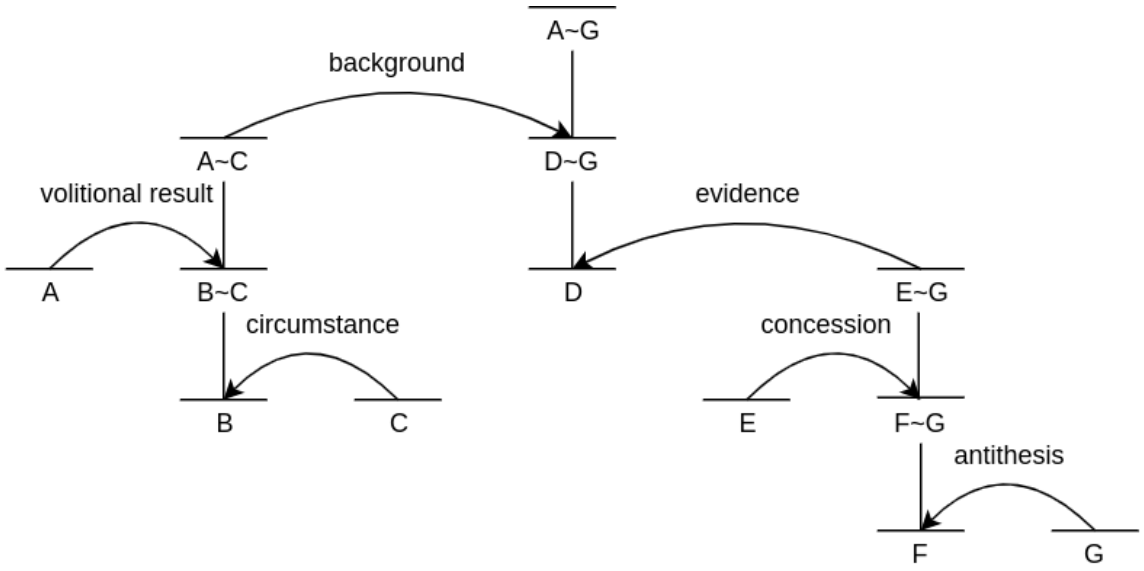


Figure 2.1: A sample RST tree from (Mann and Thompson, 1988). The tree corresponds to the following text: [Farmington police had to help control traffic recently]^{A} [when hundreds of people lined up to be among the first applying for jobs at the yet-to-open Marriott Hotel.]^{B} [The hotel's help-wanted announcement - for 300 openings - was a rare opportunity for many unemployed.]^{C} [The people waiting in line carried a message, a refutation, of claims that the jobless could be employed if only they showed enough moxie.]^{D} [Every rule has exceptions,]^{E} [but the tragic and too-common tableaux of hundreds or even thousands of people snake-lining up for any task with a paycheck illustrates a lack of jobs,]^{F} [not laziness.]^{G}

on the arc corresponds to a discourse relation. Nodes can only be linked to adjacent text spans.

All nodes and EDUs are linked in a hierchical fashion. Take for example, clauses B and C from Figure 2.1. The clause C is the context which explains the circumstances of clause B, hence the discourse relation is labelled *circumstance*. The clause receiving the arrow is known as the *nucleus*, since it is considered to carry the most important meaning in the discourse relation. The clause from which the arrow leaves is known as the *satellite*, the supporting clause. RST tree structures have been used in many NLP applications such as text summarization (Marcu, 1999), sentiment analysis (Bhatia et al., 2015) and text classification (Ji and Smith, 2017). RST parsing can be done automatically with discourse parsers such as the DPLP (Ji and Eisenstein, 2014).

The common dataset used for training RST parsers is the RST Discourse Treebank (RST-DT) (Carlson et al., 2001). The RST-DT consists of 385 documents from the Penn Treebank (Marcus et al., 1993), representing over 176,000 words. RST defines 78 relations grouped into 16 classes. Given the complexity of implicit discourse relation recognition, the large number of RST relations

and the limited dataset, our work is based on the much larger and simpler Penn Discourse Treebank (Prasad et al., 2008), also based on the Penn Treebank.

2.2 Penn Discourse Treebank

The Penn Discourse Treebank (PDTB) (Prasad et al., 2008) dataset consists of 40,600 annotated discourse relations and their arguments over the 1 million word Wall Street Journal (WSJ) corpus on which the Penn Treebank (Marcus et al., 1993) is based[1]. Compared to the RST-DT, the PDTB is based on the Discourse Lexicalized Tree-Adjoining-Grammar framework (Forbes et al., 2003; Webber, 2004; Forbes-Riley et al., 2005). The PDTB structure takes a more "shallow" view of discourse structures where relations are defined solely between adjacent sentences or close text spans. There is no notion of nucleus, or satellite, and no hierarchical discourse structure. The two textual units related by a relation are known as arguments (`Arg1` and `Arg2`). The PDTB annotates the beginning and end of `Arg1` and `Arg2` composing a discourse relation, a possible discourse connective and the labelled discourse relation (called *sense* in the PDTB).

Samples in the dataset are first differentiated according to the syntactic expression of the discourse connective, called *discourse annotation*, such as `Explicit` or `Implicit` introduced in Chapter 1.

The relation label, such as TEMPORAL, is referred to as *discourse relation sense* in the PDTB. Discourse relation senses are not unique to an annotation, they can be the same for `Explicit` and `Implicit` relations. While discourse relations are flat and not linked hierarchically, the relations themselves are organized into a hierarchy. The details of the annotations and the inventory of the senses are discussed in Sections 2.2.1 and 2.2.2.

The PDTB source files, following the PTB standard, is split into 24 sections where each section has on average 1600 samples. Following the PTB tradition, the PDTB Manual (Prasad et al., 2008) suggests that sections 2 to 21 be used for training, section 22 for validation and section 23 for testing. Sections 0, 1 and 24 can additionally be used for validation. We refer to this section split across training, validation and test datasets as the **PDTB-split**.

[1]Technically a few files from the PTB are not included in PDTB. See (Prasad et al., 2008) for details.

An alternative split is proposed by (Pitler et al., 2009) when classifying only top-level senses for Implicit relations. Later work on IDR by various authors (Zhang et al., 2015a; Zhou et al., 2010; Park and Cardie, 2012; Rutherford and Xue, 2014) also used the same split to make their work comparable with (Pitler et al., 2009). In this scheme, sections 2-20 are used for training, sections 21-22 for testing and sections 0-1 for validation. We refer to this breakdown as the **TOP-split**.

2.2.1 Discourse Annotations

In Chapter 1 we briefly introduced the concepts of Implicit and Explicit relations. There are 2 additional types of discourse relations. Like Implicit relations, AltLex, or alternative lexicalization, are also not signalled by a connective. Unlike Implicit relation, it is signaled by a textual expression, which stands as an alternative to a discourse connective. The last type EntRel, for entity relation, connects two discourse arguments via an entity-based coherence relation. We now review all forms of annotation with examples.

When discourse connectives are expressed in a text, these are labelled Explicit discourse connective. For example:

(Ex. 3) *The city's Campaign Finance Board has refused to pay Mr. Dinkins $95,142 in matching funds* because **his campaign records are incomplete.** (0041)

In (Ex. 3) the discourse connective "because" is explicitly stated and acts to connect Arg1 and Arg2. Arguments in Explicit relations are not necessarily adjacent to each other. In the case of Implicit relations, the discourse connective is omitted as in:

(Ex. 4) *The city's Campaign Finance Board has refused to pay Mr. Dinkins $95,142 in matching funds.* **His campaign records are incomplete.** (0041)

While the text does not include a discourse connective, which is a strong signal of the discourse relation, the relation is still understood by the context, and a connective can be inferred. For Implicit samples, the PDTB dataset additionally includes a suggested discourse connective that could be inserted between the two arguments. With Implicit relations, Arg2 is always the sentence immediately following Arg1, also a sentence.

	Training	Validation	Test
Explicit	14722	680	923
Implicit	13156	522	769
AltLex	524	19	30
EntRel	4133	215	217
NoRel	204	8	4
Total	**32739**	**1444**	**1943**

Table 2.1: PDTB dataset breakdown with respect to annotation types. WSJ sections 2 to 21 are used for the training set, section 22 for validation and section 23 for testing. Sections 0,1 and 24 are not included in this table. In the CoNLL-2015 and CoNLL-2016 tasks, the above annotations are split between `Explicit` and `Non-Explicit`, where the latter includes `Implicit`, `AltLex` and `EntRel` relations. The dataset breakdown for `Non-Explicit` is: 17,813 for training, 756 for validation and 1016 for testing.

There are cases however where no such `Implicit` connective can be inferred, but a relation still exists. This is the case for the so-called `AltLex` relations where the relation is *alternatively lexicalized* by another expression and inserting a discourse connective would make the expression of the relation redundant. For example:

(Ex. 5) *During the latter part of the 19th century, Russia was on a gold standard and had gold reserves representing more than 100% of its outstanding currency, but no one outside Russia used rubles. The Bank of England, on the other hand, had gold reserves that averaged about 30% of its outstanding currency, and Bank of England notes were accepted throughout the world.* [**The most likely reason for this disparity**] **is that the Bank of England was a private bank with substantial earning assets, and the common-law rights of creditors to collect claims against the bank were well established in Britain.** (0985)

The `AltLex` expression between the brackets acts as a "non-connective expression", an expression inferring a relation and acting like a discourse connective. Adding a discourse connective would break the text coherence, unlike in the `Implicit` case. The PDTB dataset contains very few samples of `AltLex`, as observed in Table 2.1. As such we have not considered `AltLex` in our work.

A second case is the `EntRel` case where no discourse relation is inferred but an entity-based coherence occurs. For example, in (Ex. 6) the same entity (Mr. Milgrim) is realized in `Arg1` and `Arg2`, as in:

(Ex. 6) *Hale Milgrim, 41 years old, senior vice president, marketing at Elecktra Entertainment Inc., was named president of Capitol Records Inc., a unit of this entertainment concern.* **Mr. Milgrim succeeds David Berman, who resigned last month.** (0945)

`EntRel` are closely related to the EXPANSION class and `Implicit` connectives, and thus are often included in `Implicit` relations (see Section 2.2.2).

In the event of absence of discourse relation or entity-based coherence, the sentence pairs are simply labelled `NoRel`. There are also very few `NoRel` in the PDTB, as seen in Table 2.1. For this reason, these cases have not been considered in our work.

2.2.2 Discourse Sense

The discourse annotations `Explicit`, `Implicit` and `AltLex` share a discourse sense tagset. `EntRel`, on the other hand, is simply labelled with the sense `EntRel`. In the PDTB, discourse senses are hierarchically structured, with four top-level *classes* of discourse relations: TEMPORAL, CONTINGENCY, COMPARISON and EXPANSION; as well as level 2 *types* and level 3 *subtypes*. The full sense hierarchy is shown in Figure 2.2, the top-level distribution can be found in Table 2.2, and the fine-grained distribution in Table 2.3. This thesis is only concerned with sense prediction of `Implicit` and `EntRel` relations, and so Tables 2.2 and 2.3 do not include `Explicit` relations data.

The TEMPORAL sense class indicates that the two arguments are linked temporally. For example:

(Ex. 7) But a Soviet bank here would be crippled unless Moscow found a way to settle the $188 million debt, *which was lent to the country's short-lived democratic Kerensky government* before **the Communists seized power in 1917**. (0035)

(Ex. 7) is labelled with sense TEMPORAL:Asynchronous:precedence, where Asynchronous indicates the arguments are temporally ordered, and precedence indicates `Arg1` precedes `Arg2`. As observed in Table 2.2, TEMPORAL is the smallest of the classes. Due to its small size, classifying TEMPORAL can be problematic, as discussed in Chapter 4.

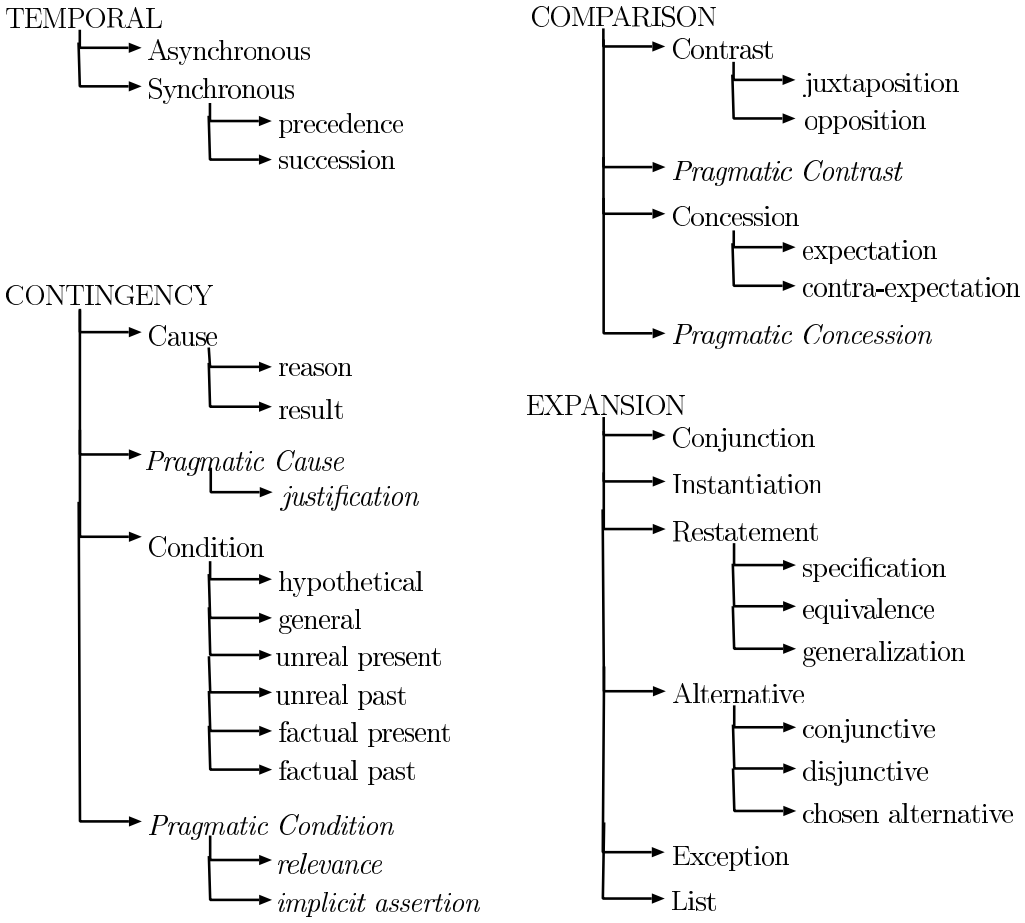


Figure 2.2: Hierarchy of discourse relations in the PDTB. Credit: (Prasad et al., 2008)

Top Level	Nb Implicit Instances
TEMPORAL	950
CONTINGENCY	4185
COMPARISON	2832
EXPANSION	8861
Total	**16828**

Table 2.2: Top-level breakdown of the PDTB with `EntRel` merged into EXPANSION. Note here we are using the `Implicit` term to mean both `Implicit` and `Entrel`, excluding `AltLex`, following the standard set by (Pitler et al., 2009).

When `Arg1` and `Arg2` causally influence each other, the discourse relation is labelled with the sense CONTINGENCY. For example:

(Ex. 8) *USAir has great promise.* **By the second half of 1990, USAir stock could hit 60.** (1881)

is labeled as CONTINGENCY:Cause:reason. The type Cause and reason indicate that `Arg2` is the cause and `Arg1` is the effect.

The EXPANSION class indicates that `Arg2` expands the discourse introduced by `Arg1`. For example:

(Ex. 9) *He says he spent $300 million on his art business this year.* **A week ago, his gallery racked up a $23 million tab at a Sotheby's auction in New York buying seven works, including a Picasso.** (0800)

is labelled EXPANSION:Instantiation where Instantiation indicates `Arg2` gives more detail about `Arg1`. Given that most frameworks consider `EntRel` to be a subset of EXPANSION, we follow the convention set by (Pitler et al., 2009): when considering only the top-level classification we merge `EntRel` into `Implicit` relation's EXPANSION class sense. The combined class results in a relatively much larger class, as observed in Table 2.2.

The final top-level sense class COMPARISON labels those relations which indicate differences in the situation between `Arg1` and `Arg2`. For example:

(Ex. 10) *Operating revenue rose 69% to A$8.48 billion from A$5.01 billion.* But **the net interest bill jumped 85% to A$686.7 million from A$371.1 million.** (1449)

is labelled COMPARISON:Contrast:juxtaposition, where Contrast indicates a contrast in the values of a shared property across `Arg1` and `Arg2`, and juxtaposition indicates those values can be alternatives of each other. As shown in Table 2.2, COMPARISON is also a small class which can be problematic to classify, discussed in Chapter 4.

For an extensive review of all classes, types, and subtypes, please refer to the PDTB 2.0 Manual (Prasad et al., 2008).

Important note on terminology: In Chapters 3 and 4, we include `EntRel` relations as part of an `Implicit` EXPANSION relation only when classifying the top-level relations (see Section 2.2.2). In this case, we refer to the combined set simply as `Implicit` relations, despite including `EntRel`. This has been done to follow the common practice in the literature (Pitler and Nenkova, 2009; Zhou et al., 2010; Rutherford and Xue, 2014). However, in classifying the lower level senses (see Section 2.2.2), we keep `EntRel` as a separate annotation, which has no labelled sense. Additionally, when referring to relations which are not `Explicit`, including `Implicit`, `EntRel` and `AltLex`, we use the term `Non-Explicit` relations, the preferred term in (Xue et al., 2015, 2016).

2.3 The CoNLL Dataset and Shared Task

In 2015 and 2016, the Conference on Natural Language Learning (CoNLL) held a shared task on Shallow Discourse Parsing (SDP), which we will respectively refer to as CoNLL-2015 (Xue et al., 2015) and CoNLL-2016 (Xue et al., 2016). The two shared tasks introduced a slightly modified version of the PDTB, which we denote C-PDTB. The C-PDTB dataset consists of the full PDTB dataset with a minor reduction in the number of subtypes, detailed in (Xue et al., 2015). CoNLL-2015 and CoNLL-2016 focused on fine-grained classification, meaning classifying the lowest level discourse relations in the discourse hierarchy of Figure 2.2. All work in this thesis on fine-grained classification uses the C-PDTB dataset.

In addition to including all of the PDTB, the C-PDTB includes a *blind test set*, a second test set created specifically for CoNLL-2015 and reused for CoNLL-2016, but only released to participants following the CoNLL-2016 conference. The blind test set consists of newswire text selected from

English Wikinews[2], accessed October 22nd, 2014. The annotation is consistent with WSJ-style text and manually annotated with discourse relations and connectives (Xue et al., 2015). The text selected for the blind dataset was chosen to resemble the WSJ in terms of grammar, style, and length. When labelling discourse relation senses on the Wikinews dataset, annotator agreement reached 91.0% for `Explicit` relations, and 80.9% for `Non-Explicit` relations. This highlights the difficulty of labelling `Implicit` and other `Non-Explicit` relations, even for humans. The blind test was used to officially rank systems at CoNLL-2015 and CoNLL-2016. Therefore we also base our performance on the blind test set. Since this thesis focuses on fine-grained classification for `Non-Explicit` relations, we list the C-PDTB breakdown in Table 2.3 for `Implicit` relations and `EntRel`, ignoring `AltLex` due to its relatively small size. However, note that the C-PDTB dataset includes `AltLex`.

2.4 Metrics

In this thesis, when performing discourse relation sense classification, we will measure the performance of our classification with *accuracy* or *F1-score*. Accuracy is simply a measure of the number of correctly classified instances, irrespective of their relevancy, giving equal importance to positive (relevant) samples and negative (non-relevant) samples. Accuracy is not a proper metric in case of one-vs-all classification for IDR.

Consider the sample distribution for the sense EXPANSION and TEMPORAL in Table 2.4. While the EXPANSION class is somewhat balanced at 56.7% positive and 43.3% negative, the TEMPORAL is extremely unbalanced at 5.4% positive and 94.6% negative. A classifier always predicting the largest class would achieve accuracy of 56.7% for EXPANSION and an impressive 94.6% for TEMPORAL. While the classifier seems to perform quite well in case of TEMPORAL, it actually does not since we value the correct identification of positive samples. We quantify our classifier with two desired properties: *precision* and *recall*.

Precision P measures the relevance of selected items, specifically the ratio of true positives T_p to the sum of T_p and false positives F_p:

[2] https://en.wikinews.org

Implicit Sense	Train	Validation	Test
EXPANSION:Conjunction	3227	120	141
EXPANSION:Restatement	2486	101	190
CONTINGENCY:Cause:reason	2059	73	113
COMPARISON:Contrast	1614	82	127
CONTINGENCY:Cause:result	1372	49	89
EXPANSION:Instantiation	1132	47	69
TEMPORAL:Asynchronous:precedence	418	25	7
COMPARISON:Concession	193	5	5
EMPORAL:Synchrony	153	8	5
COMPARISON	145	1	0
EXPANSION:Alternative:chosen-alternative	142	2	15
TEMPORAL:Asynchronous:succession	125	3	5
EXPANSION	73	6	3
EXPANSION:Alternative	11	0	0
CONTINGENCY:Condition	2	0	0
TEMPORAL	1	0	0
EXPANSION:Exception	1	0	0
CONTINGENCY:Cause	1	0	0
CONTINGENCY	1	0	0
Total	**13156**	**522**	**769**
EntRel	4133	215	217

Table 2.3: C-PDTB fine-grained sense distribution for `Implicit` relations with the addition of `EntRel`. Although the PDTB framework considers senses as hierarchical with up to 3 levels, some discourse samples do not have clear third level subtypes or even second level types, which explains why some of the above senses have fewer than three levels. C-PDTB considers `Non-Explicit` to also include `AltLex`. However, since `AltLex` samples are quite few, we ignore these. For an extensive sense breakdown, see (Xue et al., 2015).

Relation	Test (positive/negative)
EXPANSION	671 / 512
TEMPORAL	64 / 1119

Table 2.4: The distribution of a sample dataset for two one-vs-all binary classifiers for discourse relation sense classes. Notice the relative balance of class EXPANSION and large imbalance of class TEMPORAL.

$$P = \frac{T_p}{T_p + F_p} \tag{1}$$

Whereas recall R scores the quantity of selected items compared to all relevant items, specifically it calculates the ratio of T_p to the sum of T_p and false negatives F_n:

$$R = \frac{T_p}{T_p + F_n} \tag{2}$$

In the case of TEMPORAL class in Table 2.4 a max-prior system would always predict the label *other*, as in not-TEMPORAL, resulting in both an precision and recall score of 0, despite an accuracy of 94%. If the classifier predicts a single discourse relation as TEMPORAL, which turns out to be correct, and the rest as *other*, it would have precision of $P = \frac{1}{1+0} = 100\%$, a perfect precision score. Recall on the other hand would be $R = \frac{1}{1+63} = 1.56\%$, a terrible score. Hence, recall helps to counteract precision in the event the system returns as few positive labels as possible. Combining the two results in the F1-score:

$$\text{F1} = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{3}$$

which is the harmonic mean of precision and recall.

An additional metric to consider is model perplexity. We do not calculate perplexity for classification, but consider it when selecting parameters from a language model (see Chapter 5). A language model computes the probability of a sequence of words $x = [x_1, x_2, \ldots, x_N]$, where each word x_i depends on all previous words $x_1, \ldots, x_{i-1}$. We evaluate the language model's performance by calculating per word perplexity PP as:

$$PP = \exp\left(-\frac{1}{N}\sum_{i=1}^{N} log_e q(x_i)\right) \tag{4}$$

where q is the model's approximate distribution of the true language probability distribution p. Since words w_i are taken from actual test, a good model should give high probability to w_i. The lower the perplexity score the better. The Penn Treebank (Marcus et al., 1993) is a common benchmark on which language models are tested. In only the past few years, published perplexity

scores on the PTB have improved quite significantly from 92.0 (Mikolov and Zweig, 2012), to 72.1 (Grave et al., 2017) and an astoundingly low recent state-of-the-art of 52.8 (Merity et al., 2018). The ability of recurrent neural networks to model language with such performance motivates our use of RNNs in Chapter 4.

2.5 Implicit Discourse Relation Tasks

The full task at CoNLL-2015 (Xue et al., 2015) and CoNLL-2016 (Xue et al., 2016) consists of not only predicting the discourse relation sense for `Explicit` and `Non-Explicit` relations, but also segmenting `Arg1` and `Arg2` as well as annotating the discourse connective in case of `Explicit` relations. Such an approach is called *end-to-end discourse relation parsing*. This thesis focuses only on discourse relation sense prediction. Therefore, when using the data from the PDTB for top-level classification, and the C-PDTB fine grained classification, we use the data already segmented between `Arg1` and `Arg2`. The classification task in this thesis consists only of sense prediction in cases where the discourse connective is missing, including `Implicit` relations and `EntRel`.

Furthermore, given the difficulty of automatic IDR, most previous work focuses only on top-level classification; i.e. classifying only the four top-level relations with `EntRel` merged into `Implicit` class EXPANSION as preferred by (Pitler et al., 2009; Rutherford and Xue, 2014; Ji and Eisenstein, 2015). When comparing our work with these authors, we also focus only on top-level classification. The first work on classifying top-level senses on the PDTB focusing exclusively on `Implicit` relations was carried out by (Pitler et al., 2009). Given the unbalanced dataset, as shown in Table 2.2, the authors formulated the task as four one-vs-all binary classifiers. The training, validation and test datasets are thus composed of positive and negative samples. Our specific procedure for our experiment is discussed in Chapter 3.

When comparing our work with those of CoNLL-2016, we perform fine-grained classification where we predict the low-level subtypes. In fine-grained classification, `EntRel` is not merged with `Implicit`'s EXPANSION class, it is predicted along with other senses as shown in Table 2.3. To facilitate the reading, for each experiment in this thesis, the task of either top-level or fine-grained

classification will be specified. In all cases, when classifying top-level senses, the PDTB dataset with the TOP-split is used. When classifying fine-grained senses, the C-PDTB dataset with the PDTB-split is used.

2.6 Previous Approaches to Implicit Discourse Relation Recognition

We turn to reviewing previous work on IDR over the PDTB. We split this section into *conventional* machine learning, and neural network approaches. By *conventional* we simply refer to non-neural network approaches, including conditional random fields (CRFs), support vector machines (SVMs) and naive Bayes classifiers. The split is motivated by the dramatic shift in IDR research between 2015 and 2016, which saw much interest in deep learning, along with the wider NLP community. In Section 2.6.1 we also include rule-based approaches for brevity, which are technically not machine learning algorithms.

2.6.1 Conventional Machine Learning Approaches

Early work by (Pitler et al., 2009) established the first results on IDR over the PDTB, and commonly used as baseline ever since. They showed how a CRF classifier (Lafferty et al., 2001) could outperform a naive Bayes classifier given certain features. The features used by the CRF that were strongest indicators of discourse relation included word polarity, verb classes and orientation, and certain lexical features. Additionally, these features are not consistent across discourse relation classes. For example, for CONTINGENCY, the best features were verb information and first, last and first three words, whereas for expansion they were polarity tags and inquirer tags. Commonly used features are listed in Table 2.5.

(Zhou et al., 2010) showed how to augment the feature model with a language model to get better performance for certain discourse relation senses. (Rutherford and Xue, 2014) showed that Brown cluster pair features contribute to IDR.

Various combinations of the features listed in Table 2.5 are typically used with the following algorithms: naive Bayes (Pitler et al., 2009; Park and Cardie, 2012), SVM (Zhou et al., 2010; Rutherford and Xue, 2014), maximum entropy (Pitler et al., 2009; Rutherford and Xue, 2014),

Feature	Description
FirstLast-First3	The first and last words as well as the first three words of each argument are extracted.
Polarity	Words are assigned polarity according to the Multi-perspective Question Answering Opinion Corpus (Wilson et al., 2005). The feature is the number of negative positive, non-negated positive, negative and neutral sentiment words.
Verbs	The number of verb pairs across `Arg1` and `Arg2` which come from a common verb class.
Word Pairs	After stemming, all words in `Arg1` are grouped into W_1, words in `Arg2` are grouped into W_2. Then, possible word pairs $(w_i, w_j)(w_i \in W_1, w_j \in W_2)$ are generated.
Inquirer Tags	Look at semantic categories of each word in discourse arguments. The tags are drawn from the General Inquirer Lexicon (Stone and Hunt, 1963), which correspond to negated and non-negated fine-grained semantic classification for the verbs in each argument.
Production Rules	Extract all possible production rules from arguments, check whether rules appear in `Arg1`, `Arg2`, or both.
Money, Percentages and Numbers	The count of: currency symbols and abbreviations, percentages signs or the word "percent", as well as numbers in each discourse argument.
Dependency Rules	Extract all rules form the dependency trees of arguments. Define 3 binary features for each rule and check whether the rules appear in `Arg1`, `Arg2` or both.
Context	If previous or following discourse is an `Explicit` discourse relation, use the connective as feature.
Brown Cluster	Each word in `Arg1` and `Arg2` is mapped to its corresponding Brown cluster assignment, which can be 1 of 3.200 clusters. The Cartesian product is calculated over the words in `Arg1` and `Arg2`.

Table 2.5: Commonly used features for implicit discourse relation recognition (Pitler et al., 2009; Zhou et al., 2010; Park and Cardie, 2012; Biran and McKeown, 2013; Rutherford and Xue, 2014; Xue et al., 2015).

System	Learning Methods	Resources Used
ECNU	Naive Bayes, MaxEnt	Brown cluster, MPQA subjectivity lexicon
Concordia	C4.5	ClearTK, syntactic parse
Trento	CRF++, AdaBoost	Brown clusters, dependency/phrase structure parses
NTT	SVM	Brown clusters, dependency trees
AU KBC	CRF++, rules	MPQA, VerbNet, Brown clusters
Dublin 2	LibSVM, Theano, Word2Vec	Brown clusters

Table 2.6: System descriptions at CoNLL-2015, selected reproduction of Table 4 from (Xue et al., 2015). The examples were selected to highlight the variety of algorithms used, all of which are conventional machine learning approaches with the exception of Dublin 2.

AdaBoost (Pitler et al., 2009).

At CoNLL-2015 (Xue et al., 2015), for the task of fine-grained sense classification, discourse parsers relied exclusively on many conventional machine learning approaches and features. In Table 2.6, we reproduce part of Table 4 from (Xue et al., 2015) to show the wide spectrum of approaches used by researchers.

Many more systems not shown in Table 2.6 used the same conventional machine learning methods. A single system, Dublin 2 (Okita et al., 2015), used neural networks for sense classification. The system uses a paragraph vector model to obtain phrase embeddings (Le and Mikolov, 2014), but did not perform as well as conventional approaches listed in Table 2.6.

Hand-crafting features for IDR can be extremely tedious and overly task specific. Additionally, best feature combinations vary across discourse relation classes. For the one-vs-all classifiers, (Park and Cardie, 2012) found the best features for COMPARISON were *FirstLast-First3*, *verbs*, and *production rules*, whereas for TEMPORAL they were *polarity*, *inquirer tags* and *production rules*. To reduce the over-reliance on task specific and sparse features, (Rutherford and Xue, 2014) show that adding Brown clustering (Brown et al., 1992) features for IDR leads to significant performance gain, when using along with the traditional features listed in Table 2.5. The feature is the computed Cartesian product of Brown cluster assignment across `Arg1` and `Arg2`. Brown clusters are interesting features as they are data-driven and theory-independent, the same intuition which motivates later neural network approaches to IDR.

An additional issue to relying on hand-crafted features is that performance can also be extremely sensitive to preprocessing. For example (Park and Cardie, 2012) found that for features dependent on lexicons, such as polarity and inquirer tags, a failure to properly stem words decreased F1 score by roughly 10%. The reason is simply because the feature cannot be passed to the classifier if the words are not matched with the lexicon.

Using word embeddings somewhat alleviates the preprocessing issue (Mikolov et al., 2013b) when using pretrained embeddings which include a vast vocabulary size. Various word spelling should be present and have similar vectorized representation, ensuring exploitation of embedding features.

2.6.2 Neural Network Approaches

To our knowledge, (Li et al., 2014) present the first work applying neural networks to discourse parsing. They propose a recursive network (Goller and Kuchler, 1996) that predicts related clauses and their sense on the RST-DT (see Section 2.1 for details on the RST-DT). The network combines two clauses recursively, where the representation of a parent is based on its children. The recursion follows a constituent parse tree structure, a technique developed by (Socher et al., 2013).

A related approach is implemented by (Ji and Eisenstein, 2015) on the PDTB, but focusing solely on sense prediction. The model uses word embeddings based on Word2Vec (Mikolov et al., 2013b). A vectorized representation of an argument is calculated which is then augmented with the vector representation of the argument's entity, if such an argument contains an entity coreferent to the matching argument. In addition to word embeddings, the model includes additional features such as: word pair features, constituent parse features, dependency parse features and contextual features. Their approach achieves state-of-the-art accuracy on level-2 discourse relations as well as top-level classification in the case of 4 one-vs-all binary classifiers. The downside of this approach is its reliance on engineered features. Despite the use of neural networks, the basic recursive network underperforms previous state-of-the-art non-neural network approaches on level-2 multiclass classification with an accuracy of 36.98% vs 40.2% by (Lin et al., 2009). Only once adding the entity semantics and surface features does the network achieve a state-of-the-art 44.59% accuracy. The top-level state-of-the-art results are shown in Table 2.7.

Author	Comparison	Contingency	Expansion	Temporal
(Pitler et al., 2009)	21.96	47.13	76.42	16.76
(Zhou et al., 2010)	31.79	47.16	70.11	20.30
(Park and Cardie, 2012)	31.32	49.82	79.22	26.57
(Rutherford and Xue, 2014)	39.70	54.42	80.44	28.69
(Ji and Eisenstein, 2015)	35.93	52.78	80.02	27.63
(Zhang et al., 2015a)	33.22	52.04	69.59	30.54

Table 2.7: F1-scores of selected work on top-level implicit discourse relation recognition. Note `EntRel` is merged into EXPANSION as is done in the above listed works.

(Zhang et al., 2015a) propose the first neural network approach IDR relying solely on distributed representations of the discourse input, ignoring previously used features such as first, last words, part-of-speech tags and production rules (see Table 2.5). Unfortunately, the model does not perform as well as (Lin et al., 2009) or even non-neural network approaches. However, it is a promising approach which motivated future development on neural networks for IDR, including our own (see Chapter 3).

2.7 Convolutional Neural Networks

As seen in Section 2.6.2, neural network approaches have begun to be used for IDR around 2015. Most of these models are based on convolutional neural networks (CNNs), inspired by Zhang et al. (2015a) and other work on sentence classification with CNNs (such as Kim (2014); Zhang et al. (2015b)). The insight into these many works is that neural networks are better suited at capturing semantic clues between the two arguments of an implicit relation than traditional methods heavily reliant on feature engineering, as in Pitler et al. (2009); Xue et al. (2015).

We introduce convolutional neural networks of the kind used in computer vision. In its classical formulation, a CNN for classification is composed of repeated pairs of convolutional and pooling (subsampling) layers, a penultimate fully connected layer, and finally a softmax output layer (LeCun et al., 1995). For example, the CNN by (LeCun et al., 1995) shown in Figure 2.3 used for hand-writing recognition consists of two iterations of convolution and pooling.

At the heart of a CNN is the kernel. A kernel is a small convolving tensor, commonly referred to as a window, that scans an input and outputs "features". This convolutional layer automatically learns local relationships between neighbouring words, while upper hidden layers can represent

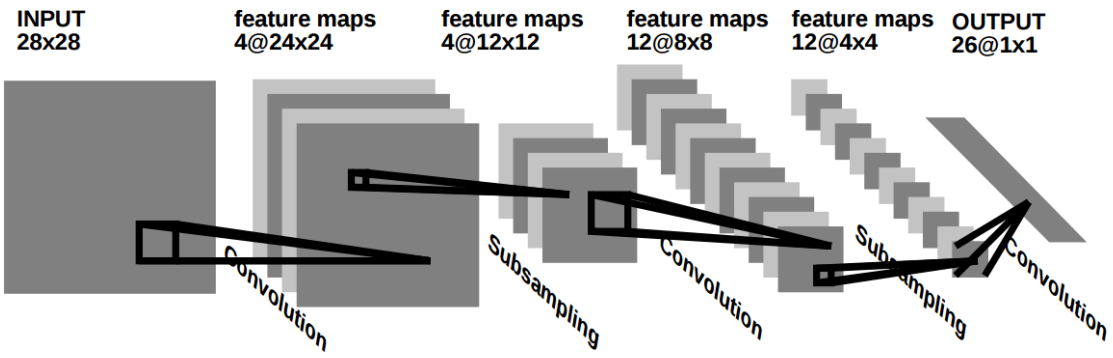


Figure 2.3: A classic convolutional neural network for image processing; in this example, handwriting recognition. Credit: (LeCun et al., 1995)

hierarchical features, akin to parse trees in the linguistic context. As information flows higher in the neural network, more "abstractive" features are learned.

Consider the simplified setup shown in Figure 2.4 for the input sentence *It has made a promising start*. One way to think of convolutions is as a sequence of operations. The operations centered at the words *has* and *made* result in output vectors c_2 and c_3,

$$\begin{aligned} c_2 &= f(W x_{has} + b) \\ c_3 &= f(W x_{made} + b) \end{aligned} \tag{5}$$

Where $W \in \mathbb{R}^{h \times k}$ is the kernel weight matrix multiplied by vector $x_{has} = [x_1, x_2, x_3]$ and $x_{made} = [x_2, x_3, x_4]$, h is the convolution output size, k is the window width, and f is a non-linearity. The parameters W and b are shared throughout the sequence. The nodes in Figure 2.4 represent real-valued numbers. In this simplified setup, each word is swapped with a real number found in a lookup table. Hence the entire input x is simply a real-valued vector. In Figure 2.4, the kernel tensor is a 3×3 matrix with 9 values (weights) learned by the neural network. A single multiplication of this matrix over the partial sequence $[x_2, x_3, x_4]$ results in the feature vector $[c_{3,1}, c_{3,2}, c_{3,3}]$.

The CNN effectively learns how to combine tuples of neighbouring words (3 words in this case) throughout the sentence. The convolution thus maps the input layer into an abstracted, yet constrained representation. This neighbour constraint results in the output layer having *some* properties

approximating the conditional probability distribution of a word sequence (see Section 3.2.4). The convolutions are followed by non-linearities and pooling, which we describe in Section 3.3.1.

2.8 Recurrent Neural Networks

Recurrent neural networks (RNNs) in the most general terms refer to neural networks with a computational node whose output at a time step is used as input at the next time step. Originally, an RNN referred to a very specific formulation which we introduce next. To reduce confusion, we refer to such a network as a Vanilla RNN, and RNN to mean any type of recurrent network. The original formulation of recurrent neural networks of the kind introduced for NLP (Rumelhart et al., 1986; Werbos, 1990; Elman, 1990) have the following simple form:

$$h_t = f_h(W_h x_t + U_h h_{t-1} + b_h) \tag{6}$$

$$y_t = f_y(W_y h_t + b_y) \tag{7}$$

where x_t is the input vector, h_t is the hidden state, y_t is the output vector, W, U are trainable parameter matrices, and b is a trainable vector. f_h and f_y are non-linear activation functions such as sigmoid or softmax. The corresponding structure is illustrated in Figure 2.5.

As shown in Figure 2.5, an RNN's output is closely aligned with its input. The alignment constraint is a reasonable assumption for sequential modelling, such as language modelling, where the current time step's output is the next time step's input. However, this constraint is inappropriate for predicting an entire sequence only once the network is given an entire input sequence, such as in dialogue generation or translation. In addition to the alignment issue, the source and target sequences can vary greatly in length. Mapping a variable length sequence to another variable length sequence is referred to as transduction (Graves, 2012). Encoder-decoders are the neural network approach to model such tasks.

Although several neural network approaches have been proposed for IDR, to our knowledge none have investigated the use of encoder-decoder models with attention, an approach successfully applied to machine translation. To improve translation, notably for longer sentences, a neural

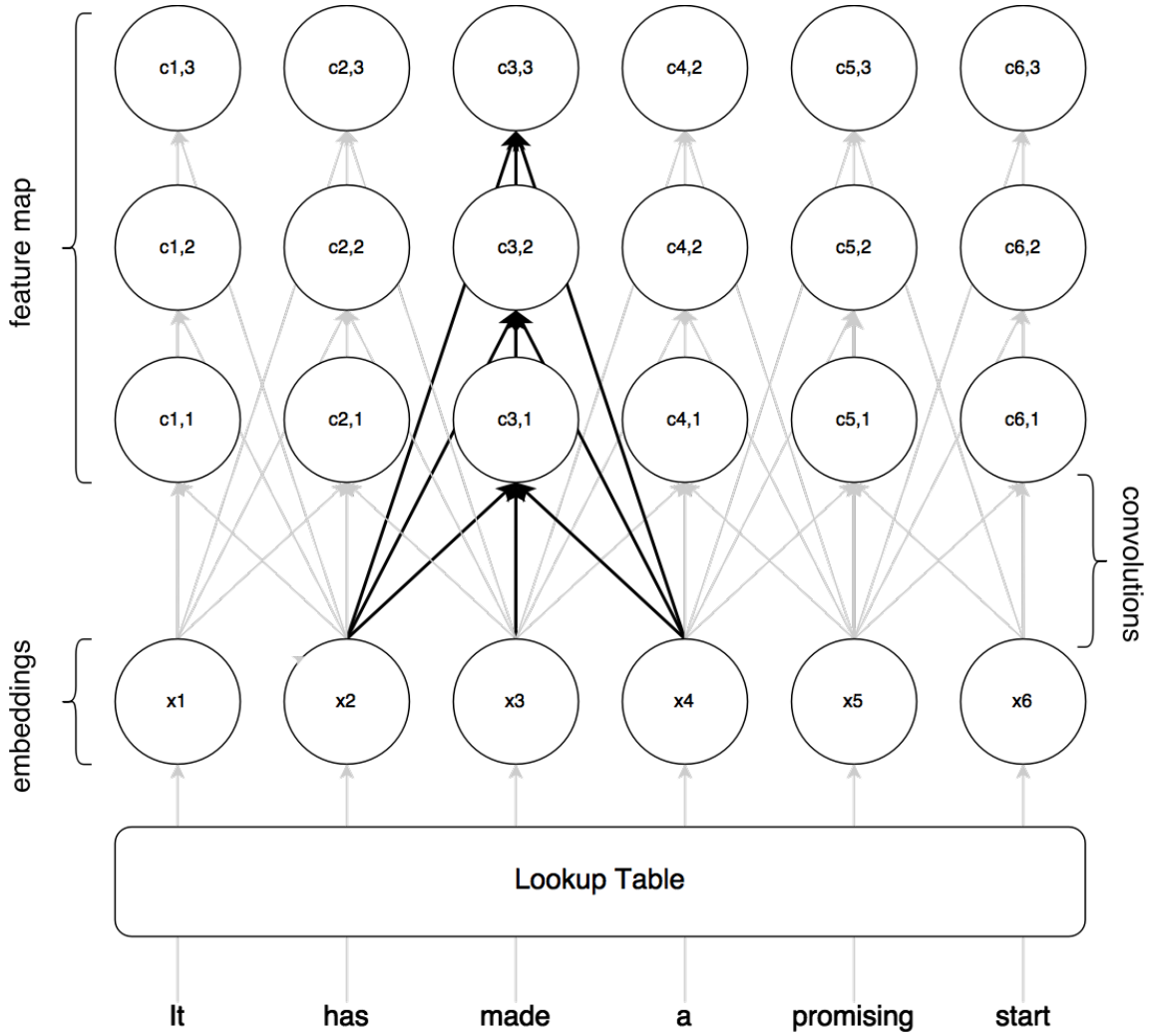


Figure 2.4: A simple convolution operation with a kernel of window size 3, stride of 1, and output of 3. In this simplified setup, words are represented by a single real number. The kernel begins at the word *it*, convolves over *it* and *has* and outputs a real-valued vector of size 3, called a *feature*. The kernel then moves one unit to the right (stride of 1) and convolves over *it*, *has* and *made* and again outputs a vector. Since the input is a single unit for each word, the window size is 3 by 1, and output size is 3, there are: $(1 \times 3) \times 3 + 3 = 12$ kernel parameters to learn. These parameters are shared across each convolution operation. The concatenation of all features forms a feature map. Note that the bias term is omitted in the figure for clarity.

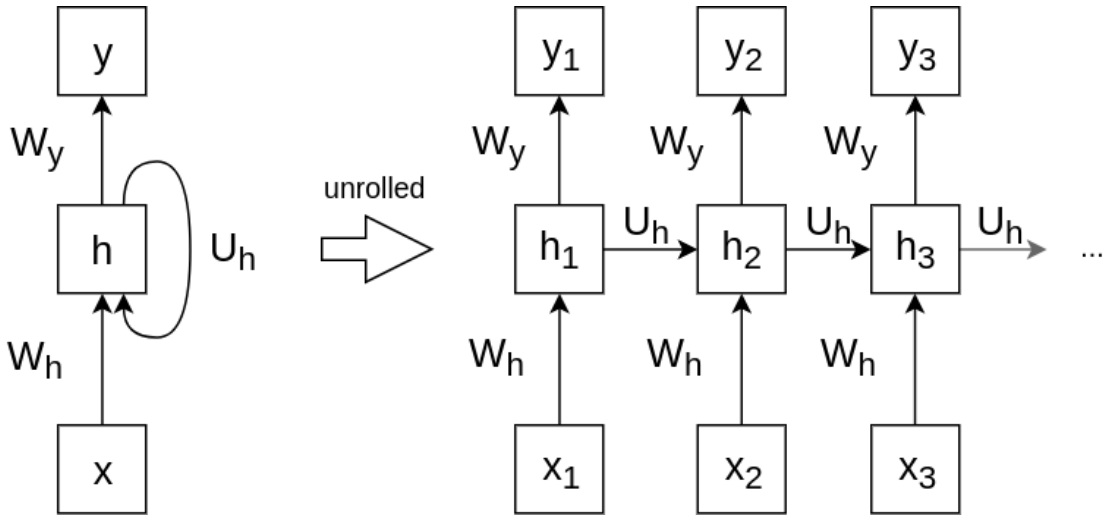


Figure 2.5: A simple recurrent neural network in recursive form (left) and unrolled through time (right). The matrices W_y, W_h and U_h are shared across time and trainable through backpropagation. This graph shows the recurrent neural network at training time where x_t may be equal to y_{t-1}. The training objective is to predict the next input, that is for y_t to equal x_{t+1}. Therefore, at test time the current time step prediction is used as the next time step's input: $x_t = y_{t-1}$.

translation model is augmented with an attention mechanism uniquely purposed for capturing alignment (Bahdanau et al., 2015). The alignment model scores how well the input words from the source language match output words in the target language. Inspired by this work, in Chapter 4 we used attention to leverage this alignment scoring for IDR as word-pair features have be shown to contribute to IDR (Pitler et al., 2009; Biran and McKeown, 2013). However, unlike these methods we make no feature engineering.

In this chapter we reviewed the main discourse theories and explained how these relate to the work in this thesis on implicit discourse relation recognition. We introduced the RST framework with hierarchical discourse structure and concluded the PDTB structure was more appropriate to our research due to its simpler assumptions and larger dataset. The most relevant features of the PDTB dataset were introduced: the difference between `Explicit`, `Implicit`, `EntRel` relations, as well as details of top level discourse relations senses.

We reviewed previous work on IDR and noted how much work focused heavily on engineering

features with varying levels of success. In addition, best features were shown to be inconsistent across discourse relation senses, making a successful IDR classifier even more task specific. We presented preliminary work on IDR with neural networks as well as other NLP research which motivates the focus in this thesis on neural network architecture as opposed to feature engineering. Finally, we explained the necessary background theory to CNNs and RNNs on which Chapters 3 and 4 are built.

Chapter 3

Convolutional Networks for Implicit Discourse Relation Recognition

In this chapter we present our work on the use of convolutional neural networks (CNNs) for `Implicit` discourse relation recognition (IDR) and examine the various approaches we experimented with.

3.1 CNN for NLP

Since their earliest inception convolutional neural networks, or CNNs, showed great potential in computer vision classification tasks. In the original work formulating CNNs, (LeCun, 1989) showed how a network constrained to local connections and shared weights could increase digit recognition accuracy by more than 10% on a given dataset. The constrained network accomplishes this by extracting local features that are combined at higher layers to form more abstract information.

As discussed in Chapter 3, CNNs, along with other deep neural networks, emerged onto the mainstream in the last few years due to their impressive performance in many realms, such as computer vision (Krizhevsky et al., 2012), speech recognition (Graves et al., 2013) and machine translation (Bahdanau et al., 2015). The first work by (Krizhevsky et al., 2012) showed how CNNs could significantly outperform previous, non neural network methods, when given enough capacity.

Adopting neural network architectures for NLP tasks entails particular challenges not present

in the field of computer vision, notably the question of how to convert the input (i.e. words) into a numerical representation for the neural network's input layer. The basic setup for neural networks for text classification on which later work is based is presented in (Collobert and Weston, 2007). The authors show how a shallow and simple multi-layer perceptron (MLP) performs comparably with state-of-the-art systems on semantic role labeling. Not only does the system perform well but it learns much faster all the while not relying on hand-engineered features. The work is also an early example of learning a task for NLP that is performed fully end-to-end with a neural network. In the last few years, convolutional neural networks have been shown to be effective for several NLP tasks such as text classification (Collobert et al., 2011), semantic parsing (Yih et al., 2011) and sentence modeling (Kalchbrenner et al., 2014).

In Section 3.2 we detail our first experiment on IDR with a CNN, a replication study, including how we preprocess our inputs in Section 3.2.1, the details of this CNN architecture in Section 3.2.2, and how we prepared the dataset for IDR in Section 3.2.3. In Section 3.2.4 we show our results and some interpretations explaining the downside of this approach. This motivates our work on a new CNN architecture for fine-grained classification introduced in Section 3.3. In Sections 3.3.1 and 3.3.2 we give the details of this architecture and its results in Section 3.3.3. In Section 3.4 we propose a hierarchical pipeline to IDR, a new approach. In Section 3.5 we provide an additional new approach, combining a CNN with an LSTM.

3.2 Baseline CNN: A Replication Study

Our first system to classify `Implicit` discourse relation uses a CNN based on work by (Zhang et al., 2015a), which is in turn based on previous work by (Collobert et al., 2011) on text classification with CNNs. As far as we know, (Zhang et al., 2015a) were the first to apply neural networks for IDR on the Penn Discourse Treebank. A previous work on discourse relation recognition using neural networks was proposed by (Ji and Eisenstein, 2015). In contrast to (Zhang et al., 2015a), their work used a recursive neural network, an approach tailored to the unique hierarchical tree structure of RST discourse relations, an approach not necessarily applicable to PDTB style discourse structure.

We now detail our implementation, which may be different to (Zhang et al., 2015a) as not all details were published.

3.2.1 Input

The first step is preprocessing the text into a form compatible for our neural network. All words are converted to lower text. Word contractions, such as "I'll" are split into two words "I" and "ll". All words are then tokenized by white space, hence in the rest of this thesis we will refer to these units as *tokens* as opposed to words. Representing tokens into a meaningful form for a neural network is a particular challenge for natural language processing which other domains do not suffer from, such as with computer vision where a pixel value is a natural fit. To address this, we follow a preprocessing method, now standard, based on the pioneering work by (Bengio et al., 2001; Schwenk and Gauvain, 2002) which focuses on learning a distributed representation for words.

We form a vocabulary V of all tokens used in our dataset. A discourse argument in our dataset is represented as a vector s of length n. Vector s is composed of tokens z_i, represented as an index mapped to a word in V. The index value i is related to the frequency of a word in our dataset, so that frequent tokens take on lower index values. We convert each token index z_i into a dense vector representation $w_i \in \mathbb{R}^d$ called word embeddings.

These embeddings come from a word embedding matrix $L \in \mathbb{R}^{|V| \times d}$, where $|V|$ corresponds to the number of tokens in V and d is the embedding size. We used the Word2Vec word embeddings (Mikolov et al., 2013b) which were pretrained on the 100 billion words of Google News by (Mikolov et al., 2013b), and are available online[1]. A lookup vector function $u()$ maps z_i to w_i, such that $u : \mathbb{Z}^+ \rightarrow \mathbb{R}^d$. The z_i index value is used as row index in L to map the word. Simply put, $w_i = L_{z_i}$. We concatenate all dense word vectors to form our argument matrix representation:

$$X = \begin{bmatrix} w_{1,1} \\ w_{1,2} \\ \vdots \\ w_{1,d} \end{bmatrix} || \begin{bmatrix} w_{2,1} \\ w_{2,2} \\ \vdots \\ w_{2,d} \end{bmatrix} || \cdots || \begin{bmatrix} w_{n,1} \\ w_{n,2} \\ \vdots \\ w_{n,d} \end{bmatrix} \tag{8}$$

[1] https://code.google.com/archive/p/word2vec/

where $||$ is the concatenation operation.

3.2.2 Model

As indicated in Section 3.2.1, our input to the network X is now a $d \times n$ matrix, where each column represents a word embedding.

Because the length of each discourse argument j may be different, this entails that X may be of a different size for each argument, an issue we must resolve. In the case of standard CNNs, as used in computer vision, the neural network input layer is of a fixed size, which are not necessarily an issue for image classification where samples do not vary within a set, but is rather problematic for text which naturally varies in length. The following equations deal with this issue.

Based on (Collobert et al., 2011) and on early neural networks for discourse relation recognition by (Zhang et al., 2015a) we first seek to replicate the results of (Zhang et al., 2015a). We begin with three convolutional feature operations across rows r in X:

$$c_r^{avg} = \frac{1}{n}\sum_{i}^{n} X_{r,i} \tag{9}$$

$$c_r^{min} = \min(X_{r,1}, X_{r,2}, \ldots, X_{r,n}) \tag{10}$$

$$c_r^{max} = \max(X_{r,1}, X_{r,2}, \ldots, X_{r,n}) \tag{11}$$

This results in three vectors of size d, invariable to length n. The same set of operations is performed on the second discourse argument. The 6 resulting vectors are then concatenated into a:

$$a = [c_{\text{arg1}}^{avg}||c_{\text{arg1}}^{min}||c_{\text{arg1}}^{max}||c_{\text{arg2}}^{avg}||c_{\text{arg2}}^{min}||c_{\text{arg2}}^{max}] \tag{12}$$

This is followed by a $tanh$ normalization to eliminate manifold differences among the features:

$$h = \frac{\tanh(a)}{\|\tanh(a)\|} \tag{13}$$

Note that in this context "$\|$" is the normalization operation (not to be confused with concatenation $||$) which scales the components of $\tanh(a)$ to unit length. Vector h is connected to an output

layer by an affine transform:

$$s = W_o h + b \tag{14}$$

Where matrix $W_o \in \mathbb{R}^{\text{size}(h) \times K}$, K is the number of labels (classes) and therefore vector $s \in \mathbb{R}^K$. Vector s can be interpreted as the unnormalized log probabilities of each element j in s belonging to class k. We compute the actual probability that s_j, a score in s, is discourse relation k given our original matrix input X and model parameters θ, with the softmax function:

$$P(y = j|X, \theta) = \frac{e^{s_j}}{\sum_{k=1}^{K} e^{s_k}} \tag{15}$$

The vectorized implementation of the softmax function takes in vector s and outputs vector $\hat{y}$. Each element j in vector $\hat{y}$ is real-valued between 0 and 1, while vector $\hat{y}$ sums to 1. This produces a proper probability distribution over all possible discourse relation senses. Additionally, due to the exponential term, the softmax *maximizes* the probability of the highest scoring term in s and "pushes" less probable scores towards zero, but never to zero (hence the name softmax). In our implementation, the softmax is computed in a single step over all scores, resulting in vector $\hat{y}$. Due to the softmax, vector $\hat{y}$ can be a close approximation of the target one-hot encoded vector y.

For a single training sample pair x, y, we compute the distance between our predicted class probability distribution $\hat{y}$ and true distribution y with the cross-entropy error:

$$H(\hat{y}, y) = -\sum_{j=1}^{K} y_j \log(\hat{y}_j) \tag{16}$$

The training objective consists of minimizing the mean cross-entropy error H between predicted label $\hat{y}$ and gold label y over all samples m, with weight-decay regularization:

$$J(\theta) = \frac{1}{m} \sum_{t=1}^{m} H(\hat{y_t}, y_t) + \frac{\lambda}{2} \|\theta\|^2 \tag{17}$$

Relation	Train	Validation	Test
COMPARISON	1942 / 1942	197 / 986	152 / 894
CONTINGENCY	3342 / 3342	295 / 888	279 / 767
EXPANSION	7004 / 7004	671 / 512	574 / 472
TEMPORAL	760 / 760	64 / 1119	85 / 961

Table 3.1: Positive/negative split for the 4 one-vs-all binary classifiers for `Implicit` sense classification, where `EntRel` is included in class EXPANSION. The number of training samples is based on (Zhang et al., 2015a). The train, validation and test split is discussed in Section 2.2, which we call TOP-split.

3.2.3 Data

For top-level classification, we used the PDTB dataset with the WSJ section breakdown as discussed in Section 2.2. In Section 2.5, following the practice established by (Pitler et al., 2009), top-level classification is formulated as four one-vs-all classification, where each classifier is one of four top-level senses. While the source PDTB dataset is the same across authors, the positive and negative training data is not necessarily consistent.

For each sense classifier COMPARISON, CONTINGENCY and TEMPORAL we selected all positive samples in WSJ sections 2 to 20. Negative samples are randomly sampled from sections 2 to 20, excluding positive samples, where the number of chosen samples is equal in count to the positive training set. For the EXPANSION classifier, on the other hand, the full positive dataset outnumbers the possible negative set, as observed by the unbalanced dataset in Table 2.2. We randomly undersampled the positive EXPANSION dataset and oversampled the negative dataset so that our training set size would equal that of (Zhang et al., 2015a). It is not clear from (Zhang et al., 2015a) why they chose this strategy.

The validation and test sets on the other hand are not equally balanced like the training set. For each binary classifier, the positive set consists of all positive labels in PDTB, whereas the negative dataset includes all other senses combined. As observed in Table 3.1, this results in a highly unbalanced validation and test dataset, notably for the TEMPORAL sense class.

	COMPARISON	CONTINGENCY	EXPANSION	TEMPORAL
ours	25.14	42.86	71.05	12.64
(Zhang et al., 2015a)	30.40	48.41	66.05	28.71

Table 3.2: F1-score of top-level classification comparing our replication study to the original experiment by (Zhang et al., 2015a)

3.2.4 Results and Analysis

We show our replication results in Table 3.2. As seen in Table 3.2, our implementation outperformed (Zhang et al., 2015a) for the EXPANSION class, the largest sense class, but underperformed for COMPARISON, CONTINGENCY and TEMPORAL, the smaller classes. These results vary from the original experiment of (Zhang et al., 2015a), most likely due to differences in the training set sample selection, different optimizers and hyperparameters. It is difficult to know the exact difference since the original code is not available.

The model presented in Section 3.2.2 proposes a solution to the variable length dilemma, however it seems that performance suffers from convolutional operations *across embeddings*, as opposed to convolutions *across time*. This would mean it models language as a bag of words and ignores the structure of language. Consider the language modeling setup. A statistical language model estimates the probability distribution of a sequence of words by predicting the probability of a word w_i given the history of words until w_{i-1}:

$$P(w_0, \ldots, w_N) = P(w_0) \prod_{i=1}^{N} P(w_i | w_0, \ldots, w_{i-1}) \tag{18}$$

Recall from Equations 9, 10 and 11 that our convolutional operations operate on ordered input X but are agnostic to the order of a language model as in Equation 18. In no way do the equations take advantage of the possible meaning extracted from word order. Furthermore, the equations naively operate on all embeddings uniformly, without weighing any word more than another. The model cannot detect keywords that are strong signals of a discourse relation nor can they measure pair-wise interactions. This motivates us to explore a different kind of CNN which convolves locally over the input, considering only a limited number of words as we see next in Section 3.3. This also motivates fully auto-regressive models which we discuss in Chapter 4.

3.3 Our CNN at CoNLL-2016

In this section we turn to convolutional neural networks (CNNs) with convolutional operations of the kind found in computer vision. We describe a CNN for `Non-Explicit` discourse relation recognition. Recall that `Non-Explicit` relations refers to `Implicit`, `EntRel` and `AltLex` relations (see Section 2.3 for details). The starting point of our setup is inspired by prior work on text classification with convolutional neural networks, notably by (Collobert et al., 2011; Kim, 2014).

Our CNN implementation is discussed in the context of our participation to the Twentieth Conference on Computational Natural Language Learning (CoNLL-2016) Shared Task on Multilingual Shallow Discourse Parsing (SDP) (Xue et al., 2016) as described in Section 2.3. The overall goal of the task is to identify discourse relation arguments, discourse connectives in the `Explicit` case, as well as the discourse relation sense.

The dataset used at CoNLL-2016 is the Penn Discourse TreeBank (PDTB) (Prasad et al., 2008) with the C-PDTB split (see Section 2.3 for details) which contains a tuple of argument 1 text (`Arg1`), argument 2 text (`Arg2`), discourse connective and sense relation. As shown in Table 2.3 the `Non-Explicit` dataset contains 17,289 training samples, 737 validation samples and 986 test samples. The blind dataset was specially created for this task and comes from the Wikinews dataset (Xue et al., 2015). The blind dataset contains 653 `Non-Explicit` discourse relations. Please see Section 2.2 and 2.3 for details of the two datasets. The two datasets also contain linguistic features such as POS tags and syntactic trees, but these are not used in our case. In the full task, a syntactic parser must parse each argument, identify the discourse connective as well as the fine-grained relation. The task is called "shallow" since, unlike Rhetorical Structure Theory (Mann and Thompson, 1988)(described in Section 2.1), a system must not output a tree of relations across an entire text, but only between two argument pairs. Discourse arguments can be clauses, sentences or phrases.

In 2015, the CLaC lab participated in the CoNLL Shared Task on Shallow Discourse Parsing (Laali et al., 2015). At the time, the CLaC Discourse Parser did not address `Non-Explicit` relations. For the 2016 edition we developed a method for IDR which was added to the end-to-end

CLaC parser (Laali et al., 2016). We briefly review our parser and then focus on our network for `Non-Explicit` discourse relation recognition.

3.3.1 Pipeline System

As shown in Figure 3.1, the end-to-end parser (Laali et al., 2016) is split into two main modules: `Explicit` Discourse Relation Annotation (EDRA) and `Non-Explicit` Discourse Relation Annotation (NEDRA). Raw text units are first passed through the EDRA for segmentation and annotation. Those discourse units which cannot be labelled are then piped into the NEDRA, as can be seen in the last two modules of the pipeline in Figure 3.1. The relation labeler neural network classifies the samples as containing a discourse relation and those not containing any relation. Samples that do contain a relation are piped to the sense labeler neural network for final classification, while the remaining samples are labeled as not containing any relation. The relation labeler and sense labeler are convolutional neural networks with an identical architecture except for the size of the output layer.

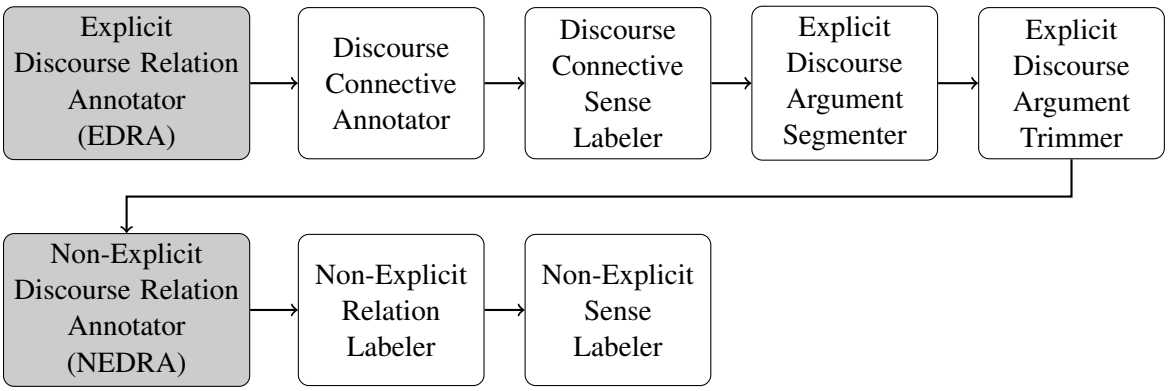


Figure 3.1: Pipeline of the CLaC Discourse Parser

3.3.2 Model

Similarly to the experiment in Section 3.2, the input to the models are pretrained word embeddings from the Google News corpus, as trained with Word2Vec [2]. Words not in the word embeddings are randomly initialized. We test static and non-static word embeddings and choose to keep them

[2] https://code.google.com/archive/p/word2vec/

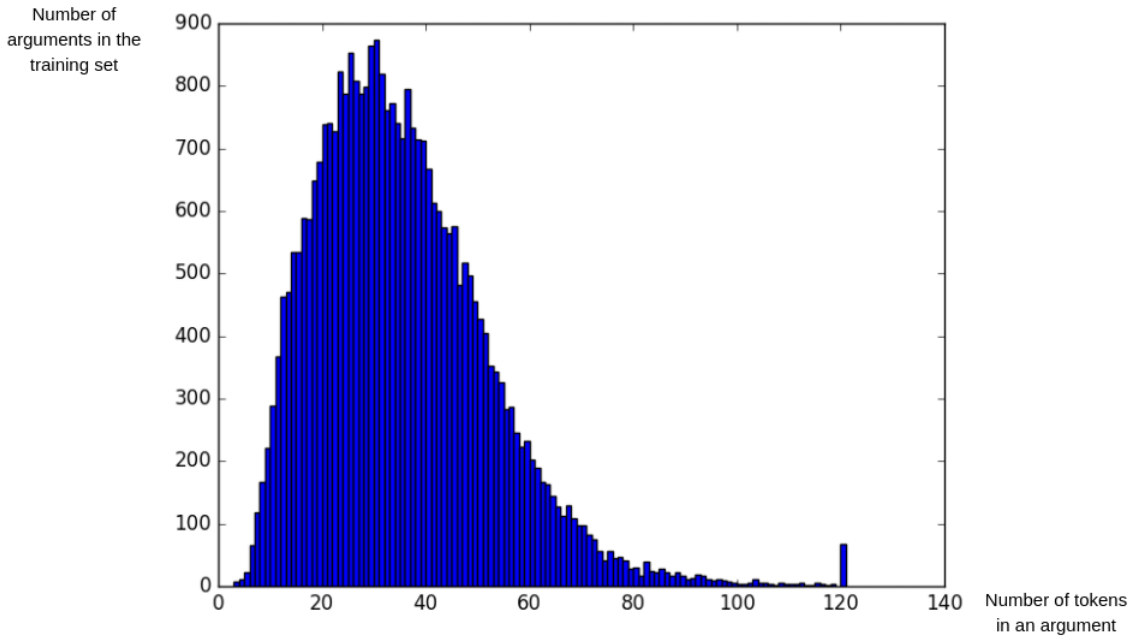


Figure 3.2: Histogram of the number of tokens per sample in the C-PDTB training set. Texts longer than 120 words are included in the 120 bar.

static as they slightly outperform non-static embeddings. Non-static embeddings are not allowed to vary during training.

In this neural network setup, we cannot allow for variable input length as we did in the model of Section 3.2.2. From Figures 3.2 and 3.3 we observe most text is clustered between 10 and 60 tokens.

Typically convolutional network inputs are zero padded to match the size of the longest input (Zhang and Wallace, 2015; Kim, 2014). Since the training set contains a few unusually long arguments, we limit the arguments to 60 tokens each, which accounts for 99.5% of the tokens in the training set (see Figure 3.2). Each input to the networks is thus composed of two discourse arguments, truncated or padded to 60 tokens. This reduces the maximum length of `Arg1` from 1000 to 60 words, and that of `Arg2` from 400 to 60 words. This dramatically decreases the model complexity with insignificant impact on performance. The two arguments are then concatenated to form a single input. Each token is then replaced with their embedded vector representation, giving us input matrix X. A visualization of the input preprocessing pipeline is shown in Figure 3.4.

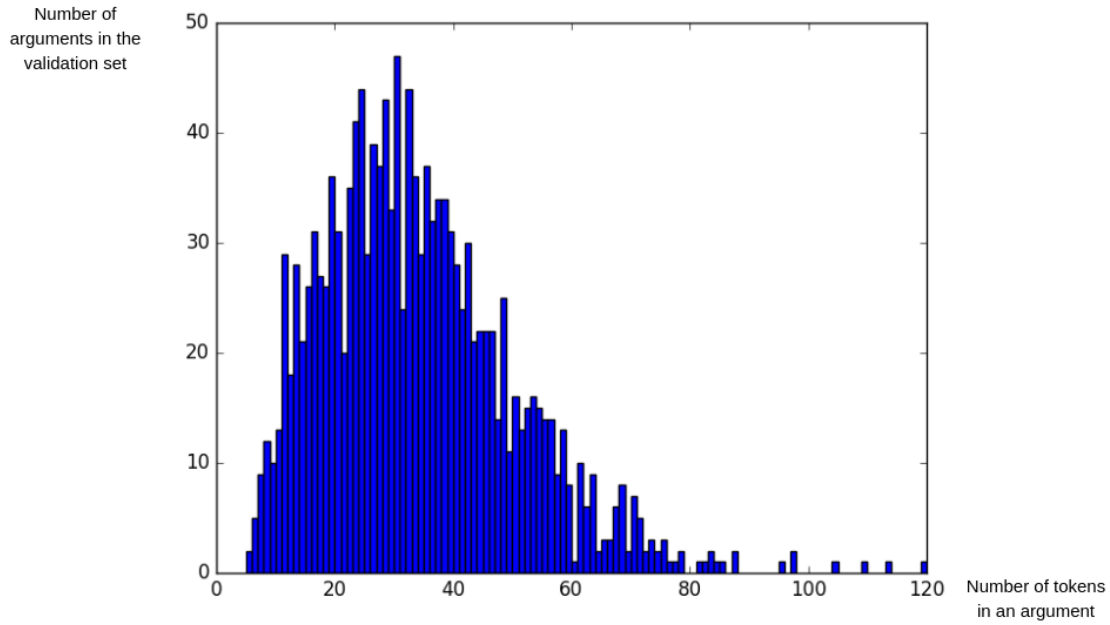


Figure 3.3: Histogram of the number of tokens per sample in the C-PDTB validation set. Texts longer than 120 words are included in the 120 bar.

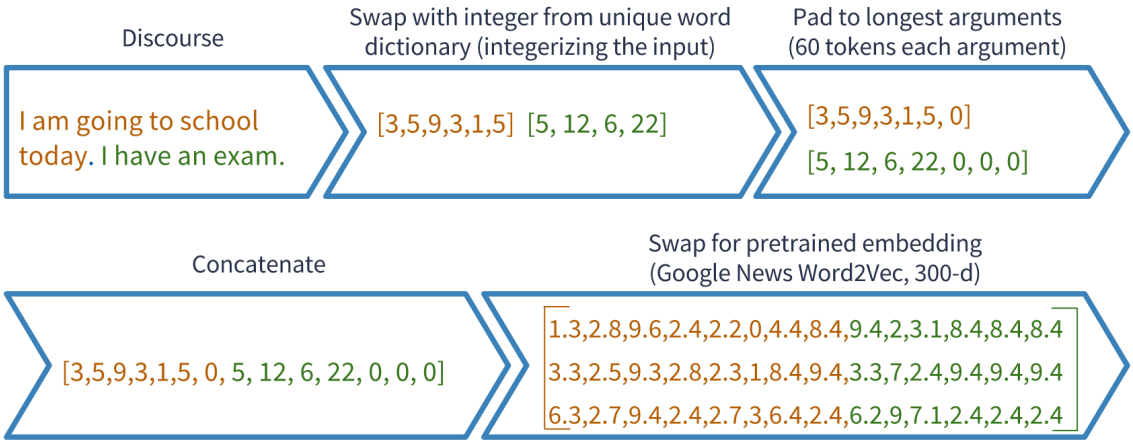


Figure 3.4: Input preprocessing pipeline. The raw discourse text "I am going to school today" (`Arg1`) and "I have an exam" (`Arg2`) are tokenized and integerized, where the integer is a dictionary index. Each argument is padded or truncated to equal the standard length of 60 tokens after which the two are concatenated. The last step embeds the input only immediately before being passed to the network to conserve memory. Each token in a mini-batch is swapped for an embedding via a lookup table.

The input matrix X has the same structure as that of X in Equation 8 and follows the same setup. However, unlike our simplified example in Figure 2.4, each token in our input is of dimension k. For the moment, consider a kernel with a single output, then the convolution consists of kernel $w \in \mathbb{R}^{hk}$. A feature c_i is computed as:

$$c_i = f(W x_{i:i+h-1} + b) \tag{19}$$

Where f is a non-linear function. In our CNN, we choose the exponential linear unit (ELU) as introduced by (Clevert et al., 2015). The ELU diminishes the vanishing gradient effect, as do ReLU and leaky ReLU (Maas et al., 2013), by being the identity function when the parameter is positive and thus having a derivative of one in that region. However unlike ReLU, the ELU and leaky ReLU have negative values, pushing activations closer to zero and hence resulting in faster learning. Additionally, ELU has a small derivative which decreases activation variation. See (Clevert et al., 2015) for a full analysis of this activation function. ELU is computed as:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(exp(x) - 1) & \text{if } x \leq 0 \end{cases} \tag{20}$$

Where α is a hyperparameter controlling the level of saturation. We set α to 1. The convolutions always convolve over the entire word embeddings (1D convolution). All c_i computed at all window locations $x_{1:h}, x_{2:h+1}, \ldots, x_{n-h+1:n}$ are concatenated to form a feature map:

$$r = [c_1||c_2||\ldots||c_{n-h+1}] \tag{21}$$

The convolution/feature map operation is repeated j times, equivalent to the 3 outputs in Figure 2.4, which results in j feature maps, stacked into matrix R, where $R \in \mathbb{R}^{j \times l}$. When using a CNN for text classification, it is key to pool by maximizing over time following a convolution as is observed by (Zhang et al., 2015b; Boureau et al., 2010). We perform a max-over-time pooling over R:

$$p_j = \max_{i \in R} x_{ji}, \; j = 1, \ldots, l. \tag{22}$$

Where vector p is the result of selecting the maximum value in each row j in R. The mot-pool operation selects the maximum element in each row of R, hence the naming *max over time pooling*. The maximum is interpreted as the most *salient* feature in time which is automatically learned by the neural network.

So far we have described the process of convolution and pooling for a single kernel. In our model, we repeat the process for m kernels with varying window sizes. The resulting p vectors are concatenated to form the penultimate layer z:

$$z = [p_1||p_2||\dots||p_m] \tag{23}$$

See Figure 3.5 for a visualization of the concatenated pooled layers. As illustrated in Figure 3.5, we applied 128 feature maps and pooled each one of these. We repeated the entire process 3 times for $w = 3, 4$ and 5, as shown to perform better in (Zhang and Wallace, 2015), and concatenated them together. This gave us a final matrix $M \in \mathbb{R}^{3\times128}$. We reshaped M to a flat vector and applied dropout as our regularization (Srivastava et al., 2014), giving us vector $u \in \mathbb{R}^{384}$. u is fully connected to a softmax output layer where loss is measured with cross-entropy. The network was trained by stochastic gradient descent in mini-batches and optimized with the Adam optimizer (Kingma and Ba, 2015).

3.3.3 Results and Analysis

We trained our model for a maximum of 200 epochs with early stopping if no improvement occurred after 20 epochs. Generally, the performance on the validation set would peak around 75 epochs.

3.3.3.1 Overall Performance

Table 3.3 shows the F1 scores of the CLaC Discourse Parser and other participants on the validation, test and blind datasets. Only the F1 scores for the supplementary task of sense labelling is shown. In this setting, argument 1, argument 2, as well as the discourse connective are given to the system and only the discourse relation need be predicted. As Table 3.3 shows, we achieved

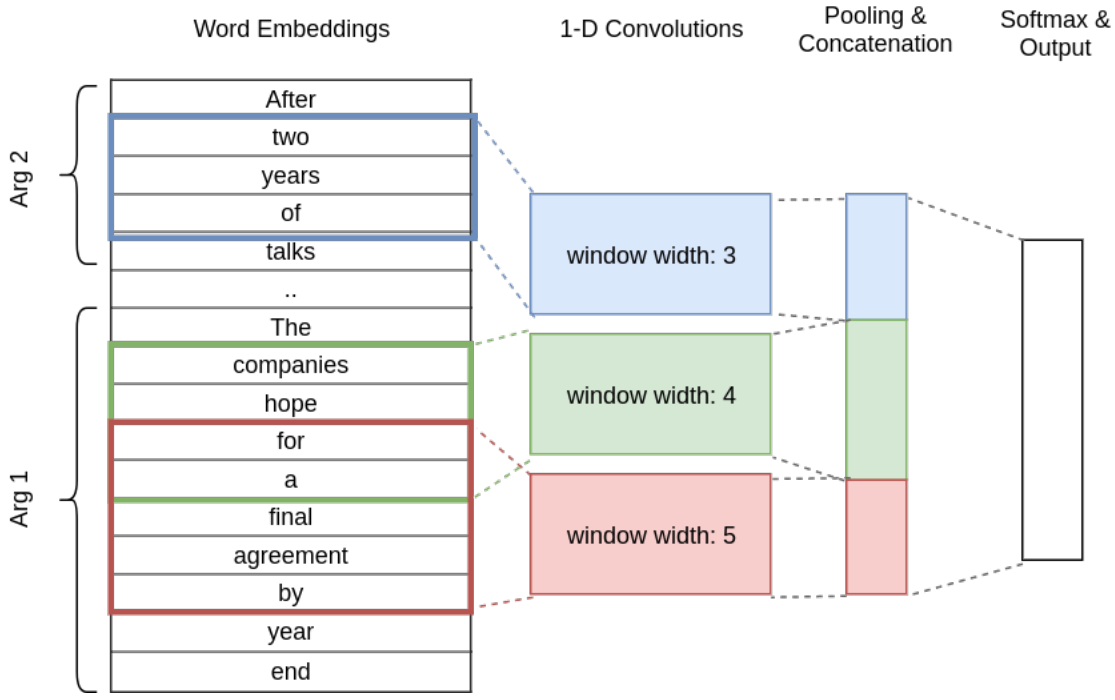


Figure 3.5: A simplified view of our convolutional neural network as part of our discourse parser (Laali et al., 2016) submitted to the CoNLL 2016 Shared Task on Shallow Discourse Parsing (Xue et al., 2016). Arguments 1 and 2 are represented as concatenated word embeddings, followed by 3 convolution operation of different sizes, max-over-time pooling and a softmax layer.

an F1-score of 27.72 on the blind test set while the top system achieved 37.67. We compared our model with that of (Wang and Lan, 2016), called *ecnucs* in Table 3.3. Surprisingly our models had fairly similar approaches. (Wang and Lan, 2016) also implemented a shallow CNN with multiple window convolutions, max-pooling and a fully connected layer. Some minor differences include ecnucs using wider window sizes of 4, 6 and 13 tokens instead of our 3, 4 and 5, as well as the use of the hyperbolic tangent activation function instead of ELU.

		`Explicit` Relations			`Non-Explicit` Relations		
ID	**Reference**	**Validation**	**Test**	**Blind**	**Validation**	**Test**	**Blind**
ttr	(Rutherford and Xue, 2016)	-	-	-	40.32	36.13	37.67
gtnlp	-	90.29	89.48	74.95	40.72	34.95	36.75
tao0920	(Qin et al., 2016)	92.26	89.59	75.74	46.33	38.20	35.38
tbmihaylov	(Mihaylov and Frank, 2016)	91.20	89.80	78.20	40.32	39.19	34.51
ecnucs	(Wang and Lan, 2016)	92.56	90.13	77.41	46.42	40.91	34.18
oslopots	(Oepen et al., 2016)	91.35	90.13	77.17	43.12	33.76	33.84
gw0_1	(Weiss and Bajec, 2016)	91.81	89.48	75.25	34.58	30.21	33.08
ykido	(Kido and Aizawa, 2016)	90.29	90.22	75.43	29.11	22.61	32.31
goethe	(Schenk et al., 2016)	91.35	90.13	76.40	45.42	37.61	31.85
nguyenlab	(Nguyen, 2016)	90.29	88.72	74.77	34.31	28.83	31.42
PurdueNLP	(Pacheco et al., 2016)	89.68	87.96	19.58	38.05	34.45	29.10
CLaC	(Laali et al., 2016)	90.74	89.48	76.22	**37.12**	**28.13**	**27.72**
steven	-	71.19	72.66	64.16	26.68	20.58	23.58
gw0_2	(Weiss and Bajec, 2016)	89.68	15.51	18.35	35.11	18.56	21.29
BIT	(Jian et al., 2016)	23.22	24.62	17.99	17.36	16.58	19.30
aarjay	-	91.50	89.70	78.56	36.85	15.60	9.95

Table 3.3: F1-score of the CLaC Discourse Parser on the discourse relation sense classification supplementary evaluation, in comparison with other submissions. Results include performance on the development, test and blind datasets for both `Explicit` relations and `Non-Explicit` relations.

On the other hand, one major difference is the fact ecnucs convolves over discourse arguments separately, as shown is Figure 3.6. The convolution outputs are concatenated only after the separate max-poolings. By comparing our two models we conclude a neural network, in this case a CNN, can extract meaningful features but only once the lower network layers are structured according to the discourse structure. Technically, the reason is likely due to the max-pooling across time. By concatenating the two discourse arguments, the pooling operation returns the max features across the entire input without distinguishing between the arguments. If the majority of the features returned come from a single argument, the later fully connected layer cannot learn any interaction across arguments. Another high-performing model by (Rutherford and Xue, 2016) also used a CNN which

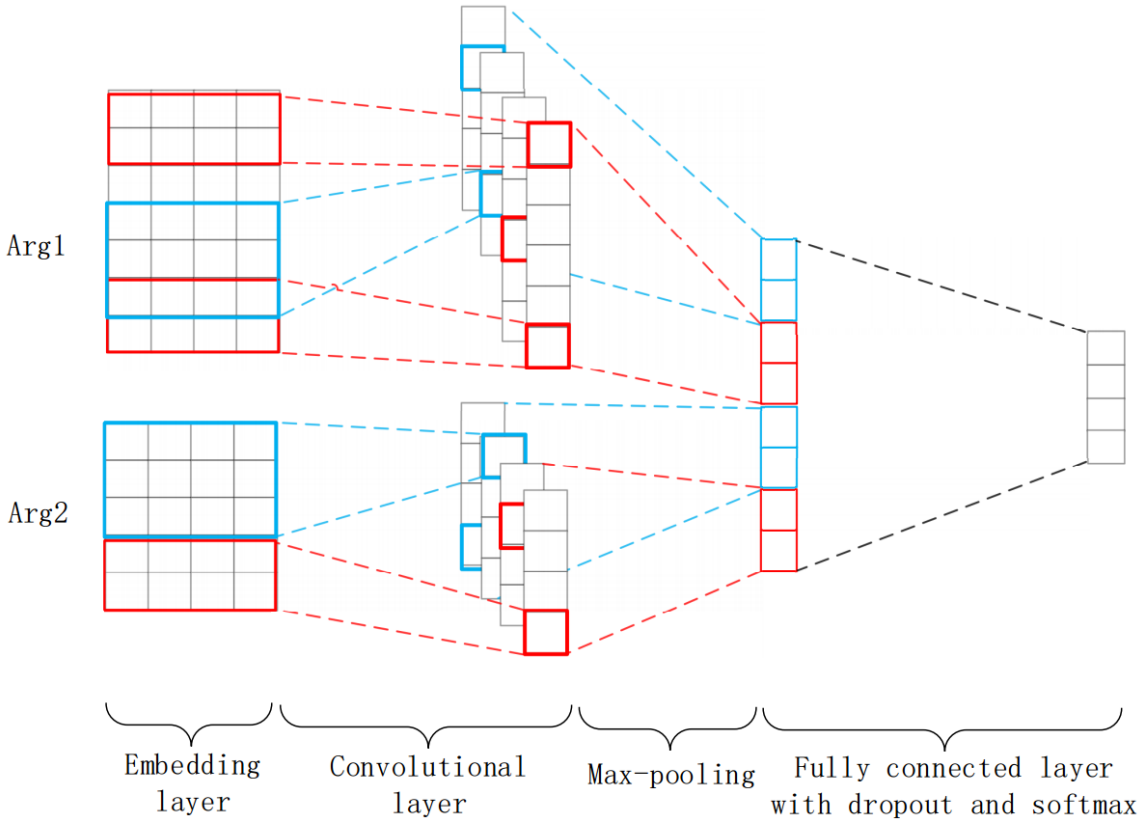


Figure 3.6: ecnucs, a CNN for IDR where discourse arguments are convolved and pooled separately. Credit: (Wang and Lan, 2016)

convolves over separate arguments. This finding is key to building our state-of-the-art model in Chapter 4.

While our work focused only on `Non-Explicit` relations classification, Table 3.3 also provides the performance with explicit relations classification to highlight the difficulty of our task, even given the most favourable setup where no segmentation is required. For example, ecnucs achieves an F1-score of 90.13 on the test set for `Explicit` relations, but only 40.91 on the `Non-Explicit` relation test set, a staggering difference of almost 50 points. This highlights the huge difficulty in classifying `Non-Explicit` discourse relation where a system cannot rely on any connectives. In fact, many discourse connectives almost always map the same discourse relation (Xue et al., 2016) hence greatly facilitating `Explicit` relation recognition. This is why most systems at CoNLL-2016 simply adopted "conventional" machine learning approaches (Xue et al., 2016) and performed quite well on `Explicit` relation recognition (see Table 3.3).

	Validation vs test set		Validation vs blind set	
	Explicit	**Non-Explicit**	**Explicit**	**Non-Explicit**
Excluding outlier	1.53	6.28	17.65	6.88
Including outlier	6.37	7.22	21.15	8.14

Table 3.4: Mean absolute performance difference. We take the absolute difference in performance for all systems (see Table 3.3) between the validation set and the test set, as well as between the development set and the blind dataset. Here we show the mean of these differences for the Explicit and Non-Explicit cases.

As shown in Table 3.3 in both Explicit and Non-Explicit relation classification most systems underperformed when comparing the blind dataset with the test dataset. However we should note a key difference between Explicit and Non-Explicit relation performance. From Table 3.4 we observe that on average the Explicit sense labellers' F1 scores on the test set is only off by 1.53 compared to the validation set, much better than the Non-Explicit sense labeller which drops by 6.28. However, the Explicit labeller is off by a large margin when comparing the development and blind set performance, by as much as 17.65, while the Non-Explicit labeller remains almost the same with only 6.88 difference.

The main reasons the Explicit models do not perform as well on the blind dataset, in terms of relative performance is due to:

(1) The domain difference: The validation and test dataset were extracted from the same original Wall Street Journal dataset, whereas the blind dataset was extracted from Wikinews. We are likely seeing domain overfitting in the case of all Explicit relation recognition models, but do not see this pattern with Non-Explicit models.

(2) Explicit models already perform well: Most Non-Explicit models are neural networks whereas Explicit models are mostly conventional machine learning models based on hand-crafted linguistic features (Xue et al., 2016). Because of this, Non-Explicit models may pick language patterns that are more generalizable to different domains when compared to Explicit models.

Another possibility, however, is that Non-Explicit models simply do not perform well enough to overfit to a particular domain. We can conclude that future work on Explicit sense

Filter Size	F1	Step	Time (in seconds)
2	31.88	85700	4598
3	32.94	91400	5139
5	31.08	78500	6234
7	32.41	51100	7092
9	31.08	70400	8447
11	31.75	54300	10071
15	31.35	71200	12813
20	31.75	32900	11770
30	33.47	71300	24831
50	32.67	90600	38643

Table 3.5: Region size of a single filter. F1 represents the best F1 achieved on the validation set during the experiment. We indicate at which step this accuracy was reached, out of a possible 97,650 steps (150 epochs). Time is the total time in seconds to complete the 150 epochs. The model was trained on an 8-core desktop computer with 16GB of RAM. Neural network operations were computed on the computer's Nvidia K620 GPU.

labelling should focus on models that work well across domains, while `Non-Explicit` models can still continue to be developed on the current dataset. One serious caveat is that most `Non-Explicit` models are implemented with neural architectures which generally require vast amounts of data to perform well (see Chapter 6).

Before submitting our official system to CoNLL-2016, we performed additional analysis to better understand our model, in particular we analyzed the effect of varying the region size and the input representation.

3.3.3.2 Further Analysis

Region Size We experimented with different region sizes to see how this would affect performance. (Zhang and Wallace, 2015) recommends training with a single window and linearly increasing the size, ranging from 2 to 30. We experimented with window sizes of up to 50, after which F1 remained low and training time became too long. Each run ran for 150 epochs, with the best F1 score achieved in the runs shown in Table 3.5. In other experiments (data not shown here) we trained several times up to 600 epochs, however the model would always overfit before 150 epochs.

Our original intuition was that larger filter sizes should capture longer range dependencies,

which should help discern discourse relation types. However this is not supported by the results in Table 3.5. Despite the increase in region size, the F1 score does not seem to increase systematically. We should ignore the small differences in accuracy as we only trained the network once per region size. A more conclusive experiment would require at least 10-fold cross-validation, but time did not permit this. From the *Step* column, we can observe that there is no linear trend in overfitting; some runs overfit at 32,900 steps, and some towards the end of the experiment at 91,400 steps. This is likely due to the neural network random parameter initialization. From the Time column, we can conclude that a smaller window size is beneficial for this task since training time is much shorter. A filter size of 2 is more than 8 times faster to train than a filter of size 50, with comparable accuracy.

Input Representation Recall that our system used at CoNLL-2016 (see Section 3.3.2) used pre-trained word embeddings and kept them static throughout the learning phase. After experimenting on pretrained word embeddings and randomly initialized embedding, we conclude that we actually obtain similar results. When first training, the training accuracy for pretrained embeddings is a little higher than for random embeddings (less than 1%). The larger memory requirements do not justify the use of pretrained embeddings in the case of discourse relation recognition. This is somewhat surprising, as research by (Mikolov et al., 2013b,a) showed that word embeddings trained with the Word2Vec method actually do embed syntactic and semantic information. (Zhang and Wallace, 2015) also concluded that pretrained embeddings were useful in sentence classification. It is not clear why they were not helpful in our system for discourse recognition, especially in the case of `Non-Explicit` relation where we must rely on semantic representation. Neural word embedding are difficult to understand on their own. They are usually understood in the context of analogies. By subtracting the vector "man" from the "king" and adding the vector "woman", we get the vector for "queen"(Mikolov et al., 2013b). We can extract all sorts of analogies, both semantic and syntactic. It is possible that very few or none of these relations are useful for discourse. Neural word embeddings that capture only narrower syntactic/semantic features could be useful. Neural word embedding such as in Word2Vec look at context, i.e. surrounding words. (Ramat-Gan, 2014) have trained neural embeddings based on *syntactic* context as opposed to lexical. It would be interesting to combine these with Word2Vec embeddings for our experiment. As noted in Chapter 6, more

work in this area is required.

3.4 Hierarchical CNN

Our third CNN model attempts to address the difficulty of fine-grained classification. Recall from Section 3.3 that the goal of fine-grained IDR is to maximize the probability of class y given two discourse arguments, $P(y|\text{arg1}, \text{arg2})$, where y is one of the fine grained classes in Table 2.3. Recall that fine-grained senses are part of a hierarchy of senses, as illustrated in Figure 2.2. Using this fact, to do fine-grained classification we augment the training objective as maximizing:

$$P(y|\text{arg1}, \text{arg2}, \text{top-sense}) \tag{24}$$

Where the top-sense must be predicted by a model. If a model can correctly predict the top-sense, then given this additional evidence we can expect a model to better approximate Equation 24 than an objective without this evidence.

3.4.1 Model

Our approach to this new objective is two steps. First, a model classifies the top-level sense. Second, taking the most likely class from that first model, we pass on the fine grained classification to a separate model, restricted to classify a fine-grained sense which is a successor of the top-level sense, as shown in Figure 2.2.

For this experiment, we used the same model as described in Section 3.3, as well as the same preprocessing. Please see Section 3.3 for details on CNN hyperparameters, equations, and input preprocessing.

3.4.2 Results and Analysis

The task is to classify fine-grained senses, exactly as in Section 3.3. However, in our experiment we add the additional task of first classifying top-level senses, as seen in Section 3.2. To limit the number of models, we created a single multi-class top-level model which classifies a discourse as belonging to one of the following discourse sense class: COMPARISON, CONTINGENCY,

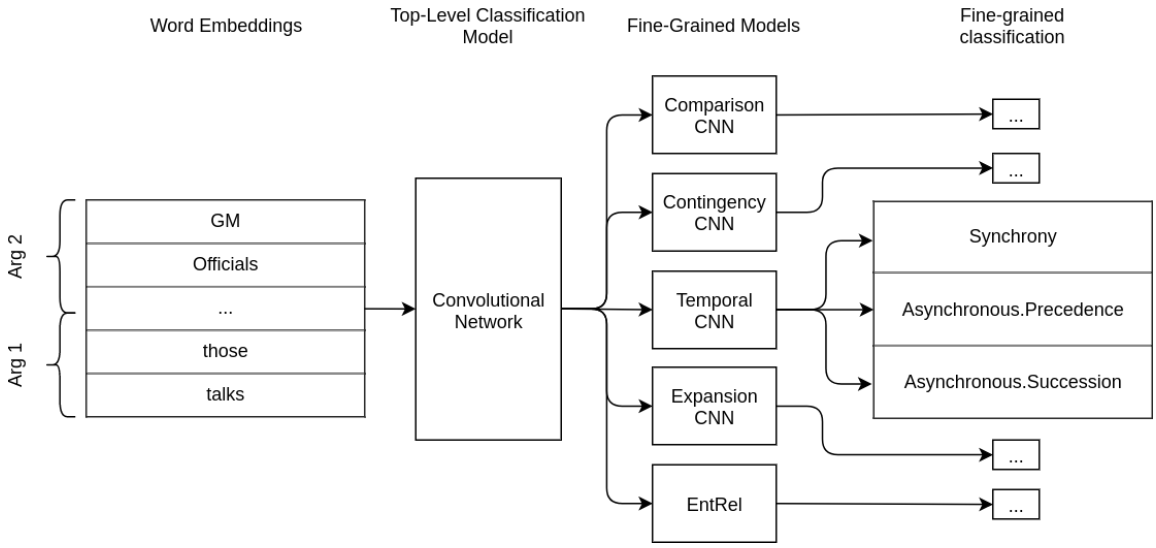


Figure 3.7: Hierarchical convolutional networks for fine-grained IDR.

TEMPORAL, EXPANSION or `EntRel`. The original discourse relation is then passed to the fine-grained model corresponding to the top-level model's most likely sense class.

Validation	**Test**	**blind**
22.19	13.14	11.98

Table 3.6: F1 score of hierarchical CNN.

Unfortunately, as we see in Table 3.6 the system's performance is quite low when compared to other systems in Table 3.3. We believe the fundamental problem is the fact we chose to pipeline the architecture into a top-level model and many fine-grained models. Since IDR models for top-level sense classification perform poorly compared to their `Explicit` counterparts (see Section 3.2), these errors cascade to lower systems, compounding their poor results. Additionally, since the lower level models train only within a given class, each model trains on much less data. We believe a better approach would share a representation across top-level and fine grained classification, which we discuss in Chapter 6.

Relation	Train	Validation	Test
COMPARISON	1988	90	134
CONTINGENCY	3660	131	218
EXPANSION	7259	280	428
TEMPORAL	773	39	19
EntRel	4133	215	217

Table 3.7: Top-level relations breakdown for Non-Explicit on the PDTB using the PDTB-split. The PDTB-split includes WSJ sections 2-21 for training, section 22 for validation and section 23 for testing. In contrast to the TOP-split in Table 2.2, Non-Explicit relations includes Implicit, Explicit and AltLex relations, whereas the TOP-split does not include AltLex. Additionally, EntRel is not merged into Implicit EXPANSION class.

3.5 Convolutional Neural Network with a Long Short-Term Memory Network

Building on the success of using convolutional neural networks for text classificaiton, we augment our network with a recurrent neural network (RNN) layer. The reasoning is to leverage the CNN's proven ability to extract meaningful features at the local level, and use the RNN to parse the extracted features as a sequence, i.e. time-dependent features. We hypothesize that a recurrent neural network would detect long-term dependencies which occur in Implicit discourse relation recognition. To illustrate this, consider the following discourse:

(Ex. 11) *Sierra has been instrumental in securing a number of the California bans.* **It has been waging an all-out campaign to beat back a proposal, pushed by Utah bike groups, to allow the cycles in federally designated wilderness areas**. (2034)

RNNs are naturally adapted to tasks in computational linguistics due to the recurrent network's recursive property. This property allows a network to parse inputs of various sizes, a natural fit for text. Hence, they are heavily used in language modeling. For example, in recent years RNN-based models have reached state-of-the-art perplexity for the Penn Treebank language modeling task. From 92.0 (Mikolov and Zweig, 2012), to 72.1 (Grave et al., 2017) and recent state-of-the-art of 52.8 (Merity et al., 2018), all with RNN-based models. We discuss in more depth RNNs in Chapter 4.

3.5.1 Model

Based on (Kim et al., 2016; Jozefowicz et al., 2016) we propose to augment our CNN model with a recurrent network. Recurrent networks are powerful tools to model sequences due to their relation to Markov chains in the sense that the network's next time step output is conditioned solely on its current time-step inputs. At each time step t, an RNN outputs a hidden state vector $h_t \in R^u$:

$$h_t = f(Wx_t + Uh_{t-1} + b) \tag{25}$$

The recurrent node mapping $h_{t-1}, x_t \rightarrow h_t$ is commonly referred to as an RNN *cell*, where u, the length of vector h_t, refers to the cell's number of hidden units. The exact cell mechanics varies depending on the RNN implementation. Since we will use an RNN to model the time component of our neural network, we must modify our previous convolutional neural network (see Section 3.3). We start with the concatenation of all convolutions from Equation 21:

$$r = [c_1||c_2||\dots||c_{n-h+1}] \tag{26}$$

This r operation is repeated j times, where j is the output size of the convolution, stacked into matrix R, where $R \in \mathbb{R}^{j \times l}$ and l is the number of convolutions. The convolution would normally be followed by a max-over-time pooling (see Section 3.3), however in this case the RNN models the time component. Hence we perform max-over-features pooling, extracting the most salient features *at each time step*.

$$p_j = \max_{i \in R} x_{ji},\ j = 1, \dots, l. \tag{27}$$

Now vector p is of length l, the time variable, as opposed to length j as in Equation 22, a result of selecting the maximum value in each column of R. Additionally, unlike in Equation 23, only a single convolutional kernel with width of size 3 is used. Once the CNN has extracted the most salient features across representation for each time step, we can pass on p to the RNN.

An RNN modeling sequential data in this fashion can in principle model data of any length. Unlike Markov states, a cell's hidden state acts like a memory which represents (theoretically) the

entire sequence $x_1, x_2, \ldots, x_t$ from first input to current time t into the single vector h_t. However, a serious issue with Equation 25 is the vanishing or exploding gradient problem which occurs during backpropagation (Hochreiter, 1991; Hochreiter et al., 2001) which limits the RNN's ability to learn long-term dependencies. This is problematic since the number of outputs produced by our CNN through time is approximately equal to the number of words in our sequences. As observed in Figure 3.2, discourse arguments can be as short as a few tokens and as long as more than 120 tokens.

To deal with the limitations of Equation 25, we employ Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) recurrent neural networks (LSTM). LSTM introduce *memory* vector $c_t \in R^u$, in addition to h_t, which purposely deals with long-term dependencies. Information flowing into c_t and removed by c_t is controlled by *gates*. At each time step, an LSTM cell produces vectors h_t and c_t as follows:

$$i_t = \sigma(W_i p_t + U_i h_{t-1} + b_i) \tag{28}$$

$$f_t = \sigma(W_f p_t + U_f h_{t-1} + b_f) \tag{29}$$

$$o_t = \sigma(W_o p_t + U_o h_{t-1} + b_o) \tag{30}$$

$$\tilde{c}_t = \tanh(W_c p_t + U_c h_{t-1} + b_c) \tag{31}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{32}$$

$$h_t = o_t \odot \tanh(c_t) \tag{33}$$

where σ is the element-wise sigmoid function, $\odot$ operator denotes element-wise multiplication, and p is the result of the pooling from Equation 27. The elements from p, the vector output from the pooling function, is the input to the recurrent neural network. Function i_t is a gate controlling the amount of information from the current input p_t and previous hidden state h_{t-1} flowing into the memory c_t. Function f_t, the forget gate, controls the amount of information to be removed from the previous memory c_{t-1}, again depending on current input p_t previous hidden state h_{t-1}. Function o_t, the output gate is used to produce the new hidden state h_t. Matrices $W_i, W_f, W_o, W_c, U_i, U_f, U_o, U_c$ and vectors b_i, b_f, b_o, b_c are parameters learned by the neural network. The architecture is detailed

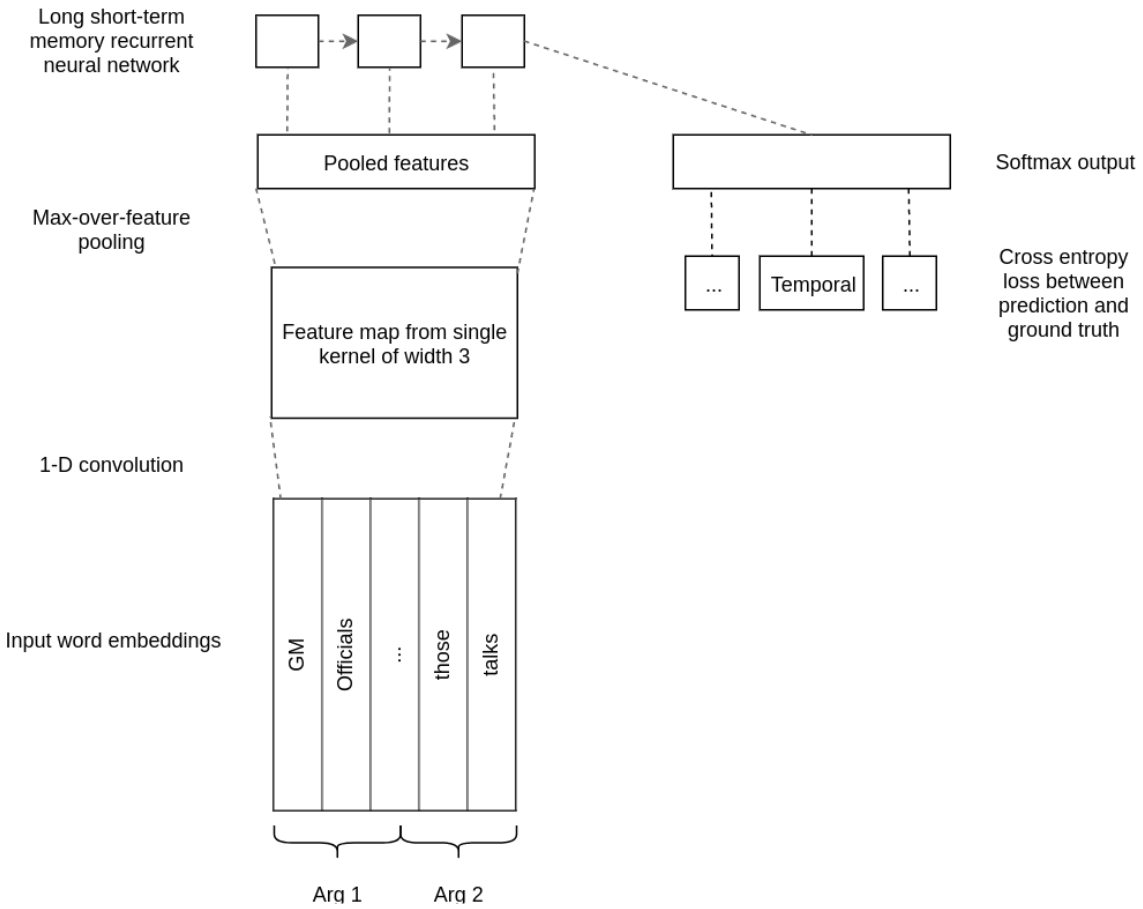


Figure 3.8: Our convolutional neural network with stacked long short-term memory network for discourse relation recognition.

in Figure 3.8. The network is trained in mini-batches of size 32 with the Adam algorithm (Kingma and Ba, 2015) just like the CNN in Section 3.3. The forget gate biases in the LSTM are initialized to 1.0 (Jozefowicz et al., 2015).

We test this setup in the context of fine-grained classification using the CoNLL-2016 dataset. Variations in several parameters are examined to see which would work best for this experimental setup. Given our previous finding in Section 3.3.3.2 that pretrained embeddings may not be beneficial, random embeddings are tested with various sizes and trained jointly with the network. Since a recurrent network parse the convolutional network's output, it is also assumed that large sized kernels are unnecessary. That assumption is tested with various kernel width.

Table 3.8 lists the network's basic hyperparameters common to all experiments unless specified (see Section 3.5.2).

Parameter	Value
window size	10
filters	100
activation	ReLU
batch size	32
embedding size	100
LSTM	128 units

Table 3.8: Base parameters used for training the CNN with LSTM model.

Dimensions	F1 score
10	0.2502
50	0.2527
100	0.2380
200	0.2405
300	0.2389

Table 3.9: Variation in embedding dimensions, F1-score on the validation set. Embeddings are randomly sampled from $\mathcal{N}(0, 0.5)$ and trained with the network.

3.5.2 Results and Analysis

The impact of various embedding sizes is listed in Table 3.9. It seems that for this particular network setup smaller embeddings around 50 dimension work best. This is a welcome result as smaller embedding dimensions reduce training time. This does come with the serious caveat that results in Table 3.9 are than those of Table 3.3. In this setting, lower embeddings perform better most likely since they reduce the number of parameters and therefore the chance of overfitting.

It would be wise to assume that without a recurrent network, wider kernels would perform well to map long-distance dependencies, while stacking a CNN with a recurrent network only requires smaller windows. Actually, evidence in Table 3.10 suggests that a kernel with a width of 10 performs better than any other size. Additionally, combining window sizes of 3, 10 and 20, acting as a kind of backoff to capture various-size dependencies, had no additional benefit. In all cases, combining two or more windows in a single run degraded performance (not shown).

In terms of pooling, max-pooling across features performed on average 3% better than average or min-pooling (not shown). This is not surprising as our other experiments confirm max-pooling works best on our dataset and convolutional models.

Dimensions	F1 score
3	0.2252
5	0.2525
10	0.2702
20	0.2583
30	0.2573
40	0.2485
50	0.2404

Table 3.10: Variation in kernel width, impact on F1-score of development set.

We also tested the network over mixed and split arguments. In the mixed case, the two arguments are concatenated as a single matrix and input into the network, as in Figure 3.8. In the split case, a convolution followed by a pooling in computed over `Arg1` then a separate convolution and pooling is computed over `Arg2`. The two pooled features are concatenated and fed into the recurrent neural network as shown in Figure 3.8. Doubling the convolution operations also adds an additional set of parameters, risking further overfitting. However, the split case actually improved performance (not shown). Unfortunately, given the network's low performance, the additional performance was only around 2%.

This last experiment suggests that stacking an CNN with an LSTM is not appropriate for discourse relation recognition. While it is theoretically possible to capture quite long-term dependencies with recurrent neural network, as does (Radford et al., 2017) in the case of language modelling, this comes with the cost of expanding the recurrent neural network's capacity. In our experiment, the LSTM has 128 hidden units, small when compared with the 4096 units in (Radford et al., 2017). Unfortunately, we cannot increase the LSTM's capacity since the CNN/LSTM network already overfits on the data. We would need more data to explore this avenue.

In this chapter we presented four types of convolutional neural network architectures. We discussed the importance of feature extraction across time by constraining each convolution over a narrow set of words, as well as the importance as separating convolutions and poolings between

each discourse argument. We presented a pipeline approach to IDR to take advantage of the hierarchical nature of discourse relation senses and the downside to such approach. We also presented a mixed architecture where a convolutional layer extracts low-level features which are then combined across time with a recurrent neural network. These findings, as well as our limited dataset, motivate a neural network explicitly built to exploit the pair-wise interaction between the two arguments, without the addition of a large number of parameters. We turn to such a solution in Chapter 4.

Chapter 4

Encoder Decoder Models for Implicit Discourse Relation Recognition

As we have seen in Chapter 3, key downsides to using convolutional neural networks for `Implicit` discourse relation recognition is that 1) CNNs are not flexible to varying input lengths and 2) they do not explicitly model sequential data. This latter point can be problematic given our preference for models with fewer parameters. A CNN with many layers could learn such sequential representation but would require much more data than available. In the CNNs used in Chapter 3, introducing any more capacity would quickly lead the models to overfit.

In this chapter we take the view that an `Implicit` discourse argument is generated not sequentially, but is conditioned on an entire previous sequence, i.e. on the first argument in an `Implicit` discourse relation. Consider for example:

(Ex. 12) *The National Institute of Health policy would require researchers to cut financial ties with health-care businesses – or lose their government money*. **Among other concerns researchers with business ties are more likely to falsify findings in order to tout new drugs**. (0975)

The `Implicit` discourse relation in (Ex. 12) is labelled with sense CONTINGENCY:Cause:reason. The entire `Arg1` (in bold) is the reason for `Arg2` (in italics). Given `Arg1`, several sequences of words could be used as argument 2. However, the possibilities are restricted given the discourse relation. We believe that a model that explicitly learns the probability of the entire sequence of

argument 2 conditioned on argument 1 should also implicitly encode the discourse relation as a latent variable, which we can recover. In order to do so, we explored the use of recurrent neural networks and sequence-to-sequence models to develop our approach to `Implicit` discourse relation recognition.

4.1 Encoder-Decoder RNNs for NLP

Encoder-decoders are a fairly recent trend in deep learning for NLP. They were introduced independently as RNN encoder-decoder (Cho et al., 2014) and sequence-to-sequence (Sutskever et al., 2014) for machine translation. This family of encoder-decoder models are commonly referred to as seq2seq, irrespective of their specific implementation. A seq2seq model is interpreted as learning the conditional distribution:

$$p(y_1, y_2, \ldots, y_{T'} | x_1, x_2, \ldots, x_T) \tag{34}$$

Where y corresponds to an output sequence conditioned on the input sequence x, and where sequence length T' and T are not necessarily identical. A seq2seq model first encodes the entire variable x input with an RNN into a fixed size vector c. Then, a separate RNN generates output $y_1, \ldots, y_T$ conditioned on previous predictions and context vector c:

$$p(y_1, y_2, \ldots, y_{T'} | x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | \{y_1, y_2, \ldots, y_{t-1}\}, c) \tag{35}$$

The implementation by (Sutskever et al., 2014) conditions y on c only at the first output, that is y_1, whereas (Cho et al., 2014) conditions every y_t on the same context vector c, closely approximating Equation 35, thus forming the basis to our model. Modifying Equation 7, and letting s_t correspond to the target sequence's hidden states, s_t is calculated as:

$$s_t = f_h(W_s x_t^d + U_s s_{t-1} + Cc + b_s) \tag{36}$$

where C is a parameter matrix and c is the context vector. Note that Equation 36 differs

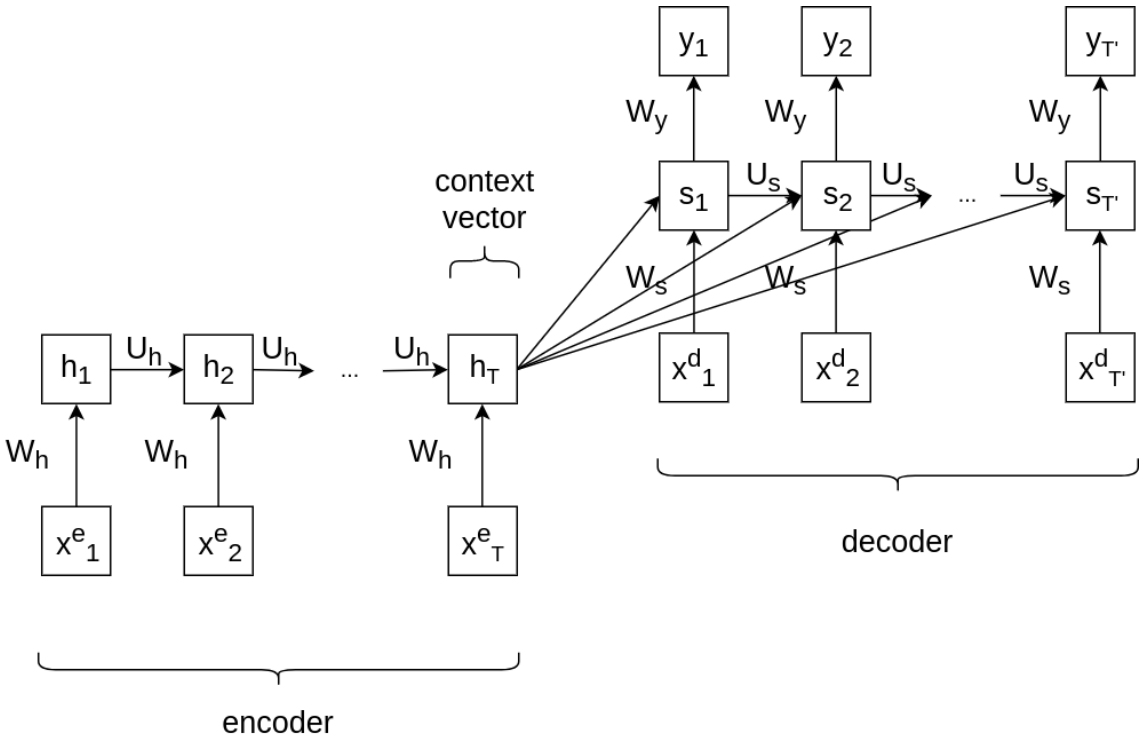


Figure 4.1: An encoder-decoder recurrent neural network. x_t^e and h_t respectively represent the encoder input and hidden state, x_t^d, s_t and y_t respectively represent the decoder input, hidden state and output prediction. The last encoded vector state, h_T is used as context vector c. Unlike the encoder hidden states, each decoder hidden state s_t are conditioned on the context vector c in addition to the previous hidden state s_{t-1} and the current input x_t^d.

from (Cho et al., 2014). Our formulation assumes a Vanilla RNN as opposed to a Gated Recurrent Unit (GRU) to simplify the notation.

By modifying the RNN in Figure 2.5 with the encoding and decoding features, we obrain the network shown in Figure 4.1.

Additionally, (Sutskever et al., 2014) found that reversing the input sequence helped the decoder make better predictions (more accurate y_t). Based on this finding, as well as those of (Bahdanau et al., 2015) and earlier work on bidirectional RNNs (Schuster and Paliwal, 1997; Graves and Schmidhuber, 2005), we experiment with encoding the input in both directions (see Section 4.2).

The downside of using standard encoder-decoder models as presented thus far is similar to using other RNNs, that is their relatively poor performance on longer sequences. (Bahdanau et al., 2015) show how in neural machine translation seq2seq models gradually degrade in performance beginning with sequences of 20 tokens or more. Discourse arguments are typically between 20 and

60 tokens, as shown in Figure 3.2, which motivates the need for models which deal well with longer sequences. Additionally, we assume in the context of discourse relation recognition that there are key words in one argument that are linked with words in the other argument. We hypothesize that a neural network model would benefit by explicitly learning these links across arguments. This brings us to the use of attention mechanisms, discussed in Section 4.2.

4.2 Encoder-Decoder with Attention

4.2.1 Attention Mechanism

In Section 4.1 we gave an overview of the standard encoder-decoder RNN model. We now discuss our encoder-decoder model augmented with an attention mechanism (Bahdanau et al., 2015). The encoder encodes an input x, where x is represented as a sequence of word embedding vectors, into a single context vector $c = q(h_1, \ldots, h_{T_x})$ and a hidden state $h_t = f(x_t, h_{t-1})$. Functions f and q are nonlinearities, in our case Bidirectional RNN (Schuster and Paliwal, 1997).

As shown in Equation 35, typically the decoder predicts a sequence of words y_i, where each y_i prediction is conditioned on past predictions $\{y_1, \ldots, y_{i-1}\}$ and the context vector c, maximizing the following joint probability:

$$p(y) = \prod_{i=1}^{T'} p(y_i | \{y_1, \ldots, y_{i-1}\}, c) \tag{37}$$

In the context of RNNs, the conditional probability of each y_t in the joint probability of Equation 37 is modeled as a nonlinear function g with input y_{i-1}, context vector c and hidden state s_t:

$$p(y_t | \{y_1, \ldots, y_{i-1}\}, c) = g(y_{i-1}, s_i, c) \tag{38}$$

(Bahdanau et al., 2015) propose the use of a unique context vector c_i for each decoding time step, redefining the decoder conditional probability for each word y_i as:

$$p(y_i | \{y_1, \ldots, y_{i-1}\}, \mathbf{x}) = g(y_{i-1}, s_i, c_i) \tag{39}$$

where the context vector c_i is a weighted sum over all input hidden states $(h_1, \ldots, h_T)$:

$$c_i = \sum_{j=1}^{T} \alpha_{ij} h_j \tag{40}$$

where a weight a_{ij} is calculated as:

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{41}$$

$$e_{ij} = a(s_{i-1}, h_j) \tag{42}$$

where scalar function a is a pair-wise scoring function, scoring the relation between the decoder hidden state s_{i-1} and encoder hidden state h_j. Learning of the scoring is done by a simple feedforward neural network. In our case, this means that the discourse argument 1 has T tokens, for each decoder hidden state s_i the network computes T scoring terms e.

Using attention leads to a vectorized representation of the second discourse argument (the decoder hidden states) which is not only informed of its context but also of its alignment with `Arg1`.

4.2.2 Recurrent Neural Network Cell

In addition to exploring the use of an attention mechanism, we also used the gated recurrent unit (GRU) (Cho et al., 2014) as the nonlinear recurrent function. This function has fewer parameters than long short-term memory (LSTM) networks while performing just a well (Cho et al., 2014; Chung et al., 2014). In Section 3.5.1, we introduced the equations that compose the LSTM. Recall that an LSTM is composed of four gating functions that control the amount of information flowing into the memory unit and the information to be forgotten from the cell at the next time step. In the GRU, only two gating functions are used. As discussed in Chapter 3, minimizing the number of parameters is a serious concern for our neural networks. Hence, the use of a GRU seems promising.

The encoder RNN with a GRU cell is computed with the following three functions:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{43}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{44}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{45}$$

where x_t is the input vector, h_t the new hidden state vector, W, U are learned parameters matrices and b are learned bias vectors. In the GRU, z_t is known as the *update* gate and r_t as the *reset* gate. With this formulation, hidden units tend to specialize into capturing either longer term or shorter term dependencies. Long-term dependency units tend to have mostly active update gates, while short-term units tend to have active reset gates (Cho et al., 2014).

The decoder cell has the same setup but is augmented with the context vector and associated parameter matrices C:

$$z_i = \sigma(W_z x_i + U_z h_{i-1} + C_z c_i + b_z) \tag{46}$$

$$r_i = \sigma(W_r x_i + U_r h_{i-1} + C_r c_i + b_r) \tag{47}$$

$$s_i = z_i \odot s_{i-1} + (1 - z_i) \odot \tanh(W_h x_i + U_h(r_i \odot h_{i-1}) + C_h c_i + b_h) \tag{48}$$

4.2.3 Augmenting the Encoder-Decoder Model for Classification

We now describe how we augment the previous seq2seq with attention model in Section 4.2.1 for classification. To do IDR recognition we used an encoder-decoder RNN with attention. However, given our classification task and since the decoder inputs (the words of discourse argument 2) are known and not to be predicted, the model is not trained by maximizing the likelihood of the decoder targets, as in Equation 37, but rather by minimizing the cross-entropy error between the predicted label $\hat{y}$ and the true label y for all possible labels l:

$$E(y, \hat{y}) = -\sum_{i=1}^{l} y_i \log(\hat{y}_i) \tag{49}$$

We experimented with two classifiers to predict $\hat{y}$.

4.2.3.1 Classifier with Attention (CA)

In the simplest case, we make the assumption that as the decoder steps through argument 2, it decodes the latent discourse relation representation which is conditioned on the previous time step hidden state and certain words from `Arg1` which align with `Arg2` to signal a discourse relation. Once decoding is finished, the final hidden state should fully reflect the decoded discourse relation. We can view the decoding process as the model slowly converging towards a discourse relation label. Hence, we use the final decoder hidden state $s_{T'}$ as the input to our classification layer:

$$\hat{y} = f(W_o s_{T'} + b) \tag{50}$$

where f is the softmax function over the final decoder hidden state $s_{T'}$ of size d, $W_o \in \mathbb{R}^{l \times d}$ is a parameter matrix and $b \in \mathbb{R}^l$ is a bias vector. In this setup the classifier only relies on the last hidden state, minimizing the total number of parameters at the expense of information loss. We name this model *classifier with attention* (CA). The architecture for this model can be visualized in Figure 4.2.

We predict a potential issue with this type of setup. Recall from Section 4.1 that encoder-decoders have difficulty handling long sequences as it is too hard a task for a recurrent network to fully express a sequence into a single final vector. Attention mechanisms work well in machine translation since the decoder predicts an output at every time step, alleviating the need to remember the entire decoded output (Bahdanau et al., 2015). However, in our case we are basically requiring the final decoded output to fully capture the discourse relation latent variable from all of `Arg1` and `Arg2` into a single vector. To alleviate this potential issue, we experimented with a second classifier: CSA.

4.2.4 Classifier from Sequence with Attention (CSA)

Since the decoder hidden states capture the interactions between the tokens of `Arg2` and `Arg1` tokens (via attention) as well as the decoded discourse sense (its hidden states), we propose a classification layer that uses all decoded hidden states as input. This new architecture, that we name

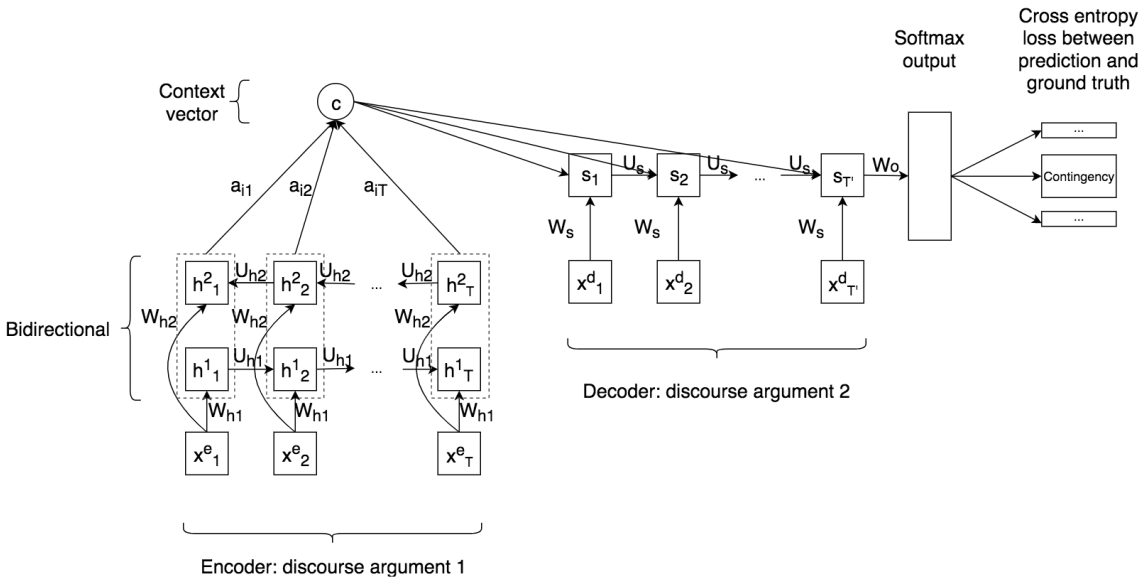


Figure 4.2: Our classifier with attention (CA): an encoder-decoder recurrent neural network with attention with the last hidden state used for classification. In the doted rectangles, the forward and backward hidden states are concatenated. Note there is no backpropagation through time from output predictions at each time step. Only the final cross-entropy error is backpropagated through time.

classifier from sequence with attention (CSA), is shown in Figure 4.3. The CSA's $\hat{y}$ is a function of:

$$p = \text{max\_pool}_{i=1}^{T'}(s_1, \ldots, s_i) \tag{51}$$

$$h = g(W_d p + b_d) \tag{52}$$

$$\hat{y} = f(W_s h + b_s) \tag{53}$$

where p is a T' sized concatenated vector of the maximum values over each decoder hidden state s_i, i.e. 1D max pooling. $W_d \in \mathbb{R}^{v \times T}$ and $Ws \in \mathbb{R}^{l \times v}$ are parameter matrices, $b \in \mathbb{R}^v$ and $b \in \mathbb{R}^l$ are bias terms, and g a nonlinearity. In this case each decoded time step informs the relation classification, as seen in Figure 4.3.

4.3 Experiments

In this section we explain our data preprocessing and experiments. As with the experiments of Chapter 3, the raw texts from the PDTB (Prasad et al., 2008) and the CoNLL SDP (Xue et al., 2016)

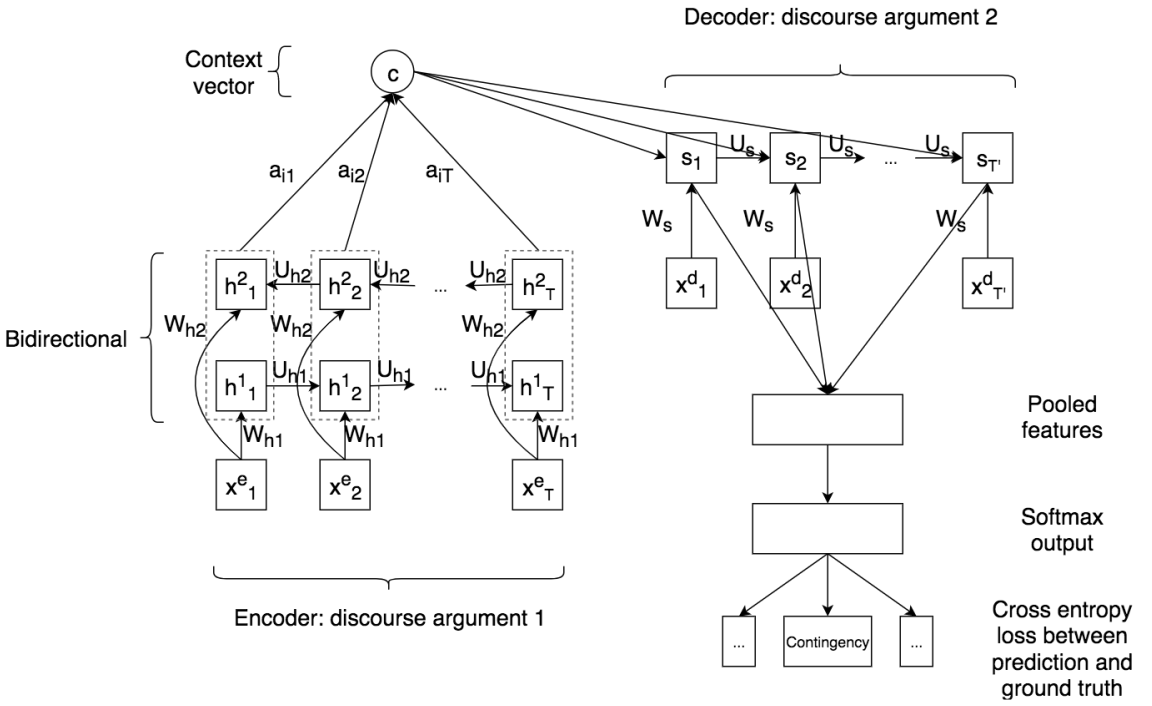


Figure 4.3: Our classifier with sequence of attention (CSA): encoder-decoder recurrent neural network with attention. The decoder hidden states are used for classification. Note that there is no backpropagation through time from output predictions at each time step.

are converted to lower case and tokenized. Then we keep only the 10,000 most common words. After forming a dictionary of unique tokens, we substitute each token with a dense word embedding from a pretrained model. Following the preferred embeddings used at the 2016 CoNLL SDP, we used the pretrained Word2Vec binaries[1], pretrained by (Mikolov et al., 2013b) for both top-level and fine-grained classification. While the PDTB samples contain additional data such as part-of-speech tags and parse trees, this additional information was not used.

The top-level classification consists of four separately trained binary classifiers, while we train a single classifier for the fine-grained classification. We experimented with the use of LSTM and GRU (Cho et al., 2014) cells, opting for the GRU since it showed slightly better results during development. The number of cell parameters were randomly searched at each training run. For the CSA, we additionally performed hyper-parameter search on the number of hidden units. Both models were optimized by Stochastic Gradient Descent with the Adam algorithm (Kingma and Ba, 2015) with mini-batches of size 32. Models evaluated on the test sets are based on optimal validation

[1]https://code.google.com/archive/p/word2vec/

Model		Parameter	Value
CSA	**CA**	batch size	32
		embedding size	300
		cell type	GRU
		cell units	100
		pooling	1D max
		dense layer units	60

Table 4.1: Architecture parameters. Dense layer refers to the CSA model's fully connected layer between pooling and softmax layers.

ID	Author	Blind	Test	Dev
ecnucs	(Wang and Lan, 2016)	34.18	40.91	46.40
tbmihaylov	(Mihaylov and Frank, 2016)	34.51	39.19	40.32
tao0920	(Qin et al., 2016)	35.38	38.20	46.33
gtnlp	-	36.75	34.95	40.72
ttr	(Rutherford and Xue, 2016)	**37.67**	36.13	40.32
CSA	ours	35.07	28.05	36.58
CA	ours	**38.25**	35.63	39.42

Table 4.2: F1-scores of our CSA and CA models for fine-grained IDR compared to the top 5 teams at CoNLL-2016.

set F1 score. Our main parameters that produced the best performance are listed in Table 4.1.

4.4 Results and Analysis

Table 4.2 summarizes our fine-grained classification results[2] on the CoNLL SDP dataset and Table 4.3, our top-level classification results on the PDTB dataset in comparison with other authors.

As shown in Table 4.2, our CA model scored 38.25% on fine-grained classification, higher than the state-of-the-art F1 score of 37.67%. Observing the blind test set results in Table 4.2 we note that our model seems to generalize well to an unseen and different dataset (Wikinews). Other top models from Figure 4.2 such as *gtnlp* and *ecnucs* (both CNNs), have a more than 10 point difference between development score and blind test score compared to only 2 points in the CA case.

On top-level classification, only our CA model (see Table 4.3) scored well in the case of EXPANSION and CONTINGENCY, the largest relation classes. Classes CONTINGENCY and TEMPORAL were better than most other approaches. The F1-score of 30.56% for COMPARISON was far from the top result in Table 4.3, likely due to the small dataset size. Our model overfits on the

[2]We used the official CoNLL-2016 scorer: https://github.com/attapol/conll16st

Author	COMPARISON	CONTINGENCY	EXPANSION	TEMPORAL
(Pitler et al., 2009)	21.96	47.13	76.42	16.76
(Zhou et al., 2010)	31.79	47.16	70.11	20.30
(Park and Cardie, 2012)	31.32	49.82	79.22	26.57
(Rutherford and Xue, 2014)	39.70	54.42	80.44	28.69
(Ji and Eisenstein, 2015)	35.93	52.78	80.02	27.63
(Chen et al., 2016a)	**40.17**	54.76	80.62	**31.32**
CSA	27.02	49.86	77.45	24.43
CA	30.56	**54.80**	**80.72**	27.15

Table 4.3: F1-scores of our CSA and CA models for top-level IDR. Note `Entrel` is merged into EXPANSION as is done in the above listed works.

smaller datasets when compared with other systems listed in Figure 4.2 possibly due to a higher parameter count from the bidirectional RNN and attention parameters. However, this is difficult to verify since we do not have access to these other models.

Our CA model is relatively shallow, a single bidirectional RNN encoder layer and a single RNN decoder layer with attention. Hence, it is possible that the chosen input embedding had a minor impact on our results. We would have liked to measure the embedding effect to compare with (Chen et al., 2016a), but to our knowledge their embedding is not publicly available.

We were surprised by the CSA's lower performance in all cases. We believed the model would be more robust if the classification layer had inputs from all decoded hidden states directly. However, using only the final state vector resulted in higher classification score while using less parameters. This may be due to overfitting. As future work (see Chapter 6), we would need to reevaluate it on a much larger dataset.

In this chapter, we presented an efficient encoder-decoder model with attention for `Implicit` discourse relation recognition, a model which fully exploits the sequential nature of the data, to appear in (Cianflone and Kosseim, 2018). Our model computes attention between discourse argument word pairs without feature engineering and without the need for additional neural network layers, possibly minimizing the number of trainable parameters. We show that our model generalizes well to unseen datasets on fine-grained classification, performing comparatively with state-of-the-art, without large variance in scoring between development and test sets. In the current chapter we

hypothesized that argument 2 is generated by conditioning on argument 1 and a latent discourse relation which we tried to recover. In the next chapter we test this hypothesis in the unsupervised setting where large language models may automatically learn discourse relations.

Chapter 5

Disentangled Representation of Discourse Relation

5.1 Background

Learning a representation with unsupervised learning has a rich history in neural networks and plays a critical role in their recent success (Bengio et al., 2013). Good representations are commonly reused across tasks, such as in computer vision (Oquab et al., 2014) and NLP (Collobert et al., 2011), a form of transfer learning.

To fight the curse of dimensionality (Bengio et al., 2003), learning a *distributed* word representation (Hinton et al., 1986) has been shown to be an essential tool of natural language processing with neural networks (Bengio et al., 2003). Words with similar meaning or similar usage, are expected to have similar distributed feature vectors, where similarity is measured with a vector distance function such as Euclidean distance. (Bengio et al., 2003) first showed that these representations can be learned on vast quantities of data in an unsupervised way.

An excellent example by (Mikolov et al., 2013b) shows an interesting degree of compositionality in the word vectors learned by their Word2Vec training method. For example, the addition of the word vectors "Germany" and "capital" result in a vector that is close to the vector for the word "Berlin". Similar analogies have also been shown for images after training a deep convolutional generative adversarial network (DCGAN) (Radford et al., 2016). The authors showed, for example,

that if you take the mean noise vector of images of men with glasses, followed by a subtraction of the mean vector for men without glasses, and finally add the mean vector of women without glasses, this results in a noise vector for women with glasses. By inputting this vector into the DCGAN's generator, the model generates a number of highly plausible images of women with glasses.

A key strength of these approaches is that they are trained in a completely unsupervised way. The learning of a good representation results in *how* the models are trained. (Mikolov et al., 2013b)'s continuous bag-of-words training consists of a neural network which predicts a *middle* word given a context window, such as the 2 preceding and the 2 following words. Skip-thought vectors (Kiros et al., 2015) extend this idea to larger text by encoding a sentence and predicting the surrounding sentences. For images, (Radford et al., 2016)'s learn representation results from training a new type of generative adversarial network (GAN) (Goodfellow et al., 2014). GANs are trained by having two networks play a kind of game where one network generates fake images (the generator) while another network predicts if a generated image is real or fake (the discriminator).

The downside to these methods is that the learned factors of variation are distributed across the learned representation. For example, there is no single neuron that turns glasses on or off in an image. The glasses factor is shared across the latent space. To isolate that factor, one must first observe images with glasses and rely on vector arithmetic.

(Bengio et al., 2013) argue that for an AI to understand the world it "can only be achieved if it can learn to identify and disentangle the underlying explanatory factor". By "disentangled", we mean a representation which learns to separate the various explanatory factors. For example, a representation that disentangles shadows from objects would have a region in latent space specialized in the representation of shadows, and a different spaces for other factors. (Kingma and Welling, 2014) showed that a Variational AutoEncoder (VAE) trained on Frey faces with only a two-dimensional latent space, learned to disentangle facial expression in one dimension, and object rotation in the other dimension. A more recent model, InfoGAN (Chen et al., 2016b), proposes a new type of GAN with the explicit goal of disentangling representations. This is achieved by augmenting the GAN objective with an auxiliary loss. The new objective consists of maximizing the mutual information between a new latent "code" and the generator's output. They show this latent code learns to disentangle factors much like VAEs (Kingma and Welling, 2014) but in the GAN setup.

5.2 Recurrent Neural Network for Disentanglement

Recently (Radford et al., 2017) showed that a high-capacity character-level language model trained on large corpora can learn disentangled representation. They trained a recurrent language model, a multiplicative LSTM (mLSTM) (Krause et al., 2017) in this case, on the Amazon product review dataset (McAuley et al., 2015) and showed that such a model automatically learns sentiment, represented in mostly a single neuron of the mLSTM's possible 4096 neurons. Based on these findings, we hypothesize that a neural network trained on a large amount of data could learn disentangled representation of discourse relations. We first describe the dataset we used in Section 5.2.1, our model in Section 5.2.2, and give details of our experiment and results in Sections 5.2.3 and 5.2.4.

5.2.1 Dataset

The mLSTM model trained by (Radford et al., 2017) on the Amazon reviews dataset took one month to train, parallelized across 4 Pascal Titan X GPUs. Given our computational limits, we could not train such a high-capacity model on such a large corpus and therefore opted to reuse the mLSTM parameters from (Radford et al., 2017). The Amazon product review datasets consists of 82 million product reviews from 1996 to 2014, with over 38 billion characters.

The mLSTM is used to extract a high quality representation of our data on which we want to predict discourse relations (see Section 5.2.3 for details of the experiment). The data we use for discourse relations is the PDTB as described in Chapter 2. Each sample consists of `Arg1`, `Arg2` and possibly a discourse connective in the case of `Explicit` relations.

5.2.2 Model

We use a character-level recurrent neural network, mLSTM in our case, to encode the characters of our data. Since the model is character based, preprocessing is minimal. We first convert our dataset to integer form, swapping the characters for their ASCII encoding. For each mini-batch, these encodings are swapped for dense embeddings of size 64. For details on integerization and embedding, see Chapter 3.

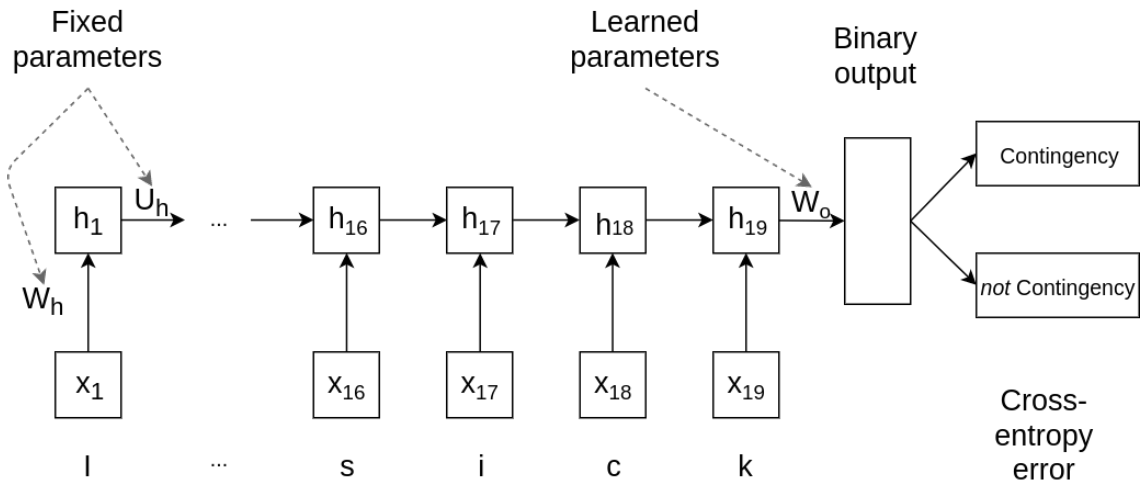


Figure 5.1: RNN using pretrained weights. At each time-step, a discourse character is encoded by the RNN, conditioned on past hidden states until the final character is encoded. With parameter matrix W_o, we predict perform binary classification. Note W_o is trained by logistic regression with lasso regularization, while all other parameters remain fixed.

The embedded sample forms a sequence of embedded characters $x_1, x_2, \ldots, x_T$ where T corresponds to the number of characters in the sample. At each time step t, character x_t is input into the network's non-linear function $f()$, the cell, which outputs a new memory vector c_t and hidden state vector h_t :

$$c_t, h_t = f(x_t, h_{t-1}, c_{t-1}) \tag{54}$$

Figure 5.1 shows an abstracted view of the flow of information through the RNN. Since we are only concerned with h_t, we omit c_t and all other mLSTM parameters. Once the entire discourse, including `Arg1` and `Arg2` is fed into the RNN, we save only the last hidden state h_T. We then perform binary classification by predicting the probability of the discourse belonging to a class or not:

$$\hat{y} = \text{softmax}(W_o h_T + b_o) \tag{55}$$

Where W_o is the learned matrix mapping h_T to the vector of length two. During training, only W_o is updated. All input-to-cell parameters and all internal cell parameters are fixed during training.

Relation	All	Explicit	Implicit
COMPARISON	11388 / 2438 / 2438	7826 / 1676 / 1676	3498 / 748 / 748
CONTINGENCY	11570 / 2478 / 2478	5238 / 1122 / 1122	5946 / 1274 / 1274
EXPANSION	21532 / 4614 / 4614	8994 / 1926 / 1926	12236 / 2622 / 2622
TEMPORAL	6586 / 1410 / 1410	5176 / 1108 / 1108	1290 / 276 / 276
EntRel	7296 / 1562 / 1562	-	7296 / 1562 / 1562

Table 5.1: Dataset distribution for training/validation/testing in 3 scenarios: 1) Explicit, Implicit and EntRel combined, 2) Explicit only, 3) Implicit and EntRel only. Note the numbers here combine positive and negative samples. In all cases, positive and negative samples are split evenly.

5.2.3 Experiments

To detect if a discourse relation is learned by the language model, we perform a series of experiments on each discourse relation sense. Since we are not concerned with classifying discourse relation senses in a natural distribution, we do not use the common PDTB breakdown as we did in Chapters 3 and 4 as the recommended breakdown is extremely unbalanced (see Table 2.2). We opt for a balanced binary positive/negative dataset as in (Radford et al., 2017). For each discourse sense, the positive class consists of all senses belonging to a same class, whereas the negative senses are randomly sampled from the rest of the dataset. 15% of the dataset is held out for validation and 15% for testing. All sections of the PDTB are used. Details of the breakdown are shown in Table 5.1.

In Section 5.2.2 we explained how we obtained the discourse representation. The experiments consists of classifying each discourse, using the representation, as belonging to a discourse relation sense or not. We hypothesize that a performing model trained with the described setup would have disentangle the representations, which we could observe. Hence it is important to perform a single binary classification for each discourse sense to isolate the location of this representation. Additionally, we believe that the representation may not necessarily be the same for Explicit, Implicit and EntRel relations. We therefore perform the classification in 3 scenarios: 1) Explicit, Implicit and EntRel combined as category *All*, 2) Explicit only, 3) Implicit and EntRel only. Since EntRel is both an annotation type and sense, we also consider it on its own. This breakdown is shown in Table 5.1.

The goal of these binary classifications is not to actually classify discourse relations, but to discover where and if discourse relations are disentangled. The key is the weight matrix W_o in

Figure 5.1. For each classification, we perform logistic regression with the lasso, the "least absolute shrinkage and selection operator", which we choose specifically as it regularizes the model by shrinking some weights and setting others to zero (Tibshirani, 1996). Weights in W_o which are not zero and strongly linked to h_T neurons are clues as to disentangled features.

It is however difficult to determine with certainty if a learned representation is disentangled. Results are not always as clear as in (Radford et al., 2017) where using a single hidden state unit helped achieve accuracy of 92.30%, close to state-of-the-art of 94.09%, in sentiment recognition. Additionally, the use of all 4096 hidden state units marginally increased accuracy to 92.88%.

To the best of our knowledge, there is no objective criteria to determining "disentanglement". We can propose 3 criteria to determining disentanglement: accuracy, dataset size and number of hidden units strongly signaled. At the moment, results are interpreted simply in relation to one another.

5.2.4 Results and Analysis

From Table 5.2 we can see that in all cases accuracy is greater than the 50% baseline. As we mentioned in Section 5.2.3, we cannot make a clear statement of disentanglement, but we can compare the results across class.

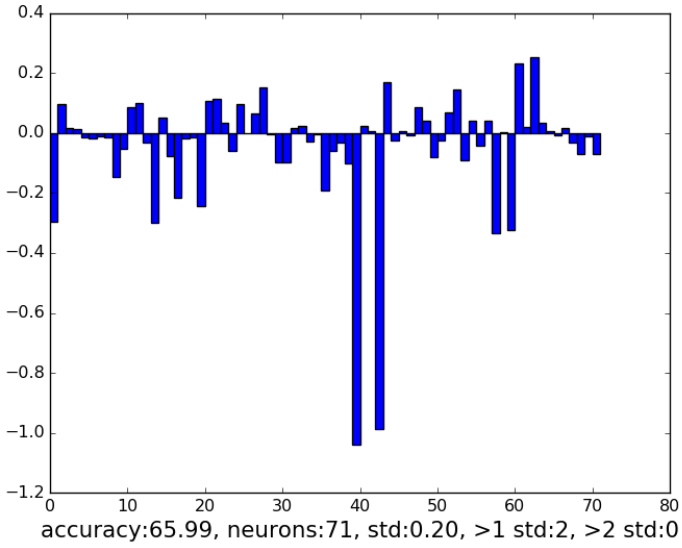
One important observation to make is the negative correlation between the number of neurons and classification accuracy, as observed in Table 5.3. In the *All* case, a dataset which includes `Explicit`, `Implicit` and `EntRel` relations, the Pearson correlation is -0.65. On their own, `Explicit` and `Implicit` are quite differently behaved, with `Explicit` relations classification accuracy much less related to the number of used neurons. In other words, while the language model does automatically learn `Explicit` relations better than `Implicit` relations, it does so in a less systematic fashion in terms of disentanglement. It is more case by case, depending on the discourse relation. For example, `Explicit` COMPARISON achieves an accuracy of 65.99% while only 57.49% in the `Implicit` COMPARISON case. And it does so with much less neurons, needing only 71 neurons compared to `Explicit`'s 278, out of a total of 4096. We can also observe this pattern visually in Figure 5.2 (a) and (b). Notice the noisy `Implicit` chart compared to the less active `Explicit` chart except for 2 significant weights, indicative of disentanglement.

Sense	Annotation type	Accuracy	#Neurons	σ	$> 1\sigma$	$> 2\sigma$
COMPARISON	All	62.14	305	0.11	16	3
	Explicit	65.99	71	0.20	2	0
	Implicit	57.49	278	0.11	32	5
CONTINGENCY	All	63.72	100	0.12	8	3
	Explicit	66.13	149	0.12	14	3
	Implicit	67.90	57	0.18	3	0
EXPANSION	All	58.32	1270	0.09	139	32
	Explicit	61.63	241	0.10	20	7
	Implicit	57.09	861	0.10	106	15
TEMPORAL	All	74.33	181	0.19	12	8
	Explicit	76.90	157	0.18	17	7
	Implicit	63.41	237	0.24	29	9
EntRel	EntRel	74.39	186	0.22	13	5

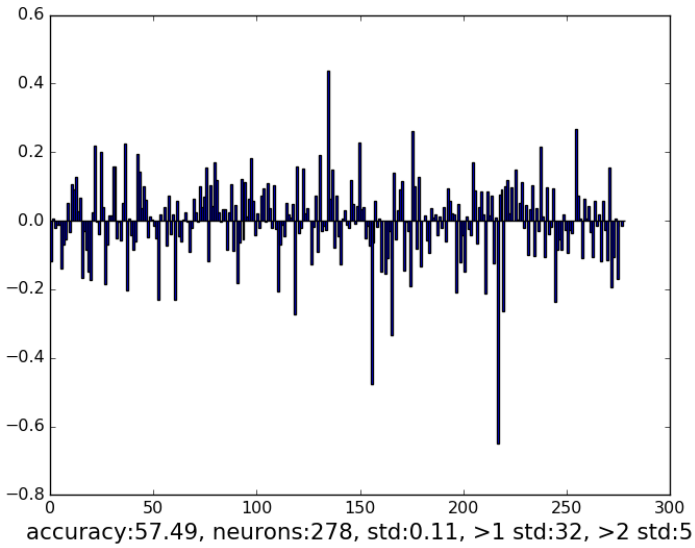
Table 5.2: Top-level sense classification results on balanced dataset. For each sense class, we show the results for the various datasets: 1) All annotation types including Explicit, Implicit and EntRel which form category *All*, 2) only Explicit, and 3) only Implicit. EntRel is also considered on its own as both a sense and annotation type. Accuracy is on the balanced binary dataset. The column *neurons* represents the non-zero coefficient resulting from the lasso regularization. It can be interpreted as the number of features (neurons) necessary for the classification. The σ column represents the standard deviation of the logistic regression weights. The last two columns represents the number of weights greater than one or two standard deviations.

	Correlation
All	-0.6352
Explicit	-0.2586
Implicit	-0.7501

Table 5.3: Pearson correlation between the number of used neurons by the logistic regression (non-zero weights) and classification accuracy.

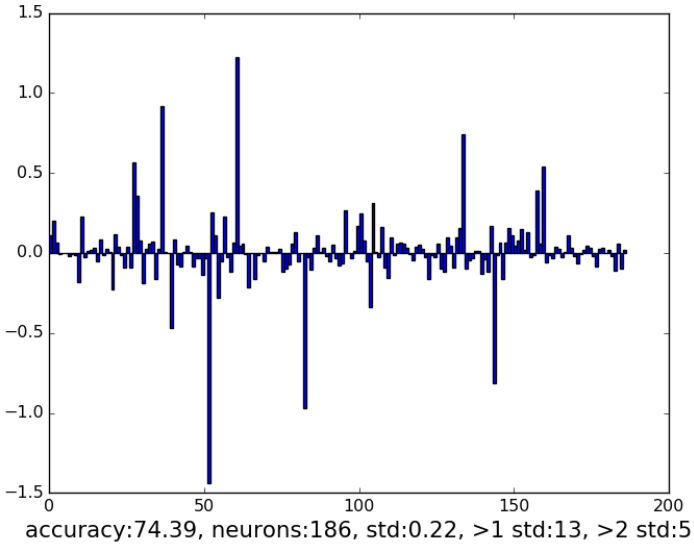


(a) active weights for `Explicit` COMPARISON

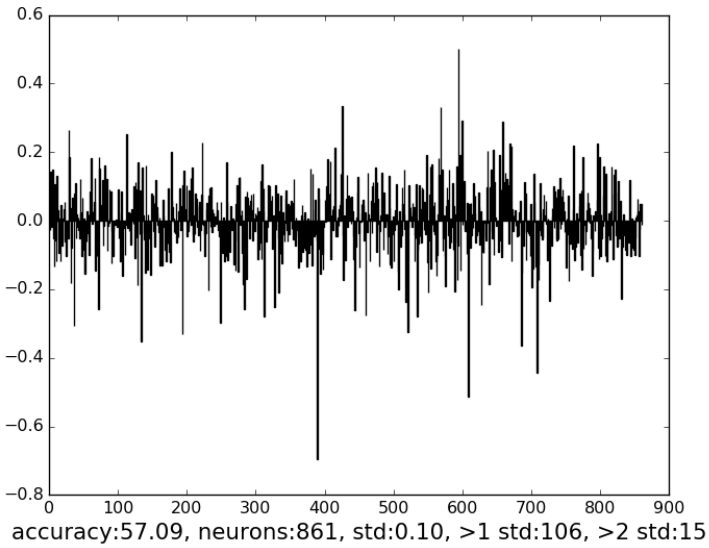


(b) active weights for `Implicit` COMPARISON

Figure 5.2: Visualization of neuron weights (the regression coefficients). Due to the lasso regularization, in most cases only a small proportion of the 4096 neuron features are needed to for the classification task. We only show the active weights leading to different x-axis sizes.



(a) active weights for `EntRel`



(b) active weights for `Implicit` EXPANSION

Figure 5.3: Visualization of neuron weights (the regression coefficients). Due to the lasso regularization, in most cases only a small proportion of the 4096 neuron features are needed to for the classification task. We only show the active weights leading to different x-axis sizes.

We can hypothesize that this is due to the nature of Explicit relations where discourse relations are highly correlated with discourse connectives. Given the large number of discourse connectives (more than 100), the discourse connective representation is scattered in a narrower representation space, by being signaled by key words, compared to Implicit relation where representation is not disentangled. We believe this is why Explicit TEMPORAL, at 76.90% accuracy, performs better than other classes. The Explicit TEMPORAL sense is more predictable given its lower number of discourse connectives (Prasad et al., 2008). Similarly, EntRel accuracy is quite high. As discussed in Chapter 2, EntRel refers to relations between entities across discourse arguments. Occasionally entity mentions will repeat, as in (Ex. 6) of Chapter 2, which makes the pattern more predictable.

There is an interesting exception to our previous observations, the case of Implicit CONTINGENCY. It is the only discourse sense where Implicit accuracy, at 67.90% is higher than its Explicit counterpart, at 66.13%. It also has the highest Implicit discourse sense accuracy in Table 5.2. And even more surprising, it is the only Implicit sense where fewer neuron features are active from the regularized logistic regression in comparison with its Explicit counterpart. The classification needs only 57 features, with 3 significant features at greater than one standard deviation from mean weight value, compared with 149 for the Explicit case. 57 is also the lowest number of used features compared to all other classification results in Table 5.2. The likely explanation is that we used parameters from (Radford et al., 2017) whose model strongly captures sentiment. (Pitler et al., 2009) showed that polarity features are helpful in distinguishing CONTINGENCY, as they contain more opposite polarity pairs than non-CONTINGENCY senses.

Also interesting is the high performance of EntRel, at 74.39% accuracy, compared to the relatively lower performance of Implicit EXPANSION, at 57.09%. Recall from Chapter 2 that the common practice is to include EntRel into Implicit EXPANSION when performing top-level classification due to its perceived interpretation as a form of EXPANSION. However, based on the large difference in performance and weight activations, it seems the language model learns these two representations quite differently, with much more ease for EntRel. We can observe these logistic weight activations in Figure 5.3 (b), which are much more noisy when compared to EntRel in Figure 5.3 (a).

In this chapter we introduced a preliminary investigation of discourse relation representation disentanglement. Our results show that a high capacity neural language model can learn discourse relation with unsupervised learning, and shows signs of disentanglement notably in the case of the `Implicit` CONTINGENCY sense. We hypothesized that our findings are supported by discourse relation theory, such as the connection between CONTINGENCY and polarity and the choice of dataset. Given this new research direction, much more work can be done. We believe finding parameters from a high capacity network trained on a large dataset in the same domain as the PDTB would yield better results. We believe such a model could generate interesting text conditioned on discourse relation, as (Radford et al., 2017) generated text conditioned on sentiment. Additionally, such a model would complement a supervised classification model presented in Chapter 4.

Chapter 6

Conclusion and Future Work

6.1 Summary

This thesis explored the recognition of implicit discourse relation (IDR) with neural networks, in particular with convolutional (CNN) and recurrent neural networks (RNN). In Chapter 3, we experimented with several varieties of CNN models. Our experiments showed that one of the key ingredients to a well performing CNN for IDR is to keep the number of layers shallow and separate the model inputs across discourse arguments. Convolution and pooling operations should not blend across `Arg1` and `Arg2`, they should be kept separate. Otherwise, due to the max-pooling the penultimate network layer operates on information which is not representative of the most important features in both arguments, but can be skewed towards a single one. We showed that small kernels work best. We did not see any benefit from the use of pretrained embeddings but believe that may be because of the limited depth of the convolutional network.

We hypothesized that building a pipeline system linking top-level classification with fine-grained classification could result in better fine-grained IDR. However, given the difficulty of top-level multiclass classification, misclassification error simply compounded the performance of a fine-grain classifier. We also experimented with an RNN layer to exploit the sequential nature of the data, but results were disappointing.

These results led us to create a sequence-to-sequence model in Chapter 4, which to our knowledge is the first of its kind for IDR. We experimented with two attention models, one where we

combine information from all decoder hidden states, and another where we only consider the last hidden state. We showed that the simpler attention model with a classification layer based only on its last hidden state could achieve state-of-the-art performance on fine-grained IDR, and state-of the art on some top-level discourse relation senses.

Classification models of Chapters 3 and 4 showed we could discover latent discourse relation from natural text in a supervised fashion. In Chapter 5 we explored the representation of discourse relation in completely unsupervised training. Our preliminary results showed that some discourse relations are automatically learned, but that more work is needed for this new research direction on discourse relation recognition.

6.2 Contributions

This thesis presented a number of theoretical and practical contributions, including:

- the implementation of various convolutional neural networks for IDR, including our participation to the CoNLL-2016 International Shared Task on Shallow Discourse Parsing (Laali et al., 2016) (see Chapter 3)
- insights relating discourse theory and convolutional neural network architectures for IDR, and the settings under which they work best (see Chapter 3)
- a novel neural network architecture, based on sequence-to-sequence models, for IDR which achieves state-of-the-art performance (see Chapter 4)
- insights into the possible automatic learning of discourse relation representation in unsupervised learning, a new research direction for discourse relations (see Chapter 5)

6.3 Future Work

In Chapter 3 we discussed the wide performance margin between `Explicit` and `Implicit` relations. Results in Section 3.3.3 showed how `Explicit` discourse relation recognition models perform quite well on the PTB test dataset but not so well on the Wikinews test dataset, with

high variance across models when compared with their `Non-Explicit` counterparts. This suggests `Explicit` models have begun to overfit the PTB dataset as a whole and further development needs to focus on alternative datasets, which, unfortunately are lacking. We have shown that neural networks achieve state-of-the art performance on IDR and do not seem to overfit to the current PTB domain. However, given the fact neural networks achieve their full potential on natural language processing tasks only once datasets reach a large minimal size (Zhang et al., 2015b), developing discourse relation datasets is key to the future development of better discourse relation sense classifiers in both cases of `Explicit` and `Non-Explicit` relations.

In Chapter 3 we examined a hierarchical system for IDR. However, to exploit the hierarchical nature of discourse relation senses, it would be interesting to explore another approach for future work, inspired in part by the hierarchical softmax used for language modelling (Morin and Bengio, 2005; Mnih and Hinton, 2009; Mikolov et al., 2013a). The hierarchical softmax is a trick used in neural networks to deal with large output distributions, such as large vocabularies. A Huffman tree of the vocabulary is built, assigning short binary codes to frequent words. The probability of words down the Huffman tree are conditioned on previous node splits. This resembles fine-grained classification where low-level sense subtypes are dependent on sense types, in turn dependent on top-level sense classes. Additionally, hierarchical softmax is implemented in a way such that the parameters used to predict each node all share common input features. Sharing neural network layers for multiple tasks, in our case predicting the various hierarchical discourse senses, has been shown to be beneficial, as recently and impressively as in AlphaGo Zero (Silver et al., 2017).

We also believe future work should focus on data augmentation and regularization for the PDTB dataset. Common techniques to combat overfitting include dropout (Hinton et al., 2012) and batch normalization (Ioffe and Szegedy, 2015), common to many supervised learning domains using neural networks including NLP. Data augmentation is another strategy quite common in computer vision (Krizhevsky et al., 2012; Antoniou et al., 2017) but less common in NLP. In NLP, model inputs, such as words or characters, cannot simply be transformed as done with images or speech recognition, since small perturbations can render a sentence meaningless. Some preliminary work suggests augmenting text by replacing some words with semantically close words based on a thesaurus (Zhang et al., 2015b). We believe a better alternative would be to replace words using close

words based on word embeddings, allowing for a much larger choice of vocabulary than a thesaurus, and objective closeness function.

In Section 3.3.3.2 we presented some preliminary analysis of the effect of CNN kernel filter size on IDR. We hypothesized that wider filters would map longer dependencies across discourse arguments. However this was not supported by the results in Table 3.5. In future work we would like to fully analyze the effects of various filter sizes as well as various simultaneous filters with at least 10-fold cross-validation. In Section 3.3.3.2 we also found no difference in using pre-trained word embeddings or training randomly initialized embeddings. In future it would be interesting to experiment with combining word embeddings with other strategies. For example, (Lee et al., 2017) represent their text input with a combination of two pretrained word embeddings and character embeddings, the latter which is trained simultaneously with their task of coreference resolution. Alternatively, Skip-Thought vectors (Kiros et al., 2015), where the entire sentence is embedded can be used instead of working with word embeddings.

In Chapter 4 we introduced novel attention models for IDR. In recent months, new kinds of attention mechanisms have been introduced for various tasks such as attention-over-attention for reading comprehension (Cui et al., 2017), self-attention for document summarization (Paulus et al., 2017) and multi-head attention for translation (Vaswani et al., 2017). We believe these could be quite valuable for IDR and encourage further research in this direction.

Finally, in Chapter 5 we introduced some preliminary work on disentanglement of discourse representations. We propose using alternative datasets more closely aligned with the PDTB content and style to discover discourse relation neurons. A high-capacity model trained on such a dataset may be able to generate text conditioned on discourse relation sense, an interesting prospective.

In this thesis we presented only a small fraction of potential research on implicit discourse relation recognition. Considering how far automatic IDR is from human performance and the relatively recent interest in applying neural network techniques, much more research still remains.

Bibliography

Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Advances in Neural Information Processing Systems (NIPS-2015)*, pages 649–657, Montreal, Canada, December 2015.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. In *Advances in Neural Information Processing Systems (NIPS-2001)*, pages 932–938. MIT Press, 2001.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155, 2003.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. Better document-level sentiment analysis from rst discourse parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP-2015)*, pages 2212–2218, Lisbon, Portugal, 2015. Association for Computational Linguistics.

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

Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-2010)*, pages 111–118, 2010.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue (SIGDIAL-2001)*, pages 1–10, 2001.

Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 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 (ACL-2016)*, pages 1726–1735, 2016a.

Xi Chen, , Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems 29 (NIPS-2016)*, pages 2172–2180, 2016b.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP-2014)*, pages 1724–1734, Doha, Qatar, 2014.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Advances in Neural Information Processing Systems 28 (NIPS-2014) Workshop on Deep Learning*, 2014.

Andre Cianflone and Leila Kosseim. Attention for implicit discourse relation recognition. In *(forthcoming) Proceedings of the 11th International Conference on Language Resources and Evaluation: (LREC-2018)*, Paris, France, 2018. European Language Resources Association (ELRA).

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Ronan Collobert and Jason Weston. Fast semantic extraction using a novel neural network architecture. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL-2007)*, pages 560–567, 2007.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL-2017)*, pages 593–602, 2017.

Jeffrey L Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

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

Katherine Forbes-Riley, Bonnie Webber, and Aravind Joshi. Computing discourse semantics: The predicate-argument semantics of discourse connectives in d-ltag. *Journal of Semantics*, 23(1): 55–106, 2005.

Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *IEEE International Conference on Neural Networks*, volume 1, pages 347–352, 1996.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27 (NIPS-2014)*, pages 2672–2680, 2014.

Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. *Proceeding of the 2017 International Conference on Learning Representation (ICLR-2017)*, 2017.

Alex Graves. Sequence transduction with recurrent neural networks. *International Conference of Machine Learning (ICML-2012) Workshop on Representation Learning*, 2012.

Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP-2013)*, pages 6645–6649, 2013.

Geoffrey Hinton, J. L. McClelland, and D. E. Rumelhart. Distributed representations. In David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pages 77–109. MIT Press, Cambridge, MA, USA, 1986.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91, 1991.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In John F. Kolen and Stefan C. Kremer, editors, *Field Guide to Dynamical Recurrent Networks*. Wiley-IEEE Press, 1st edition, 2001.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML-2015)*, pages 448–456, 2015.

Peter Jansen, Mihai Surdeanu, and Peter Clark. Discourse complements lexical semantics for non-factoid answer reranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-2014)*, pages 977–986, Baltimore, Maryland, 2014.

Yangfeng Ji and Jacob Eisenstein. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-2014)*, pages 13–24, 2014.

Yangfeng Ji and Jacob Eisenstein. One vector is not enough: Entity-augmented distributional semantics for discourse relations. *Transactions of the Association for Computational Linguistics (TACL)*, pages 329–344, June 2015.

Yangfeng Ji and Noah A. Smith. Neural discourse structure for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL-2017)*, pages 996–1005, 2017.

Ping Jian, Xiaohan She, Chenwei Zhang, Pengcheng Zhang, and Jian Feng. Discourse relation sense classification systems for conll-2016 shared task. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2016)*, pages 158–163, Berlin, Germany, 2016.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-2015)*, pages 2342–2350, 2015.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-2014)*, pages 655–665, 2014.

Yusuke Kido and Akiko Aizawa. Discourse relation sense classification with two-step classifiers. In

Proceedings of the Twentieth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2016), pages 129–135, 2016.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP-2014)*, pages 1746–1751, Doha, Qatar, 2014.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016)*, pages 2741–2749, Phoenix, Arizona, USA, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceeding of the 2015 International Conference on Learning Representation (ICLR-2015)*, San Diego, California, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceeding of the 2014 International Conference on Learning Representation (ICLR-2014)*, Banff, Canada, 2014.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28 (NIPS-2015)*, pages 3294–3302. Curran Associates, Inc., 2015.

Ben Krause, Liang Lu, Iain Murray, and Steve Renals. Multiplicative lstm for sequence modelling. In *Proceeding of the 2017 International Conference on Learning Representation (ICLR-2017)*, Toulon, France, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS-2012)*, pages 1097–1105, 2012.

Majid Laali, Elnaz Davoodi, and Leila Kosseim. The CLaC discourse parser at CoNLL-2015. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2015)*, Beijing, China, 2015.

Majid Laali, Andre Cianflone, and Leila Kosseim. The CLaC discourse parser at CoNLL-2016. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2016)*, Berlin, Germany, 2016.

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

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

Yann LeCun. Generalization and network Design Strategies. CRG-TR-89-4. Technical report, University of Toronto, 06 1989.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The Handbook of Brain Theory and Neural Networks*, 3361(10):1995, 1995.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP-2017)*, pages 188–197, 2017.

Jiwei Li, Rumeng Li, and Eduard H Hovy. Recursive deep models for discourse parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP-2014)*, pages 2061–2069, Doha,Qatar, 2014.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-2009)*, pages 343–351, 2009.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 997–1006, 2011.

Annie Louis, Aravind Joshi, and Ani Nenkova. Discourse indicators for content selection in summarization. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 147–156, 2010.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 2013 International Conference on Machine Learning (ICML-2013)*, volume 30, 2013.

William C Mann and Sandra A Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281, 1988.

Daniel Marcu. Discourse trees are good indicators of importance in text. *Advances in automatic text summarization*, 293:123–136, 1999.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The Penn treebank. *Computational Linguistics*, 19(2):313–330, June 1993. ISSN 0891-2017.

Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2015.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. *International Conference on Learning Representations (ICLR-2018)*, 2018.

Todor Mihaylov and Anette Frank. Discourse relation sense classification using cross-argument semantic similarity based on word embeddings. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2016)*, pages 100–107, Berlin, Germany, 2016.

Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. *Spoken Language Technology Workshop (SLT), 2012 IEEE*, 12:234–239, 2012.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS-2013)*, pages 3111–3119, Lake Tahoe, USA, 2013b.

Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems 21 (NIPS-2009)*, pages 1081–1088, 2009.

Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252, 2005.

Minh Nguyen. Sdp-jaist: A shallow discourse parsing system at CoNLL 2016 shared task. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2016)*, pages 143–149, Berlin, Germany, 2016.

Stephan Oepen, Jonathon Read, Tatjana Scheffler, Uladzimir Sidarenka, Manfred Stede, Erik Velldal, and Lilja Øvrelid. Opt: Oslo–Potsdam–Teesside. Pipelining rules, rankers, and classifier ensembles for shallow discourse parsing. *Proceedings of the Twentieth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2016)*, pages 20–26, 2016.

Tsuyoshi Okita, Longyue Wang, and Qun Liu. The DCU discourse parser: A sense classification task. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning-Shared Task (CoNLL-2015)*, pages 71–77, 2015.

Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-2014)*, pages 1717–1724, 2014.

Maria Leonor Pacheco, I-Ta Lee, Xiao Zhang, Abdullah Khan Zehady, Pranjal Daga, Di Jin, Ayush Parolia, and Dan Goldwasser. Adapting event embedding for implicit discourse relation recognition. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2016)*, pages 136–142, Berlin, Germany, 2016.

Joonsuk Park and Claire Cardie. Improving implicit discourse relation recognition through feature set optimization. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL-2012)*, pages 108–112, 2012.

Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

Emily Pitler and Ani Nenkova. Using syntax to disambiguate explicit discourse connectives in text. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL/IJCNLP-2009)*, page 13–16, Suntec, Singapore, August 2009.

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

Emily Pitler, Annie Louis, and Ani Nenkova. 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 (ACL/IJCNLP-2009)*, pages 683–691, 2009.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. The Penn Discourse TreeBank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC-2008)*, Marrakech, Morocco, 2008.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. Shallow discourse parsing using convolutional neural network. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2016)*, pages 70–77, Berlin, Germany, 2016.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceeding of the 2016 International Conference on Learning Representation (ICLR-2016)*, 2016.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

Israel Ramat-Gan. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-2014)*, pages 302–308, Baltimore, Maryland, USA, 2014.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, December 2004. ISSN 1532-4435.

Kipling Rudyard. *From sea to sea : letters of travel*. Doubleday & McClure company, 1899.

David E Rumelhart, Geoffrey Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–538, 1986.

Attapol Rutherford and Nianwen Xue. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2014)*, pages 645–654, Gothenburg, Sweden, April 2014.

Attapol Rutherford and Nianwen Xue. Robust non-explicit neural discourse parser in english and chinese. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning: Shared Task (CoNLL- 2016)*, pages 55–59, Berlin, Germany, 2016.

Niko Schenk, Christian Chiarcos, Kathrin Donandt, Samuel Rönnqvist, Evgeny A Stepanov, and Giuseppe Riccardi. Do we really need all those rich linguistic features? a neural network-based approach to implicit sense labeling. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2016)*, pages 41–49, Berlin, Germany, 2016.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

Holger Schwenk and Jean-Luc Gauvain. Connectionist language modeling for large vocabulary

continuous speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-2002)*, volume 1, pages I–765, 2002.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP-2013)*, pages 1631–1642, 2013.

Swapna Somasundaran, Galileo Namata, Janyce Wiebe, and Lise Getoor. Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-2009)*, pages 170–179, 2009.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine learning research*, 15(1):1929–1958, 2014.

Philip J. Stone and Earl B. Hunt. A computer approach to content analysis: Studies using the general inquirer system. In *Proceedings of the May 21-23, 1963, Spring Joint Computer Conference*, AFIPS '63 (Spring), pages 241–256, New York, NY, USA, 1963.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS-2014)*, pages 3104–3112, 2014.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Llion Jones, Jakob Uszkoreit, Aidan N Gomez, and Ł ukasz Kaiser. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS-2017)*, pages 5994–6004. Curran Associates, Inc., 2017.

Jianxiang Wang and Man Lan. Two end-to-end shallow discourse parsers for English and Chinese in CoNLL-2016 shared task. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2016)*, pages 33–40, Berlin, Germany, 2016.

Bonnie Webber. D-ltag: extending lexicalized tag to discourse. *Cognitive Science*, 28(5):751–779, 2004.

Gregor Weiss and Marko Bajec. Discourse sense classification from scratch using focused rnns. *Proceedings of the Twentieth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2016)*, pages 50–54, 2016.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP-2005)*, pages 347–354, 2005.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi PrasadO Christopher Bryant, and Attapol T Rutherford. The conll-2015 shared task on shallow discourse parsing. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2015)*, pages 1–16, Beijing, China, 2015.

Nianwen Xue, Hwee Tou Ng, Attapol Rutherford, Bonnie Webber, Chuan Wang, and Hongmin Wang. Conll 2016 shared task on multilingual shallow discourse parsing. *Proceedings of the Twentieth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2016)*, pages 1–19, 2016.

Wen-tau Yih, Kristina Toutanova, John C Platt, and Christopher Meek. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL-2011)*, pages 247–256, 2011.

Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan, and Junfeng Yao. Shallow convolutional neural network for implicit discourse relation recognition. In *Proceedings of the 2015*

Conference on Empirical Methods in Natural Language Processing (EMNLP-2015), pages 2230–2235, Lisbon, Portugal, September 2015a. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS-2015)*, pages 649–657, Montreal, Canada, December 2015b.

Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *The 8th International Joint Conference on Natural Language Processing (IJCNLP-2017)*, 2015.

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