Argument Labeling of Explicit Discourse Relations using LSTM Neural Networks

Sohail Hooda

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

May 2019

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Sohail Hooda**

Entitled: **Argument Labeling of Explicit Discourse Relations using LSTM Neural Networks**

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. Emad Shihab

_____ Examiner
Dr. Tien Bui

_____ Examiner
Dr. Tristan Glatard

_____ Supervisor
Dr. Leila Kosseim

Approved by _____
Lata Narayanan, Chair
Department of Computer Science and Software Engineering

_____ 2019

_____
Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

Argument Labeling of Explicit Discourse Relations using LSTM Neural Networks

Sohail Hooda

A discourse relation can be described as a linguistic unit that is composed of sub-units that, when combined, present more information than the sum of its parts. A discourse relation is usually comprised of two arguments that relate to each other in a given form. A discourse relation may have another optional sub-unit called the discourse connective that connects the two arguments and describes the relationship between the two more explicitly. This is called Explicit Discourse relation. Extracting or labeling arguments present in an explicit discourse relations is a challenging task. In recent years, due to the CoNLL competitions, feature engineering has been applied to allow various machine learning models to achieve an F-measure value of about 55%. However, feature engineering is brittle and hand-crafted, requiring advanced knowledge of linguistics as well as the dataset in question. In this thesis, we propose an approach for segmenting (or identifying the boundaries of) `Arg1` and `Arg2` without feature engineering. We introduce a Bidirectional Long Short-Term Memory (LSTM) based model for argument labeling. We experimented with multiple configurations of our model. Using the Penn Discourse Treebank (PDTB) dataset, our best model achieved an F1 measure of 23.05% without any feature engineering. This is significantly higher than the 20.52% achieved by the state of the art Recurrent Neural Network (RNN) approach, but significantly lower than the feature based state of the art systems. On the other hand, because our approach learns only from the raw dataset, it is more widely applicable to multiple textual genres and languages.

Acknowledgments

I would like to thank Dr. Leila Kosseim for all her efforts in guiding me and in encouraging me throughout my Masters. A few years ago, when I decided to enroll myself in a Master's program, my goal was to learn more about AI and Natural Language Processing. Dr. Kosseim has been nothing short of a shining star in my pursuit of this knowledge. Starting from her course in Introduction to AI, to her readings of my works, it has been an absolute pleasure to have her as my mentor, my teacher and my supervisor. I could not have asked for more passionate guide throughout my journey as a student of this discipline.

I would also like to thank my lab colleagues, Majid Laali, Andre Cianflone and Elnaz Davoodi for indulging me and my ideas and for allowing me to be a part of their fascinating discussions at the lab.

Lastly, I would like to dedicate my thesis to my wife, Alima Alibhay and my family who have been a never-ending source of love and encouragement throughout my Masters and even before as well. I would not be here without their unconditional love, guidance and support throughout my career.

Contents

List of Figures

List of Tables

Acronyms

ANN Artificial Neural Network. 31

BOW Bag of Words. 7

CoNLL Conference on Computational Natural Language Learning. ix, 17, 21, 22, 23, 37, 38, 40, 46, 47, 51, 52, 54, 56, 57, 58, 61

CRF Conditional Random Field. 22

D-LTAG Discourse Lexicalized Tree Adjoining Grammar. 9, 12, 60

GloVe Global Vectors for Word Representation. 29, 44, 45, 50, 51, 52, 57, 59

GRU Gated Recurrent Unit. 63

HMM Hidden Markov Model. 7

KL Kullback Liebler Divergence. 43

LSTM Long Short-Term Memory. iii, v, vi, ix, 5, 25, 27, 31, 32, 33, 41, 45, 50, 52, 56, 57, 58, 63

NLP Natural Language Processing. 7, 8, 9, 33

PDTB Penn Discourse Treebank. iii, viii, ix, 9, 12, 14, 15, 16, 18, 19, 21, 22, 37, 38, 39, 62, 63, 65

POS part of speech. 23

RNN Recurrent Neural Network. iii, v, ix, 5, 23, 25, 27, 31, 32, 33, 52

RST Rhetorical Structure Theory. 8, 9, 12, 60

RST-DT RST-Discourse Treebank. viii, 9, 12, 14, 60, 63

SVM Support Vector Machine. 7

Chapter 1

Introduction

1.1 Problem Definition

Consider the following statement:

"Never put off till tomorrow what may be done the day after tomorrow just as well." - Mark Twain

The textual construction provided above can be broken down into many textual units. One level of textual units could be syntactic or parts of speech. This level gives an insight into the structure of the language and the usage of the tokens provided in the dictionary that form the vocabulary of the language. Another level could be the semantic level where the word tokens as well as their combination could be used to infer understanding of the overall sentence.

Note that the semantic de-constructions of the textual segment provided is able to capture the meaning of the segment independently. In other words, the de-constructed semantic units each represent some meaning independent of the rest of the text. These meanings can be composed to re-create the same meaning of the original text itself. Thus in this example, the meaning of the entire text can be mapped to its individual semantic units in a lossless fashion.

However, consider the following example:

"Always do what is right. It will gratify half of mankind and astound the other." - Mark Twain

In this case, the following larger semantic units can be created:

"Always do what is right." and

"It will gratify half of mankind and astound the other."

These units (or any other combination of further de-constructed semantic units) are unable to express the entire meaning conveyed by the original text in an independent manner. A discourse relation is a relationship between two independent textual units that, when put together, add further meaning to the overall text.

Discourse relations can be either implicit, such as the one expressed above, or explicit such as the one exemplified below:

"Thunder is good, thunder is impressive; **but** it is lightning that does the work." - Mark Twain

In an explicit discourse relation, the relationship between the two textual units is explicitly provided via a discourse connective. **The goal of our thesis is to identify explicit discourse relations and automatically extract the textual units or arguments present in a discourse relation.**

Separating arguments present within a discourse relation is a difficult task. This is due to two main reasons.

(1) A discourse relation can span multiple sentences, most of which can be removed without much loss of information in the original discourse relation.

(2) Within the segment of text, the structure and placement of the arguments vary greatly.

1.2 Motivation

Automated argument labeling indeed poses a significant challenge. However, a generalized solution to this problem can greatly advance our understanding and usage of various corpora. For example, recognizing discourse relations, can permit a system to generate higher quality summaries of a given document. Question answering can also benefit greatly if the relationship can be established automatically between sentences that are significantly far apart in a given relation. Finally, in the context of written texts as well as conversations in real-time, analysis and understanding of discourse relations could identify hidden meanings, or even general inconsistencies within a given document, allowing for further insight into the author's works. Thus, even though the problem statement is extremely challenging, the advantages of resolving it, merit further investigation and study into this subject.

On the practical side, current solutions to this problem are brittle and focus on exploiting features or logical consistencies (grounded in syntax and semantics), that hint towards the start and end of a given argument as well as the discourse connective. These consistencies cannot be completely generalized and may only present themselves in some corpora as well as languages but not others. Hence, there is a need for investigating approaches that can lead to a generalized solution for the problem at hand.

1.3 Contributions

Our work has made advances in generalizing a solution to the problem of segmenting (or identifying the boundaries of) `Arg1` and `Arg2` defined above. We implemented a neural network model based on bidirectional LSTM units that allows us to explore a less manual approach where the system can learn the features automatically based on the training dataset provided. We also experimented with injecting pre-computed syntactic information for the word tokens in the dataset using pre-computed word embeddings. This allows us to realize the potential of providing pre-computed information as a feature to the model in order to enhance the performance of the approach. Our best model achieved an F1 measure of 23.05% without any feature engineering. This is significantly higher than the 20.52% achieved by the state of the art RNN approach, but significantly lower than the feature based state of the art systems. On the other hand, because our approach learns only from the raw dataset, it is more widely applicable to multiple textual genres and languages. This work has also been published in the proceeding of the 2017 edition of the Conference *Recent Advances in Natural Language Processing* (Hooda & Kosseim, 2017).

1.4 Structure of the thesis

This chapter attempts to introduce the reader to the problem domain briefly so as to provide a general introduction to the subject. The rest of the thesis is structured to explain the problem and our work in the following way. Chapter 2 explains the problem domain in greater detail while presenting a historical review of the work done for this task. Chapter 3 discusses our approach and describes our experiments in detail. Chapter 4 presents a review of the results obtained from our

approach and compares them against other approaches established in the field. Finally, in Chapter 5, we present challenges and further possibilities along with work that can be performed to extend our study into this field.

Chapter 2

Literature Review

Chapter 2 provides a literature review on the subject of Discourse Parsing and focuses more specifically on Argument Labeling approaches.

2.1 History of NLP

In recent years, Natural Language Processing (NLP) has seen strides of advances thanks to the increase in available data, computing resources, better algorithms and a deeper understanding of languages (Hirschberg & Manning, 2015). Much of this has happened gradually since the 1950s where Language Processing involved in-depth knowledge of rules of the language itself that were coded into algorithms to better comprehend and parse complex syntactic structures. By the early 1970s, a need for more extensible approaches was deemed necessary and during the 1980s statistical approaches were widely considered to provide more generalized and adaptable solutions. The use of probabilistic methods on issues concerning languages became ubiquitous and machine-readable data became readily available in huge quantities (K. S. Jones, 2001). The field of NLP slowly transformed into the big data domain. This allowed research to move away from understanding and coding features into algorithms into building more robust and generalizable systems for processing natural languages. Algorithms that could apply specific features automatically based on historical data such as Naive Bayes were considered to be of paramount importance. Consequently, it was realized that simple methods such as Bag of Words (BOW) trained on large datasets would achieve

significant results. Later on, even more complex statistical approaches such as Hidden Markov Models (HMMs) and Support Vector Machines (SVMs) were applied (Manning & Schütze, 1999). However, increasing the baseline set by the earlier and simpler statistical methods proved to be quite challenging. This further motivated the need for even more complex and generalized approaches along with a richer understanding of languages. In the last decade, a resurgence of neural networks in Machine Learning has inspired Computational Linguistics to push the boundaries even further while simultaneously removing the need for learning features of the dataset (Jian, She, Zhang, Zhang, & Feng, 2016). This has further garnered a special interest as neural networks are adept at generating representations of a given dataset which can then be extrapolated to other sample data in order to parse syntactic, semantic and even complex discourse structures. These representations can also be studied further to gain a better understanding of the language at hand (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013). Thus, in a way, we have gone full circle from understanding languages so that they may be applied to algorithms, to applying language data to algorithms so that it may be better understood for further applications.

2.2 Discourse and Discourse Relations

With the availability of large scale annotated datasets, research in NLP has incorporated works not only on the syntactic and semantic units of languages but also on discourse structures. This thesis focuses on the rhetorical or discourse nature of linguistic structures and attempts for the automated understanding of various units of a discourse segment. In the late 1980s, there was an emergence in the understanding of linguistic units larger than those comprised at the semantic and syntactic levels. Rhetorical Structure Theory (RST) was formalized to describe the organization of natural texts (Mann & Thompson, 1988). RST outlines the hierarchical nature of a text and describes how large textual units may be subdivided (Mann & Thompson, 1988). Consider, for example, the statement “It was time for lunch and John was hungry”. According to the RST framework, a discourse can be broken down into unitary constituents that are called *spans*. In this case, the constituent spans would be “It was time for lunch”, and “and John was hungry”. Pairs of spans can then exhibit a relationship that connects them together. Thus a relation is a connection by which a

pair of spans relate to each other. This is one of the integral and significant elements of the RST framework. As an example, Figure 2.1 shows an example of a hierarchical tree structure built using the software of (Zeldes, 2016).

An RST relation defines a semantic or rhetorical relationship that holds between two non-overlapping segments of text or spans. These pairs of spans can be further categorized as a *nucleus* or a *satellite*. In general, a nucleus is the span that is more essential and deletion or replacement of it will render the satellite span incomprehensible. In contrast, a satellite span is the less essential constituent of a relation and lends itself easily to substitution without making the overall relation necessarily incomprehensible (Mann & Thompson, 1988). In our example, the first span, "It was time for lunch", would be the nucleus whereas the latter, "John was hungry", would form a satellite.

From another perspective, these spans can also be seen to have an anaphoric connection between them. This connectedness is expressed by a *discourse connective*. In this case, the structure of a relation is slightly modified to include the discourse connective as a separate constituent along with the two text spans that it connects. Based on this approach, we can restructure our example statement to have the spans "It was time for lunch" and "John was hungry" which are related by the discourse connective *"and"* representing a conjunction or simultaneity of the two events co-occurring. This is a more lexically grounded approach and relies on an explicit connection formed by the discourse connective or an implicit connection formed via other anaphoric entities in the constituent arguments (Webber, Stone, Joshi, & Knott, 2003). More formally, the latter approach looks for discourse connectives as predicates that connects two arguments together. This approach is termed Discourse Lexicalized Tree Adjoining Grammar (D-LTAG) (Polanyi, Culy, Van Den Berg, Thione, & Ahn, 2004; Webber, 2004). The RST and the D-LTAG frameworks are considered to be the two most popular frameworks for analyzing discourse in the field of NLP. D-LTAG has given rise to the largest dataset of discourse annotation to date: PDTB (see Section 2.3.

Another area where the RST-DT and the PDTB diverge is in the classification of the senses of a discourse relation. While the PDTB implements a deep tree-based sense classification system, RST-DT attempts a more flattened approach. Overall, the PDTB defines 4 major classes which, in turn, have deeper subclasses. These subclasses detail the sense classification of a given discourse relation. On the other hand, the RST-DT defines 16 classes that specify over 78 relation types. The

details of the senses for RST-DT and PDTB are provided in Figure 2.2 and Figure 2.3 respectively.

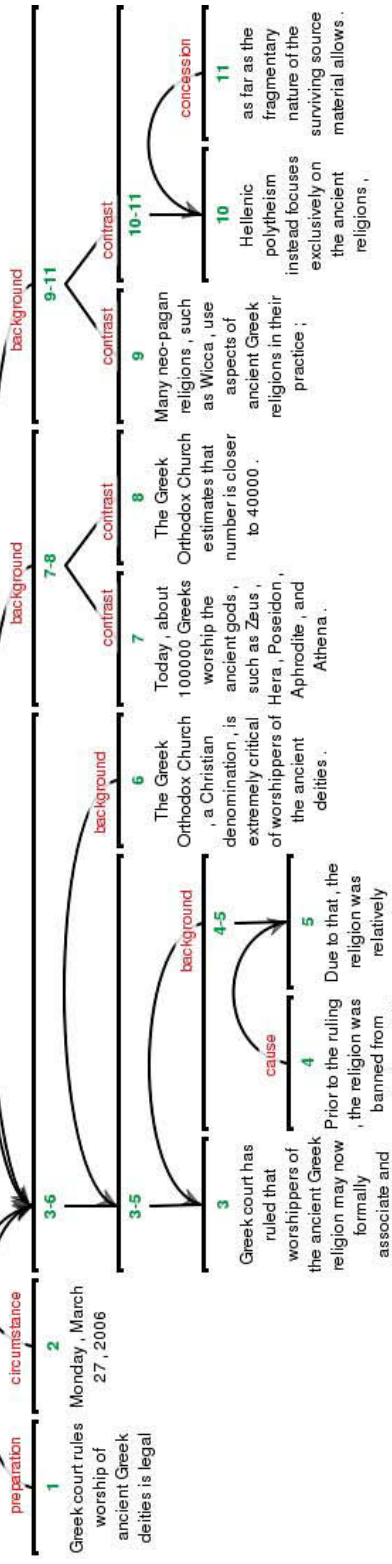

Figure 2.1: A hierarchical discourse relation annotated using the RST Framework.

(1) **Attribution**: attribution, attribution-negative

(2) **Background**: background, circumstance

(3) **Cause**: cause, result, consequence

(4) **Comparison**: comparison, preference, analogy, proportion

(5) **Condition**: condition, hypothetical, contingency, otherwise

(6) **Contrast**: contrast, concession, antithesis

(7) **Elaboration**: elaboration-additional, elaboration-general-specific, elaboration-part-whole, elaboration-process-step, elaboration-object-attribute, elaboration-set-member, example, definition

(8) **Enablement**: purpose, enablement

(9) **Evaluation**: evaluation, interpretation, conclusion, comment

(10) **Explanation**: evidence, explanation-argumentative, reason

(11) **Joint**: list, disjunction

(12) **Manner-Means**: manner, means

Figure 2.2: List of senses and relation types as defined in the RST-DT (Carlson & Marcu, 2001)

2.3 Discourse Corpora

Based on the RST framework and the D-LTAG approach, two major datasets have been created each of which uses one of the two theories to define and tag discourse relations. The RST-DT (Carlson, Okurowski, & Marcu, 2002) is grounded in the RST framework of Mann and Thompson (1988). In contrast, the PDTB (Prasad et al., 2008) relies on the anaphoric approach of characterizing discourse relations. The PDTB is a superset of the RST-DT and unlike the RST-DT, it does not necessarily follow the hierarchical classification of subdividing a large body of text into relations connected via a tree structure.

The approach adopted by the PDTB thus results in shallow discourse relations. These shallow discourse relations are further sub-divided into Explicit, Implicit and non-Implicit discourse relations. The Explicit relations are those where a lexical entry representing a discourse connective

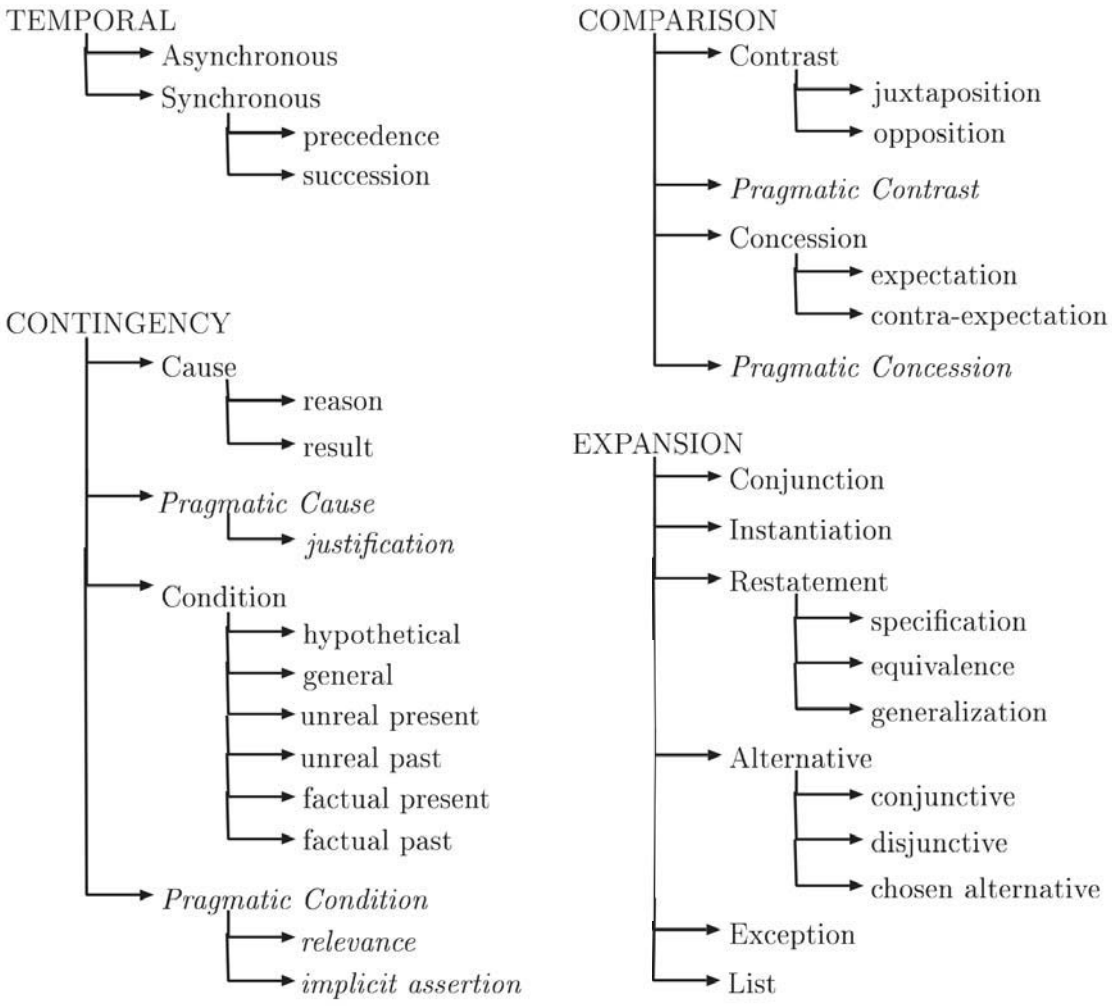

Figure 2.3: List of senses as defined in the PDTB (Prasad et al., 2007)

exists. For example, in:

(1) "...even if **you're not a reporter or a researcher or a scholar or a member of Congress**" (Prasad et al., 2007).

Here, the underlined terms represents the discourse connective, forming a relationship between the two arguments that occur before and after it. Adjacent sentence pairs where a relation can be inferred but no term explicitly signals the relation, are called Implicit relations. For example, the following relation contains an Implicit relation; where the term "so" is not explicitly used but can be inferred

Table 2.1: Number of instances in the PDTB Dataset (Prasad et al., 2008) and in the RST Dataset (Carlson et al., 2003)

Dataset	**Count**
PDTB Explicit	18,459
PDTB Implicit	16,224
PDTB Non-Implicit	6,088
PDTB Total	40,771
RST-DT	21,789

(2) "The projects already under construction will increase Las Vegas's supply of hotel rooms by 11,795, or nearly 20%, to 75,500. (Implicit = so) By a rule of thumb of 1.5 new jobs for each new hotel room, Clark County will have nearly 18,000 new jobs." (Prasad et al., 2007)

Finally, sentence pairs where a relation does exist but no Implicit connectives are considered appropriate are non-Implicit relations. The following sentence serves as an example of a non-Implicit relation:

(3) "The market for export financing was liberalized in the mid-1980s, forcing the bank to face competition." (Prasad et al., 2007)

Table 2.1 shows a comparison between the number of discourse relations in the PDTB corpus in contrast to those in the RST-DT corpus.

Since there is no convention on the predicate-argument nature of discourse relations, the argument which is syntactically connected to the connective is called *Arg2* and the other argument is referred to as *Arg1*. Figure 2.4 illustrates this syntactic connectivity.

Such problems will require considerable skill to resolve. ***However*, neither Mr. Baum nor Mr. Harper has much international experience.**

Figure 2.4: Example from the PDTB. `Arg1` is stylized using regular font. `Arg2` and the Connective are bold. The connective is also *italicized* to indicate its syntactic connectedness with `Arg2`.

The goal of our research is to segment explicit relations defined in the PDTB dataset texts into `Arg1` and `Arg2`. These explicit relations are based on discourse connectives of three grammatical varieties:

(i) *Most balloonists seldom go higher than 2,000 feet* and **most average a leisurely 5-10 miles an hour.**

(ii) *Metropolitan Houstons population has held steady over the past six years.* And, **personal income, after slumping in the mid-1980s, has returned to its 1982 level in real dollar terms.**

Figure 2.5: Examples of discourse relations taken from PDTB. In (i), the arguments exist in a single sentence, whereas in (ii) the arguments exist in separate sentences. In both cases, the `Arg1` is styled in *italics*, `Arg2` is represented using **bold** formatting, and the discourse connective is underlined.

(1) Subordinating Conjunctions such as *"because"*, *"although"*, and *"however"*,

(2) Coordinating Conjunctions such as *"and"*, and *"or"* and

(3) Discourse Adverbials such as *"for example"*, and *"instead"*.

Each of these three types of connectives dictate the position of the arguments in the relation. Subordinating conjunctions have the subordinate clause bound to the connective which becomes `Arg2`. In a relation dictated by subordinating conjunctions, the `Arg2` can precede `Arg1` or follow it, or even be embedded within `Arg1`. On the other hand, coordinating conjunctions and discourse adverbials simply generate a straightforward relation where `Arg1` is followed by `Arg2`. However, there are scenarios for discourse adverbials where an `Arg1` may be embedded within an `Arg2` (Prasad et al., 2007). These arguments of a given relation are neither constrained spatially nor structurally. That is to say that arguments may appear as separate disconnected sentences or within a single sentence. For example, consider the sentences in Figure 2.5.

However, a minimality constraint (Prasad et al., 2007) does apply to an argument which states that an argument must contain the necessarily minimal amount of information required for complete interpretation of a relation. According to the PDTB, any other extraneous span of text that may exist in between the arguments is considered to be supplementary information. Figure 2.6 shows an example of supplementary text for `Arg1`.

The PDTB labels this information as "Sup1" and "Sup2" with respect to which argument it relates to. The minimality principle also dictates the generation of *Attribution* segments. The Attribution segments are textual spans that specify arbitrary relationships between an agent and an

"Underclass youth are a special concern. Are such expenditures worthwhile, then? Yes, if targeted."

Figure 2.6: An example of a discourse relation from the PDTB with supplementary information for `Arg1` is underlined.

"($_{Sup1}$ Workers described "clouds of blue dust") *that hung over parts of the factory,* even though **exhaust fans ventilated the area.**"

Figure 2.7: An example of a discourse relation with attribution from the PDTB. The text in parenthesis is supplementary information that is not necessarily required for the discourse relation. The `Arg1` is in *italics*, `Arg2` is in **bold** and the discourse connective is underlined.

abstract object. These attribution spans may or may not be considered necessary to satisfy the minimality principle of establishing a valid argument in a relation (Prasad, Webber, & Joshi, 2014). Figure 2.7 shows an example of a case where supplementary information is present in a discourse relation

The PDTB characterizes the arguments or predicates of a relations as abstract objects that represent events, states or propositions. This results in arguments that are non-clausal in structure and may signal event information as well as references to abstract objects (Asher, 1993). Consequently this allows argument spans to be useful in practical contexts of automatic text summarization and natural language generation as well as in generating knowledge bases and question answering systems that go beyond factoid based mechanisms. These practical applications rely significantly on identifying the location and length of arguments in a given relation. Since `Arg2` is established to be structurally related to the discourse connective, identifying the connective can lead to easy identification of `Arg2` and understanding the minimality of a predicate level clause can give an idea of its length as well. `Arg1`, however, can be present next to the connective or any arbitrary distance away and therefore can be much harder to locate and define. The PDTB dataset contains relations where `Arg1` is identified either in the same sentence as the connective, in a preceding sentence, in a sentence following that of the connective or in a non-adjacent sentence. Variations also exist on `Arg1` occupying an entire sentence or more or even sharing sentence or sentences with supplementary textual spans. While there is no definite structural pattern to a relation that can point accurately the

location and the length of its constituting Arg1, the majority of the relations in the PDTB dataset are either in the same sentence as the connective or in a sentence immediately preceding it or in a non-adjacent sentence with respect to the connective. In regards to the span, Arg1 tends to occupy a single full sentence or just part of it (Pitler et al., 2008). This allows for some structural features to be incorporated in algorithms for locating Arg1 as well as Arg2 once the discourse connective is identified. Since these connectives are well defined, locating and bounding Arg2 is much less of a challenge as compared to Arg1. Table 3.1 provides counts on the distribution of discourse relations based on the position of Arg1 with respect to the discourse connective.

2.4 Prior Work

As indicated in Section 2.3, our goal is to segment (or identify the boundaries of) Arg1 and Arg2. In the last few years much work has been accomplished to push the understanding of argument identification and spanning length. In part, an important reason for this advancement in the field can be attributed to the CoNLL Shared Tasks for the Nineteenth and Twentieth Conference on Computational Natural Language Learning (Xue, Ng, Pradhan, Bryant, & Rutherford, 2015; Xue et al., 2016). These shared tasks are in fact competitions on discourse parsing organized by the CoNLL Committee that a number of well known research teams in the field participated in. The shared task included the challenge of identifying as well as categorizing discourse relations. This included parsing relations, the sense expressed in a relation, the arguments of the relation and finally the connective in case of an explicit relation. One sub task of the Discourse Parsing competition was *argument span labeling* for explicit relations. As a result, new techniques, approaches as well as features were identified and utilized that improved the state of the art for argument span labeling as dictated by the minimality principle.

Prior to the CoNLL tasks, some work had already been done for argument labeling and feature identification for relations and their arguments. In 2005, some features had been described that relied on syntactic structures of an intra-sentential relation based on attribution (see Section 2.3) to identify arguments and their spans (Dines et al., 2005). One of the first attempts in argument labeling that utilized machine learning approaches was proposed by Wellner and Pustejovsky. In

Drug makers shouldn't be able to duck liability *because* **people couldn't identify precisely which identical drug was used.**

Figure 2.8: An illustration of a discourse relation with head word identification (Wellner & Pustejovsky, 2007). In this case, the head word for `Arg2` in **bold** is identify. The discourse connective is marked in *italics*

2007, Wellner and Pustejovsky proposed techniques based on machine learning, specifically Naive Bayes approaches to extract argument spans. The idea was to re-cast the argument labeling problem into a form where a "head" word dictated the argument in a given parse tree of a relation. For example, in the discourse relation in Figure 2.8, the head word for `Arg2` would be identify. Thus the problem was re-formulated into identifying only the head words of an argument. The end of an argument was considered to be simply the "natural" syntactic end of the text segment. A log-linear ranking model belonging to the family of Maximum Entropy based models was used. A classifier was constructed for each argument with various features in order to determine the true arguments from the argument candidates. The candidates, in turn were identified using a head finding algorithm developed previously and modified slightly (Wellner & Pustejovsky, 2007). Once a given argument for `Arg1` was identified, it was paired against candidates for `Arg2` to find the best matching argument candidate for the given relation. Furthermore, many more features based on the constituent, dependencies, and syntactic knowledge of the relation and the connective as well as contextual features were identified and employed in this approach.

The work of Wellner and Pustejovsky (2007) was further expanded upon by Elwell and Baldridge (2008), who realized that distinct models targeting the different kinds of relations (based on the explicitly defined discourse connective) produced better results then generalizing all senses into a single model. In this case, however, issues relating to lack of adequate amounts of data were encountered. This was balanced by creating type specific models for the 3 different types of connectives:

- Coordinating conjunctions
- Subordinating conjunctions
- Adverbial phrases

A generalized model was also included along with models for each of the different types of connectives as well. The final classifiers were built via an optimized version of all 4 models combined via linear interpolation. This allowed for building a classifier that accounts for the specificity of some relations while maintaining the generalized understanding of all relations as well. A balance in the data availability was also made possible via the interpolation of the specific, semi-specific and generalized models together. Along with this approach, newer features were also introduced that defined the morphological properties of discourse connectives as well as their arguments, syntactic configurations and finally allowed for a wider context that included the previous as well as the following connectives. The latter feature assisted specifically in determining the given relation more coherently based on the connectives in neighbouring relations. This technique improved the absolute accuracy by about 3.6% over the model proposed by Wellner and Pustejovsky, bringing it to an accuracy of 82.0% for `Arg1` and 93.7% for `Arg2` on the gold standard of the PDTB Dataset (see Section 2.3). Accuracy, here, is defined as the percentage of correctly identified Arguments. However, it should be noted that while this method does perform better on argument identification tasks, it does not improve the state of the art on the problem of argument span labeling.

The work of Prasad, Joshi, and Webber (2010), shifted the focus in argument identification from detecting the head words as shown in Figure 2.8 of an argument to identifying the sentence containing the argument itself as shown in Figure 2.9.

In early 1600s, Galileo published his findings soon after he confirmed that Jupiter's moons were in orbit around Jupiter. **But most astronomers refused to believe that Galileo's discoveries were true attributing it to errors in measurement.**

Figure 2.9: An illustration of a discourse relation from the PDTB where sentence based argument identification is exemplified. Here, *`Arg1`* in represented by the sentence in *italics*, `Arg2` is in **bold** and the discourse connective is underlined.

The reasoning behind this is sound, as the PDTB has few arguments that are subordinate or embedded clauses rather than the main clause in the given sentence. Table 2.2 shows the number of `Arg1`s that were found in more than one discourse relations. Table 2.3 shows the total number repetitions for `Arg1`s. However, this approach, like the head word based argument identification, suffers from defining the exact bounds of an argument and thus is unable to locate argument spans

Table 2.2: Total number of duplicate or embedded `Arg1`s in the training set.

Relation Type	**Count**
Explicit	5,608
Implicit	1,634
Others	343
Total	7,585

Table 2.3: Total count of duplicate or embedded `Arg1`s in the training set.

Relation Type	**Count**
Explicit	10,958
Implicit	7,314
Others	737
Total	19,009

within a relation. This issue of argument span labeling was addressed in (Prasad et al., 2010) via a set of heuristic-based approaches that incorporated various features as well as extraneous knowledge from other overlapping corpora. Later works such as (Ghosh, Johansson, & Tonelli, 2011) also incorporated the use of Conditional Random Fields as a series of decision based chain relying on the Markov Assumption. As before, various linguistic features including those derived from the syntactic and semantic nature of the relations were utilized. A new trend in research was thus started consisting of a pipeline process to discourse parsing whereby the classifiers were connected in a cascading pattern and outputs from a classifier for `Arg2` were fed into a classifier for `Arg1`. Figure 2.10 gives an example of such a pipeline structure.

This allowed for more continuity and direct processing of arguments and exploited their relationship in order to understand the structure of a given relation better. The Conditional Random Fields

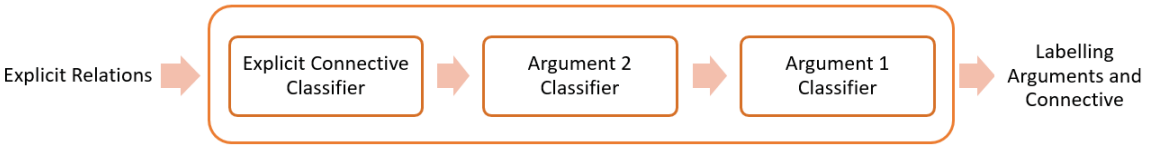

Figure 2.10: An illustration of a pipeline structure for explicit discourse parsing.

were well suited to the pipeline architecture as they rely on decisions made in the past as a feature for making a given classification which in turn is derived from the Markov Assumption (Ghosh et al., 2011). At this point machine learning approaches were part of mainstream Computational Linguistics and techniques such as Support Vector Machines had also been tested with success in this domain (W. Wang, Su, & Tan, 2010).

More recently, in 2014, works from Lin, Ng, and Kan (2014) and Kong, Ng, and Zhou (2014) promoted further interest in this domain and its sub-problems which culminated into the CoNLL 2015 and CoNLL 2016 Shared Tasks (Xue et al., 2015, 2016). The first end-to-end parser (Lin et al., 2014) was developed which combined features and classifiers in a pipeline fashion in order to mimic the entire annotation process of the PDTB. This included the argument labeling classifiers and its features as well. A basic, yet powerful classifier was constructed that improved well upon the previous methods and achieved impressive accuracy of 53.85% (Lin et al., 2014). However, in comparison, this score is still far behind the 90.2% value achieved by humans (Prasad et al., 2008). Also, of note, was another additive technique which employed the linear aspects exploited by Ghosh et al. (2011) as well as utilizing the subtree extraction technique similar to the head words approach by Wellner and Pustejovsky (2007). This approach first classified constituents from a parse tree as possible argument candidates which were then re-labelled to be of `Arg1`, `Arg2` or neither via a linear tagging mechanism. Global information across candidates was then incorporated further to smoothen the final outputs via Integer Linear Programming. This allowed the classifier to incorporate linear constraints on the construction of `Arg1` and `Arg2` constituents. Thus a scenario where consecutive words of a relation were classified alternatively as `Arg1` and `Arg2` was avoided and smoothened to have a chunk of segment of `Arg2` labels followed or preceded (or even encapsulated) by `Arg1` labels.

2.5 CoNLL 2015 and 2016

The sixteenth CoNLL Conference organized in 2015 focused on Discourse Parsing and Argument Segmentation and labeling as the subject of its shared task. In essence, this was designed as a standardized competition geared to push the state of the art for discourse parsing and attract

Table 2.4: Number of instances in the PDTB Dataset

Dataset	Explicit	Non-Explicit	Total
Sections 2-21	15,246	17,289	32,535
Section 22	699	737	1,436
Section 23	923	1,016	1,939
Total	**16,868**	**19,042**	**35,910**

attention and further interest in the topic. The results of the competition proved this to be the case. The CoNLL competition provided the PDTB 2.0 dataset (see Section 2.3) for training and development of the models. As is standard with this dataset, Sections 2-21 of the PDTB were set aside for training only, Section 22 was marked for development and Section 23 for testing of the models. Table 2.4 describes in detail the size of the sections with respect to explicit and non-explicit discourse relations.

In line with standard machine learning tests, the organizers created a separate dataset. At CoNLL 2015, this was composed of English Wikinews accessed on October 22^{nd} 2014. The separate dataset was annotated by two different annotators independently according to the PDTB 2.0 guidelines (Prasad et al., 2007). The inter-annotator agreement between the two was 96% overall indicating a high degree of adherence to the PDTB 2.0 standard. For our purposes, the inter-annotator agreement for explicit argument segments was 89.6% for argument 1, 98.7% for argument 2, and 88.7% for both combined. These scores not only indicate that a pattern exists that is more or less mechanical and readily understood, but also helps to set a theoretical upper limit on how well machine learning algorithms can be expected to perform.

Both the 2015 and 2016 competitions were divided into two streams, the Closed and the Open tracks. The closed track was restricted on the usage of extraneous data beyond the PDTB 2.0 dataset and was only allowed to utilize phrase structure parses predicted via the Berkeley parser (Petrov & Klein, 2007) and dependency parses that were produced by the Stanford parser (Manning et al., 2014) using the former dataset as input. In contrast, the open track was allowed to use any dataset or extraneous knowledge to dynamically build the model or algorithm for the task. For the 2015 challenge, all participants opted for the closed track. For the argument labeling sub task, the rules of the competition imposed strict regulations on the identification of an argument and considered it to

be correctly identified only if it matched its corresponding gold standard argument span exactly with correct labeling (i.e. whether it was `Arg1` or `Arg2`). Failure to match the corresponding argument span exactly was not considered to be a partial match but simply inaccurate. Thus the scoring criteria was binary with no room for a single-character error on the part of the systems. The actual scoring system was based on the F1 score which is defined as the harmonic mean of Precision and Recall. Precision focused on the ratio of true positives over all positive signals emitted by a given system whereas Recall focused on true positives over true positives and false negatives identified by the given system. (For more details on these metrics, see Figure 4.1). The task also required the development of an end-to-end system that attempted full discourse parsing of explicit and implicit relations, segmenting out the arguments as well as the connectives for the explicit relations along with defining the sense of the relation. This concept of an end-to-end discourse parser first originated from the work of (Lin et al., 2014) and consequently participants used this as an inspirational point to derive their own pipeline based architectures. For argument extraction, most participants resorted to casting the problem as a sequence labeling task at the token level while a few teams, in contrast, applied rule based approaches to extract the arguments. For the sequence labeling approach, a popular choice of learning algorithm were the Conditional Random Fields (CRFs). These are types of probabilistic undirected graphical models that are discriminative and rely on the Markov Property to establish a relationship between current and historical inputs and outputs. As a result, they can account for and learn dependencies between words of a relation that appear in a non-consecutive format thereby learning the start and end points of an argument span.

While most models did rely on conventional machine learning techniques, one group from Dublin University (L. Wang et al., 2015) attempted to use neural networks for the argument labeling task. This topic is discussed in Section 2.6 in more detail. The Dublin University team utilized a specific model of the deep neural network family called RNNs. They used token level features such as part of speech (POS) tags extracted from a fixed window of words that is centered on a target word token. This was the first time that Deep Neural Network algorithms were applied to argument extraction. Table 2.5 shows the results of all teams at CoNLL 2015. The accuracy on Explicit argument segmentation task was 26.27% for `Arg1`, 37.33% for `Arg2` and 18.59% for both combined on the development set. To compare, the best results in the competition were offered by

the East China Normal University Team (J. Wang & Lan, 2015) with accuracy values of 80.56% for `Arg2`, 61.56% for `Arg1` and 54.05% for both combined. In general, the parser developed by the East China Normal University Team had an overall F1 score of 24.00%. It can be easily noticed that the RNN model from the Dublin University Group is lacking in terms of performance compared to the model produced by the East China Normal University Team.

Table 2.5: F1 Scores from the CoNLL-2015 task on Explicit Argument Labeling and Connective Identification (Xue et al., 2015).

Rank	Participant	Explicit				
		`Arg1` and `Arg2`	`Arg1`	`Arg2`	Connective	Parser
	Blind Test					
1	University of Texas Dallas	40.04	49.68	70.06	89.90	30.58
2	East China Normal University	41.35	48.31	74.29	91.86	30.38
3	University of Trento	39.59	49.03	70.68	89.92	29.97
4	**Concordia University**	**36.60**	**45.18**	**69.18**	**90.19**	**27.32**
5	Japan Advanced Institute of Science and Tech.	34.23	44.08	51.35	61.66	27.20
6	AU KBC Research Center	34.73	44.49	64.20	84.49	26.73
7	UIUC Cognitive Computing Group	30.05	37.89	60.11	87.98	23.32
8	Soochow University	30.42	36.43	73.04	91.62	22.95
9	Chinese Academy of Sciences	27.20	36.40	61.00	82.60	22.20
10	Nippon Telegraph and Telephone Lab Japan	21.61	28.13	38.02	51.04	16.93
11	Goethe University Frankfurt	19.04	26.41	36.85	51.18	13.51
12	India Institute of Technology	13.65	22.32	61.99	89.30	12.36
13	Dublin City University 1	12.47	18.05	36.65	87.81	9.12
14	Shanghai Jiao Tong University 1	10.55	13.94	48.97	81.68	8.04
15	Dublin City University 2	11.10	16.65	28.13	79.43	7.85
16	Peking University	3.57	6.07	20.89	59.11	2.32
	Standard WSJ Test (Section 23)					
1	East China Normal University	45.20	50.66	77.40	94.21	39.96
2	University of Trento	44.58	50.05	76.23	92.77	39.54
3	University of Texas Dallas	41.57	49.75	68.55	89.33	37.59
4	Nippon Telegraph and Telephone Lab Japan	38.82	46.07	68.38	89.12	34.47
5	Japan Advanced Institute of Science and Tech.	38.16	43.82	56.25	63.89	33.22
6	**Concordia University**	**38.07**	**44.69**	**72.34**	**91.38**	**32.60**
7	UIUC Cognitive Computing Group	30.39	37.25	66.67	91.83	27.02
8	AU KBC Research Center	30.77	36.64	49.68	86.44	26.78
9	Chinese Academy of Sciences	28.70	36.07	63.53	90.64	25.75
10	Soochow University	30.21	34.02	74.48	94.77	25.30
11	Goethe University Frankfurt	25.20	30.79	50.74	68.19	21.89
12	Dublin City University 1	19.36	24.42	46.20	93.18	17.38
13	Dublin City University 2	14.66	21.10	38.20	88.06	13.21
14	India Institute of Technology	13.78	20.34	59.38	93.06	12.90

Table 2.5: (continued)

Rank	Participant	Explicit				
		Arg1 and Arg2	Arg1	Arg2	Connective	Parser
15	Shanghai Jiao Tong University 1	10.29	14.68	48.77	78.67	9.97
16	Peking University	4.28	6.31	24.05	58.04	3.53
	Development					
1	AU KBC Research Center	54.69	62.90	75.91	92.80	49.11
2	East China Normal University	54.05	61.56	80.56	95.14	48.16
3	University of Trento stepanov	57.10	78.70	93.79	46.89	40.08
4	Nippon Telegraph and Telephone Lab Japan	47.90	55.68	72.16	88.94	43.02
5	University of Texas Dallas	48.51	57.46	72.24	93.43	41.49
6	Japan Advanced Institute of Science and Tech.	45.14	51.56	57.79	65.53	41.17
7	**Concordia University**	**45.91**	**53.16**	**75.34**	**92.25**	**39.52**
8	UIUC Cognitive Computing Group	34.78	43.18	65.97	91.45	31.18
9	Chinese Academy of Sciences	33.16	41.71	67.99	91.52	30.25
10	Soochow University	34.67	38.67	74.37	94.22	29.78
11	Goethe University Frankfurt	27.37	33.93	48.32	63.17	23.77
12	Dublin City University 1	20.52	28.55	41.78	93.23	17.70
13	Dublin City University 2	18.59	26.27	37.33	86.33	15.82
14	India Institute of Technology	17.09	25.94	65.52	93.55	15.59
15	Shanghai Jiao Tong University 1	15.15	18.35	58.27	86.09	14.57
16	Peking University	3.14	4.77	19.08	51.54	2.79

2.6 Machine Learning and Deep Learning

Deep Learning is branch of Artificial Neural Networks which in turn is part of the Machine Learning Domain. Machine Learning involves methods that are able to discern patterns from within a given dataset and are able to use these patterns to predict future instances (that are similar in nature to the original dataset) or make decisions under uncertainty (Murphy, 2012). This is done by developing an algorithm that is able to find patterns in a given sample dataset that correlate to specific attributes about the overall population of the data. This can be used to provide the necessary output needed to solve a given problem for the data points existing outside of the sample dataset. A simple example of such an algorithm is linear regression where a straight line going through, or closest to as many points as possible, is able to provide an indication of the trend of the data. Artificial Neural Networks are simply structural combinations of a mathematical expression similar to linear regression for a single dimensional dataset. This involves putting together neurons or

linear-regression-like units together to increase dimensionality in order to accept a data point of multiple dimensions. These layers of neurons can then be stacked in order to decompose data over multiple lower dimensions or to converge it to fewer higher dimensions so that a pattern maybe isolated and recognized between the decomposed or merged portions of a data point. Creating such a feed-forward structure of layers of neurons can pose issues where the number of parameters that need to be learned greatly increase leaving the system with not enough sample data to learn the most optimal values for isolating a pattern (Bengio, Ducharme, Vincent, & Jauvin, 2003). Thus various other architectures exist that build on reusing parameters while keeping a scalable architecture such as Recurrent Neural Network (RNN) where a layer feeds back into itself for a specific dimension of the input data or LSTMs where a learnable parameter is used to decide whether to re-use other parameters to learn other dimensions of a data point as well (Hochreiter & Schmidhuber, 1997). Research in Neural Networks has revealed that stacking such layers in an architecture is much more effective than increasing the number of neurons per layer (Goodfellow, Bulatov, Ibarz, Arnoud, & Shet, 2013). Thus Deep Learning is simply an evolution of Neural Network architectures that attempts to re-use parameters from within the network in order to create a deep stack of layers that is able to achieve effective state-of-the-art results in Machine Learning and Artificial Intelligence tasks (Schmidhuber, 2015).

2.6.1 Neural Networks

Hornik, Stinchcombe, and White (1989) provided mathematical proof that neural networks can serve as "Universal Approximators". Neural networks are able to approximate any function from a one finite-dimensional space to another up to any desired degree of accuracy, provided certain conditions are met. Among many others that govern the problem domain these conditions also include the presence of at least one hidden layer as well as a squashing function for example a sigmoid function to serve as an activation to a neuron. The degree of desired accuracy required is directly proportional to the number of hidden units available. Therefore neural networks are particularly useful in learning features of a given dataset in a supervised learning environment. This is a situation where a neural network system is exposed to every data point in a sample dataset, along with its corresponding labels. This allows the network to "learn" the parameters needed to

approximate a function connecting the data point with its right label. The neural network learns this information via an algorithm known as "backpropagation" (Rumelhart, Hinton, & Williams, 1986). In general the neural network is initialized with random parameters. Then, information first flows from the input data point towards its label. If the correct label is predicted, then nothing is changed. If not, then the parameters are adjusted to according to the error of prediction in cascading fashion from the label to the data point. Then the next data point is considered. The system is left to go through a certain number of such learning cycles in order to get an efficiently generalized learning model established.

2.6.2 Recurrent Neural Networks

Recurrent Neural Network (RNN) are a special architectural form of the regular feed-forward neural networks. They are different in the sense that one dimension of the input data is used as a recurrent loop over the same neuron. For example, consider a stream of sentences of a constant n words as input and "adjective", "adverb" or "neither" labels as output. In this case, the network is expected to predict the correct part of speech (or "neither") label for every word in the given input sentence. Here, a single recurrent neuron could be used to loop over every single word in a given input sentence. Then, via backpropagation, and accounting for the loop, the errors can be adjusted for the predicted labels of all the words in entire sentence. This allows reuse of the parameters in the same neuron for all n words as opposed to forming a feed-forward neural network of n neurons in the first layer. In general, RNNs are better suited for input data that is sequential in nature. The recursion and re-use of parameters also allows to incorporate dependencies over the range of loop. In our example, the RNN will take into account words preceding an "adverb" or an "adjective" in order to find a pattern of either of two occurring.

2.6.3 Long Short-Term Memory

Long Short-Term Memory (LSTM) are a specialized type of "gated" recurrent neuron where the recursion aspect is modeled for learning rather than as a constant. With vanilla RNNs, it was noticed that long term dependencies were hard to model due to the problem of vanishing and exploding gradients (Hochreiter, Bengio, Frasconi, & Schmidhuber, 2001). This occurs when during a backward

propagation through the loops, the actual correction value (due to repeated differentiation) becomes either negligible or very large (close to infinity) resulting in corrections being either over-saturating or under-saturating the parameters. With LSTMs, not only are long-term dependencies accounted for by learnable parameters, but the ability to forget specific dependencies is also learned. For example, if a value in a vector depends on another value encountered two steps before it, and not on the immediately preceding value, then that could be forgotten as it does not constitute a dependency. Thus LSTMs are able to model much longer dependency ranges and are therefore the most suited for our problem of argument labeling and identification.

2.7 Word Embeddings

Word Embeddings are simply vectors that map a word into a latent space. The latent space describes a rich feature set for the word itself. This is usually a multi-dimensional plane where each dimension describes a feature of the overall language (such as grammatical or semantic rules). Thus, a word embedding is a vector of size N where N is the number of dimensions in latent space (Turian, Ratinov, & Bengio, 2010). Conventionally, multiple approaches exist that apply different techniques in order to build a word embedding from a given set of data. A well known approach involves leveraging the occurrence of a word within a document (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990). This approaches suffer from a problem that arises from the language structure itself, where the majority of the most common words (such as "the", and "and") do not convey meaning or semantic relatedness of other words but are rather syntactic artifacts of the language. Thus the co-occurrence of such common words within the same document will have a higher impact than those that convey meaning within the document (Pennington, Socher, & Manning, 2014).

One of the more recent approaches known as "word2vec" (Mikolov et al., 2013) relies on a simple neural network to build a rich set of word embeddings from unsupervised data. This is done by simply demarcating a window of words known as a *context window* and attempting to predict the occurrence of a given word in that context. Thus, a form of neural language model is generated where word embeddings are elementary units within the model. The model can be

leveraged for many purposes. One example is to predict the next word in a sequence of words while another is to be able to find words for a similar concept when provided with a sample, such as the relationship between the word embeddings for the words `King` and `Man` can be leveraged with the word embedding for `Woman` to produce the word embedding for `Queen`. Since word embeddings are vectors, this can be done using basic mathematical operations such as the one shown below:

$$Embedding[\text{“}Queen\text{”}] = Embedding[\text{“}King\text{”}] - Embedding[\text{“}Man\text{”}] + Embedding[\text{“}Woman\text{”}]$$

However, approaches that rely on a context window tend to ignore the information provided by the co-occurrence of words within the training set. This prevents the model from learning from the repetition within the dataset. Another approach known as Global Vectors for Word Representation (GloVe) Embeddings Pennington et al. (2014) attempts to improve upon both models presented earlier. This is done by building a co-occurrence matrix of every word presented within the corpus. The matrix is then used to compute probabilities of occurrence of every word within the context of another word. Finally ratios are built for all probabilities in order to generate a word embedding. Thus, in this way, GloVe Embeddings take advantage of both, the context window as well as the co-occurrence of words provided within a corpus (Pennington et al., 2014). Furthermore, GloVe optimizes over the standard “word2vec” approach thereby reducing the need to re-compute embeddings from scratch when more word tokens are appended to the vocabulary.

2.8 Deep Learning based Argument Labeling

As mentioned in Section 2.4, almost all of the explicit argument segmentation and labeling approaches proposed involve using an algorithmic model that employ various features that exploit the structure of a discourse in order to provide the argument location and span. In the case of rule based approaches, these features directly guide the system in isolating the argument and labeling it, while in the case of classifiers based on machine learning algorithms, the features indirectly assist the algorithm in understanding the argument and its characteristics so that it can be tagged by the algorithms. Most of these features are grounded in linguistic theory, while some are formulated using

more mathematical approaches. In either case, these features are devised by trying to understand the sample dataset in order to elicit its properties that may be generic enough to apply to the entire population of the data. In this case, however, we tend to move away from this approach and attempt to realize the learning process entirely without using any features. The idea here is to simply let the machine learning algorithm extract the features itself from the training dataset and use them over the test set. The learning process is controlled to account for the fact that the features remain generalizable enough to be applicable over the entire population of the dataset.

2.9 Summary

In this chapter, we presented a historical review of the field of discourse relations in general with particular focus to argument labeling. We also gave a brief introduction to machine learning and deep learning approaches that were introduced in the domain by our contributions.

In the next chapter, we will explain our approach to the problem of argument labeling. We will describe our methodology and explain in detail our experimental structure.

Chapter 3

Experimental Methodology

The goal of this thesis is to automatically extract and label discourse arguments without any prior feature engineering. In order to achieve this, we have used an Artificial Neural Network (ANN) model that accomplishes this goal. However, the discerning factor in this endeavor is that the model does not require any prior knowledge about the distribution of the discourse segments or any grammatical or other linguistic properties of the segments in a given text. This chapter describes the model developed for this task, the framework used, as well as the parameters adjusted for, and the dataset used to evaluate the accuracy of the model.

3.1 The LSTM Approach

3.1.1 LSTM

In Section 2.6.3, we gave a brief introduction to LSTMs. Here, we discuss in detail how LSTMs work and how they relate to the problem of argument labeling in explicit discourse relations.

As mentioned in Section 2.6.3, the LSTM is a variation of the RNN which has memory built into the recurrent node of the network. This concept of memory is enhanced by the fact that the LSTM is able to not only learn how to retain information from past tokens of an instance through time but also forget them thereby releasing resources that are not needed for understanding or predicting future tokens in a given instance. Figure 3.1 compares an LSTM and an RNN.

A standard hidden recurrent node of an RNN can be expressed mathematically as:

$$h^t = g(b + Wh^{t-1} + Ux^t)$$

where t is the current time step, W and U are the weight matrices of the hidden node, b is the bias of the hidden node, g is a squashing function (such as $tanh$) and finally h^{t-1} is the output of the hidden recurrent node from the previous time step.

Thus the hidden node of an LSTM, in turn, can be expressed as follows:

$$h^t = g(s^t)q^t$$

where s is the internal state of the LSTM cell, and q is the output gate. The internal state further comprises an input gate, a self loop and a forget gate. The representation of the internal state s can be mathematically expressed as follows:

$$s^t = f^t s^{t-1} + i^t \sigma(b + \sum_j Ux^t + \sum_j Wh^{t-1})$$

where f is the forget gate and i is the input gate. The forget, input and the output gates are simpler units that apply an element-wise non-linearity to affine transformations of inputs and their own outputs from the previous timesteps. The element-wise non-linearity is usually a sigmoidal transformation. These gates are mathematically expressed below:

$$f^t = \sigma(b^f + \sum_j U^f x^t + \sum_j W^f h^{t-1})$$

$$i^t = \sigma(b^i + \sum_j U^i x^t + \sum_j W^i h^{t-1})$$

$$q^t = \sigma(b^q + \sum_j U^q x^t + \sum_j W^q h^{t-1})$$

Similarly to that of an RNN, an LSTM input instance is a sequence of inputs of an arbitrary length. This sequence is considered a sequence in time whereby the input unit at a given position in the instance is provided to the LSTM cell at the timestep equal to the position of the input unit in

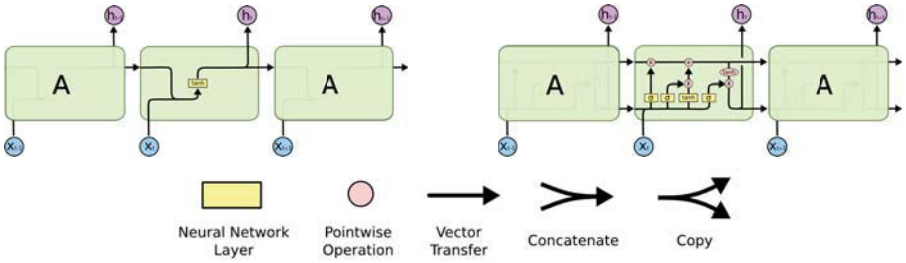

Figure 3.1: A side by side comparison of an RNN (left) and an LSTM (right) architecture (Olah, 2015)

the instance.

At every timestep t, the LSTM cell holds a representation or memory generated from the previous timestep $t-1$. At first, the input at the current timestep t is passed through a non-linear transformation just as it would be in an RNN. Next, the input gate (i^t) in the LSTM cell decides the amount of newly computed input information provided that is incorporated into the newly computed state for time t. Meanwhile, a forget gate (f^t) decides on the amount of information held in the state from the previous timestep $t-1$ that is not necessary and therefore can be forgotten or removed. This is considered as the LSTM's inner self loop. The combination (vector addition) of the newly computed input information and the remaining state information from the previous timestep now become the new state. Finally an element-wise non-linearity (usually $tanh$) is applied to the newly computed state information. Lastly, an output gate (q^t) decides how much of this information is provided as the output at the current timestep. In this way, the LSTM manages to incorporate new information (based on the *input* gate), selectively remembers past information (based on the *forget* gate) and output information relevant only at this timestep (based on the *output* gate). The weights and biases at the input, forget and output gates are learned during training of the neural network.

3.1.2 Bidirectional Long Short-Term Memory

A bidirectional LSTM is a variation of a bidirectional RNN where two separate RNNs of the same node type are fed the same training sequence in opposite orders. The output of both the RNNs are then connected to same output layer or another neural network layer. In the case of a bidirectional LSTM, two LSTM memory cells are passed at each and every training instance at the same time in opposite orders. The forward LSTM cell receives each training instance from

time t_0 to time t_1 and so on whereas the backward LSTM cell receives the same training instance in reverse from time t_n to t_{n-1} and so on. The outputs from the two LSTM cells can then be concatenated. The combined outputs of the two LSTM cells are then passed forward to the next layer in the network. Thus for every point in time t of a given training instance, the overall network has full information of the entire instance as well as the points before and after it. Furthermore, since the LSTM cells allow for prior information to be retained regardless of how far in the past (or future) it is, the network is fully independent to learn the target outputs (Graves & Schmidhuber, 2005). Bidirectional LSTMs have been shown to be successful in many NLP applications such as in phoneme classification (Graves & Schmidhuber, 2005), speech recognition (Graves, Jaitly, & Mohamed, 2013) as well as emotion recognition using speech and facial expressions (Wöllmer, Metallinou, Eyben, Schuller, & Narayanan, 2010).

3.1.3 Batch Training

The usual training process for a neural network involves a forward pass of the inputs to calculate the predicted outputs, a backward pass of the difference between the predicted outputs and the expected outputs (also known as backpropagation of the errors). During the backward pass the weights for the network are adjusted based on the error rate between the predicted and expected outputs. This process of optimization via gradient descent has 3 major variants: Online or Stochastic Gradient Descent, Mini-batch based, and Batch based. In stochastic gradient descent, the gradients are updated at every instance whereas in batch-based gradient descent the errors for every neuron in the network are accumulated and applied at the end of the entire training set. Mini-batch based gradient descent behaves in the same way as batch-based gradient descent, however the batches are much smaller chunks of the entire training set, drawn randomly from the training set. Mini-batches range anywhere between 1 to a few hundred instances (Goodfellow, Bengio, & Courville, 2016). There are multiple advantages of using the mini-batch process including faster convergence than batch based and higher chances of avoiding a local minima than the stochastic approach. In our case, we used a mini-batch size of 128 instances.

3.1.4 Embedding Layer

An embedding layer is simply a neural network layer designed to convert an integer into a dense 2D vector. The integer here represents the position of 1 in a one-hot encoding of a word in the given dataset. A one-hot encoding, in turn, is a simple encoding method where a sparse vector of size $|V|$ is used to represent a word in the vocabulary V of the given dataset. The vector is set to 0 at every point except one where it is set to 1. The index at value 1 represents the position of the word in the dictionary. The embedding layer creates a weight matrix W^E of size $|V| \times D$ where D is the size of the embedding dimension and $|V|$ is the size of the vocabulary. This matrix is learned and optimized throughout the training process of the overall network.

3.1.5 Adam Optimizer

An optimizer is a function used to iteratively traverse the decision boundary of a given objective function and solve for values that define a possible global (but definitely local) optimal point at which the gradient of the decision boundary is given a value (Kingma & Ba, 2014). In simpler terms, an optimizer is simply a function that calculates the new weights based on a given instance (or a mini-batch of instances). These new weights are calculated to perform better on the given instance (or mini-batch) than the previous weights. The Adam Optimizer (Kingma & Ba, 2014) is an algorithm for stochastic optimization of a neural network's loss function that can be significantly faster in some cases than simpler optimizers such as stochastic gradient descent. The Adam Optimizer simply computes the first moment estimate (estimated mean) and the second raw moment estimate (estimated variance about 0). Along with the given learning rates, the first and second raw moment estimates from the given and previous instances are used to compute the new weights. Adam holds in memory a moving average of estimated mean and estimated variance about 0. This allows for a small memory footprint of the Adam Optimizer as well as a higher computational efficiency. At the first weight update process, the moving averages of the first and second raw moments are initialized to 0. This initialization to 0 requires a bias-correction step which results in the higher effectiveness of the Adam optimization method (Ruder, 2016).

3.2 Frameworks and Libraries

Here we provide a brief description of the frameworks and libraries used in our project as well as our rationale for selecting them.

3.2.1 TensorFlow

TensorFlow (Abadi et al., 2015) is an open-source machine learning framework developed at Google. TensorFlow was a significant upgrade over previous machine learning frameworks such as Scipy (E. Jones, Oliphant, & Peterson, 2001) and MALLET (Andrew Kachites McCallum, 2002). Given a mathematical equation, TensorFlow allows the equation to be decomposed into various constituents such that the data provided as input to the equation is made to flow from one part of the equation to another. Intuitively, this is performed by mapping operations of an equation in order of priority to supply outputs to the next operation. For example, an equation $f(x) = ax + b$ can be broken down as $h(g(x))$ where $g(x) = ax$ and $h(g) = g + b$. The input x can then be used to compute the node $g(x)$ and the result of that can be used to compute the node $h(g)$ which results in the value for $f(x)$. The most important advantage of this approach from the perspective of neural networks is that an equation can be broken down into easily differentiable pieces which is important for backpropagation algorithms. Using basic calculus principles, these nodes can be automatically differentiated in order to assist with the backpropagation procedures. This is in addition to other advantages such as allowing for large scale distributed computation of certain nodes as well as the ability to debug and visualize the data resulting from "steps" in a given equation.

While TensorFlow is created as a node computational graph library, its major use is in Deep Learning algorithms. As such, TensorFlow provides tooling as well classes to generate a neural network as well as plug-in pre-written layers. However, since TensorFlow is not a mature library, API stability is an important concern and therefore programming projects using TensorFlow as a library requires consistent maintenance for updates as well as possibility of bugs and unexpected changes in the library code.

3.2.2 Keras

Keras (Chollet, 2015) is a high level library that abstracts away the underlying computational engine and allows for faster and rapid prototyping for proof of concepts as well as larger projects. Since most computational engines for neural networks are new and subject to change often, Keras acts as a middle layer thereby allowing for rapid changes to be matched against a more conformant and backward compatible API. This allows for lesser maintainability than writing projects directly against a computational engine. As a drawback however, newer and cutting edge features and models provided by a computational engine are harder to make use of. Since Keras allows abstraction against the computation engine, it also provides the ability to switch out TensorFlow for Theano (Al-Rfou et al., 2016) or CNTK (Seide & Agarwal, 2016) which are alternatives to TensorFlow as an underlying computational engine. This allows a developer flexibility to switch to the most promising computational engine given a particular use case. Finally, Keras also provides a more simpler API out-of-the-box along with some plug-and-play models, and neural network layers with defaulted parameters based on latest research that allow for even faster prototyping for some specific cases.

For the purposes of this project, the initial work was performed using TensorFlow. However, the final implementations and iterations were switched to use Keras instead.

3.3 Experimental Architecture and Design

This section describes how the experiments were created as well as the construction of the Neural Network used and the data pre-processing step.

3.3.1 Data Pre-processing

For this experiment, we chose to work with the PDTB version 2.0 dataset described earlier in Section 2.3. Due to the CoNLL competitions (discussed in Section 2.5), the PDTB version 2.0 dataset has become a reference point for argument labeling methodologies. Furthermore, we used the corpus in the same way as defined in the CoNLL 2015 and 2016 competitions. Using the corpus in this format for our experiment, allowed us to directly compare the results with other known

techniques and network architectures used for this problem.

Since our work focuses only on explicit relations, we only considered the explicit relations out the entire PDTB dataset. In accordance with the guidelines set out in the CoNLL competitions, Sections 2-21 were used for training and consisted of about 15,246 instances of discourse relations. For testing, Section 22 was used which, in contrast, had about 699 instances. Details of the dataset are shown in Table 2.4.

The PDTB consists of a collection of documents annotated with a `DocId`. Each document itself is composed of an ordered collection of sentences. Each sentence in turn contains `dependencies`, as well as a `parseTree` along with an ordered list of `words`. The list of `words` is further broken down into the actual raw token as well as information about the token itself. This information includes the beginning and ending of the word token's character position in the sentence as well as the part of speech it belongs to. Also included in the `word` is a set of `Linkers` that define which discourse relation, if any, does the word belong to and what constituent of the relation does the word conform. In a nutshell, the structure of the parsed file is shown in Figure 3.2. An example instance of this structure is shown in Appendix A.

Table 3.1: Number of instances in the PDTB Dataset seperated by the position of `Arg1` (Prasad et al., 2008)

`Arg1` Position with respect to the connective	**Count**
Same Sentence	11,236
Immediately Preceding Sentence	5,549
Non-adjacant Preceding Sentence	1,666
Following Sentence(s)	8
Total	**18,459**

As the data was the only input for the neural network, it had to be provided stripped of all attributes in regular text format. While the PDTB provides this as part of the overall dataset, it requires reading and connecting it to the parsed version. This can be an error prone process. Therefore, we decided to directly read the parsed dataset and strip out the annotations as and when necessary. Thus, for the training set, we were able to isolate only the relations themselves. Any intermediate structures that were not an annotated relation were automatically discarded via this processing pipeline. However, for our test set, extra steps had to be taken so that we were also allowing for true negatives in the test set; i.e. the instances that are not true explicit (or implicit) discourse relations (as

annotated in the PDTB). We did this by providing 2 contiguous sentence structures as potential candidates for being an explicit discourse relation. The original raw dataset is simple textual articles so the effect of having 2 contiguous sentences presented as explicit discourse relation candidates would be the same. While there are possible discourse relations that span more than 2 contiguous sentences in the PDTB, they are rare and hence we chose to ignore them. Table 3.1 shows in detail the number of explicit discourse relations in the PDTB separated by the spatial structure of the discourse relations. Furthermore, most participants in the CoNLL 2015 and 2016 competitions (see Section 2.5) use of contiguous sentences to evaluate potential explicit discourse relation candidates as well so our approach attempts to stay consistent with this part.

```
DATASET:= [
  DocId: {
    Sentences: [
      {
        dependencies: { ... }
        parseTree: { ... }
        words: [
          [
            wordToken,
              {
                CharacterOffsetBegin,
                CharacterOffsetEnd,
                Linkers: [ discourseConstituent_discourseId ]
              }
          ]
        ]
      }
    ]
  }
]
```

Figure 3.2: Structure of the overall PDTB dataset

Since the input to our neural network required only the words themselves, we were able to ignore dependency data and parse tree information and instead focus only on the words and sentence structure. For our purposes, we extracted out the `Linkers` values and used them to regenerate a given discourse relation. This had to be done carefully as a given word token could be part of

multiple discourse relations forming different constituents of every single discourse.

As part of the competition, the final output had to be in a specific format. This format required providing the word token index at the level of the document itself for each word token that formed a constituent of an identified discourse relation. Therefore, as part of the input, the index of word tokens within the document was also computed and provided to the neural network. Once the discourse relations were mapped out, it was used to generate the vocabulary of the entire dataset. The vocabulary also contained a special `ZERO_WORD` which was used to pad every relation to a fixed length. This allowed the input data to be uniform throughout the pipeline making it easy to reason about it with respect to matrix computations as well for proper processing through the embedding layer of the network itself. Another vocabulary list was also generated at the document level. This consisted of the token offset for every word with respect to the document itself which was the main attribute required for formatting the output of the network.

The overall vocabulary was used to generate word embeddings that were simply a numeric integer value assigned to every word. These integer values were further used to compute discourse embeddings which, in turn, were vectors of integers where each vector represented a discourse relation. The length of the vectors was set to the length of the longest discourse relation in the dataset. The values of the vectors were the integer values or word embeddings corresponding to the word at the position in the discourse relation. Another corresponding vector was generated for every discourse embedding which represented the discourse constituent labels for every word in the discourse relation itself. Thus the final input data was as shown in Figure 3.3.

This final output was then wrapped into a thin wrapper that provided extra functionalities during training and testing such as the generating random batches of the dataset by shuffling the actual data. Another functionality included the parsing of the inputs and outputs to a CoNLL acceptable format so that the results can be read by the official CoNLL validator and accuracy can be computed at any given point during the training and testing stages.

There were a few caveats relating to this pre-processing that should be taken into account. For consistency and ease of experimentation, the training and test datasets were pre-parsed before the training was done. This was done to make sure that the vocabulary contained all the words present in the entire dataset and so that the neural network was properly initialized to accept the right length

```
INPUT:= [
     [ a, b, c, d, ... , ZERO_WORD, ZERO_WORD, ZERO_WORD ],
     [ ... , ZERO_WORD, ZERO_WORD, ZERO_WORD ],
     ...
],
LABELS:= [
     [ Arg1, Arg1, Connective, Arg2, ... , 0, 0, 0 ],
     [ ... , 0, 0, 0 ],
     ...
]
```

Figure 3.3: Structure of the overall inputs and outputs for the neural network, where `a`, `b`, `c`, `d` are integer values and `Arg1`, `Connective` and `Arg2` are the output labels.

of a discourse relation (which was set to be the length of the longest discourse relation in the training and test sets combined). In order to make the system more robust, one can make the embedding layer be flexible so that it does not require accepting a fixed length of discourse embeddings (but this may be difficult and ultimately slower). Also, the vocabulary can be made to contain a special `UNKNOWN` word token that is assigned to a word that was seen in the test set but not in the training set. However, care must be taken to parse the token back to its correct offset within the document so that the validator provided by CoNLL works correctly on the discourse relations that contain this token.

3.3.2 Network Architecture

The proposed architectures are shown in Figure 3.4. The first network architecture constructed for this problem consists of a fully connected embedding layer followed by a bidirectional LSTM layer that feeds into a fully connected layer which finally results in a softmax output. We call this model `m1`. As mentioned in Section 3.3.2, the first embedding layer is simply a memory efficient dictionary look-up that allows a unique vector to be assigned for every integer that may occur in the dataset. These integers represent the words in the given discourse relation instance. Thus the unique vectors (replaced in place of the integers) representing the words are constructed as part of the network and are therefore learned during the training process. The bidirectional LSTM layer uses the Glorot Uniform technique Glorot and Bengio (2010) for initializing the weights and

contains 100 LSTM cells in each direction that are passed the results from the embedding layer simultaneously. The final connected layer, provides the output as a vector of probability scores over 4 possible labels: `Arg1`, `Arg2`, `connective` or `none` for each input word. The problem is modeled as a sequential labeling task and thus every word in a given statement is labelled as either "Arg1", "Arg2", "Connective", or "None". The network is optimized via the Adam Optimizer (see Section 3.1.5) (Kingma & Ba, 2014). No other supplementary information is used during training and no features are supplied to the model before the training process. The system is able to learn dependencies based on the syntactic structure between the "Connective" and "Arg2" and long-term dependencies for "Arg1".

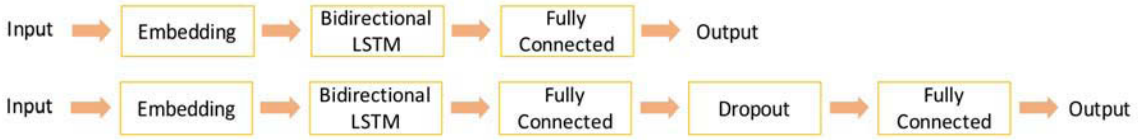

Figure 3.4: Architectures of model `m1` (top) and model `m2` (bottom)

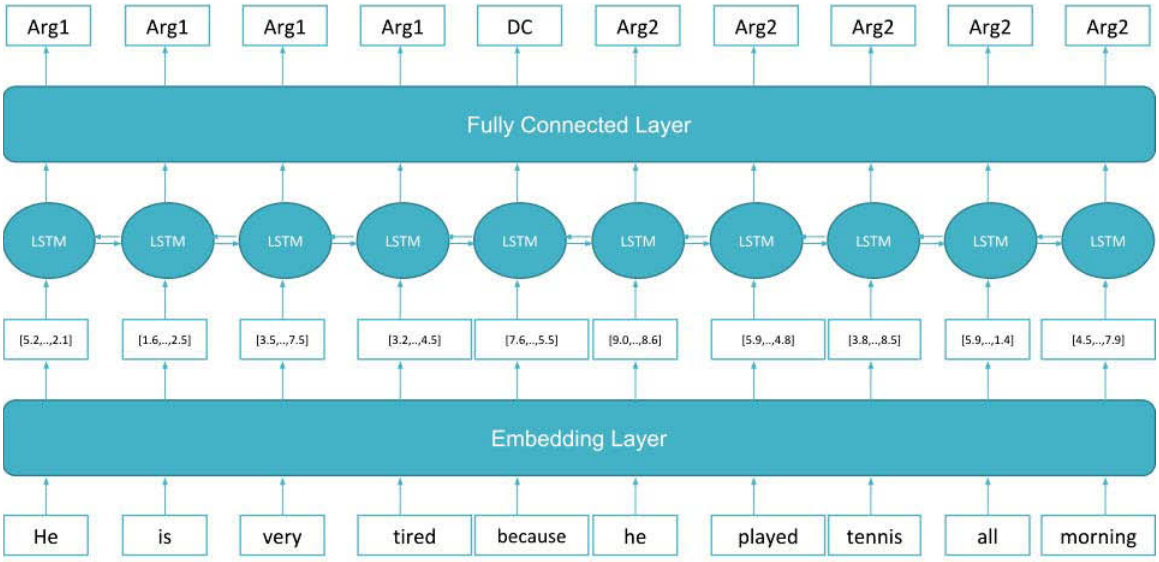

Figure 3.5: An illustration of the architecture using model `m1` as an example

As an example of our architecture, an illustration for the first model is provided in Figure 3.5. As a variation we decided to modify the architecture to include a dropout layer and another fully

connected layer at the end of the original network architecture. We call this next model `m2`. The reasoning behind this was to assist the architecture in differentiating between instances where the same text segments can be part of `Arg1` and `Arg2` for two different discourse relations. As an example, Figure 3.6 shows a pair of discourse relations taken from the training set where the first discourse relation is labelled as `Arg1` for the second discourse relation. Thus the textual span labelled `Arg2` for the first instance is labelled as `Arg1` for the second instance.

(i) *Solo woodwind players have to be creative* **if** they want to work a lot.

(ii) *Solo woodwind players have to be creative if they want to work a lot* **because** their repertoire and audience appeal are limited.

Figure 3.6: An embedded discourse relation pair taken from (Prasad et al., 2008). The discourse relation (i) at the top appears as `Arg1` in the discourse relation (ii) at the bottom.

3.3.3 Loss Function

Kullback Liebler Divergence (KL) (Kullback, 1997) can be used to measure the difference between two separate probability distributions. KL has a value of 0 if the two given probability distributions are equal. Mathematically, KL can be expressed as:

$$D_{KL}(P||Q) = \mathbb{E}_{x\sim P}[logP(x) - logQ(x)]$$

Cross-entropy ($H(P,Q)$) can be defined as a simplification of the KL quantity but without the $logP(x)$ term. This can be mathematically described as:

$$H(P,Q) = -\mathbb{E}_{x\sim P}[logQ(x)]$$

The KL equation presented earlier can be re-written as:

$$D_{KL}(P||Q) = H(P,Q) - H(P)$$

where $H(P)$ can be described as:

$$H(P) = \mathbb{E}_{x \sim P}[logP(x)]$$

We can model our gold standard and the outputs predicted by the neural network, as two independent probability distributions over the input data x. Using the formulations above, we can measure how different these two distributions are in terms of bits. Minimizing this cross-entropy with respect to the predicted probability distribution $Q(x)$ can be used as a way to minimize the differences between the true and predicted distributions, thereby forcing the neural network to converge to the true distribution. Categorical Cross Entropy differs from Binomial Cross Entropy as it encodes probability distribution over more than two labels. As is the case here with multiple labels for the multiple constituents of a discourse relation, categorical cross entropy is well suited to our task. Thus, the true distributions of the labels were modeled as follows:

(1) `Arg1`: $[1, 0, 0, 0]$

(2) `Connective`: $[0, 1, 0, 0]$

(3) `Arg2`: $[0, 0, 1, 0]$

(4) `None`: $[0, 0, 0, 1]$

where `None` was the class where a word belonged to if it was categorized in none of the other labels.

3.3.4 Using Different Word Embeddings

In order to test the effect of extraneous information to the neural network, we also applied a variation of our experiments using pre-computed word embeddings instead of generating random vectors and learning them as part of the network. Consequently, we used GloVe embeddings (Pennington et al., 2014) in one set of experiments and random values in another set. Since we had two different testable network architectures: `m1` and `m2` (see Section 3.3.2), this variation added to our process resulting in 4 independent runs of our experiment:

(1) `m1_random`

(2) `m1_GloVe`

(3) `m2_random`

(4) `m2_GloVe`

3.4 Training Process

This section describes the training process and parameter adjustments performed on the network.

3.4.1 Parameters

For our experimentations, hyper-parameter fine-tuning was performed on a trial and error basis. The available hyper-parameters to tune were the length of the word embeddings, as well as the number of LSTM cells. The highest accuracy achieved by our system on the test set was with 100 LSTM cells and word embeddings length of 300 (see Table 4.1). It is possible that increasing these thresholds leads to a dramatic increase in computational time and a decline in accuracy due to the small size of the dataset which does not contain enough discourse relation instances leading to sparsity in weight adjustment due to the curse of dimensionality (Bengio et al., 2003). Therefore our experiment was limited to the following 4 configurations:

(1) `m1_random`: Embedding Layer of size 1,170 words, Bidirectional LSTM layer of 100 LSTM cells each + randomly generated word embeddings of length 300

(2) `m1_GloVe`: Embedding Layer of size 1,170 words, Bidirectional LSTM layer of 100 LSTM cells each + GloVe based word embeddings of length 300

(3) `m2_random`: Embedding Layer of size 1,170 words, Bidirectional LSTM layers of 100 LSTM cells each + Dense + Dropout + Dense + randomly generated word embeddings of length 300

(4) `m2_GloVe`: Embedding Layer of size 1,170 words, Bidirectional LSTM layers of 100 LSTM cells each + Dense + Dropout + Dense + GloVe based word embeddings of length 300

3.4.2 Training Time

Using preliminary experiments that were run over hundreds of epochs, we established that for our architecture, the training process stabilizes after about 50 epochs and does not improve results further. Thus for our experiments, all the models were learned over 50 epochs. The computations were made on the supercomputer "Helios" from Université Laval, managed by Calcul Québec and Compute Canada. The operation of this supercomputer is funded by the Canada Foundation for Innovation (CFI), the ministère de l'Économie, de la Science et de l'Innovation du Québec (MESI) and the Fonds de Recherche du Québec - Nature et Technologies (FRQ-NT).

The experiments were run using a node consisting of a single K20 GPU with 1 CPU Core. Using the Keras based implementations, the network run takes roughly about 12-13 minutes per epoch to learn. For a 50 epochs run, this is just about 10-11 hours. The time required for data pre-processing in comparison is very small. The entire input dataset is generated in about 5 minutes. As can be expected, the testing process is also significantly faster and takes about 1 to 2 minutes to run. Once trained, the model is saved and can be reloaded for more test runs. Reloading the model itself takes about 2-3 minutes.

3.5 Summary

In this chapter we explained the architecture of our four models, as well as our overall experiments. We described in detail the pre-processing of the dataset from its original state to a form that was provided to the model. We also discussed in detail the loss function provided to the model.

In the next chapter, we discuss and analyze the results of our experiments and compare them with those presented at CoNLL 2015 and 2016.

Chapter 4

Results

This chapter first introduces the evaluation criteria used in the CoNLL competition in Section 4.1, then discusses the results of our experiments detailed in Chapter 3. Finally, in Section 4.5, the results of our approach are compared with that of other research teams at the CoNLL competition.

4.1 Evaluation Criteria

As mentioned in Section 3.3.1, we used the CoNLL works as a baseline for our task. Thus for calculating the accuracy of our model, we also used the official CoNLL scoring module[1].

The CoNLL scoring module was designed to compute the score of argument segmentation by maximizing the alignment of the gold labels (manually annotated) along with the predicted labels. It does this by simply iterating through the set of all predicted spans of `Arg1` and `Arg2` separately and testing for a match against a given respective argument span from the gold label set. A confusion matrix is generated based on the count of correct matches found for a given argument span from the gold label, no match found as well as any predicted spans that could not be matched with any gold relation. Finally, in order to compute the final outputs of the performance metrics of the classifier, the resulting data is plugged in the formulae for Precision, Recall and F1 Score. These formulae are shown in Figure 4.1. The CoNLL Scoring module was also designed to be able to perform

[1] available at https://github.com/attapol/conll16st

$$\text{Precision} = \frac{TruePositives}{TruePositives + FalsePositives}$$

$$\text{Recall} = \frac{TruePositives}{TruePositives + FalseNegatives}$$

$$\text{F1 score} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Figure 4.1: Formulae for Precision, Recall and F1 Score

partial as well as complete matching of discourse labels. In partial matching, an attempt is made to align all possible predicted spans with a given span from the gold standard for `Arg1` and `Arg2` separately and then compute the F1 scores using those alignments. The F1 scores are subjected to a cut-off value which must be met for the labels to be considered partially matched. Partial matching was introduced in CoNLL 2016 as a means of providing diagnostic and debugging information for participating classifiers. It should be noted that in certain conditions, the relation alignment could become a difficult problem to solve computationally. Consequently, this matching scheme was never used for official competition purposes.

For our purposes, we used the exact matching scheme only. The scoring module was, however, slightly adjusted to calculate the performance for explicit discourse relations only. Specifically, the scoring module was adjusted to provide the scores for the exact match for `Arg1` only, `Arg2` only and `Arg1+Arg2`, for every instance in the test set. Connective and sense classification as well implicit relation classification metrics were made to be ignored. It should also be noted that the official scoring module was created to compute precision and recall separately along with the F1 score. However, for our model, we ignored the notion of providing confidence scores and therefore computed a labeling sequence for every test instance provided. Thus the precision, recall and F1 metrics were equal and are referred to here as the accuracy score.

4.2 LSTM based Neural Network

The results of our experiments (described in Chapter 3) are presented here. All four models were made to learn discourse relations labeling over 50 epochs for either configurations. In each

run, the performance was evaluated at the end of every epoch. Figure 4.2 shows the F1 scores for all the models for `Arg1` only, `Arg2` only and `Arg1+Arg2`.

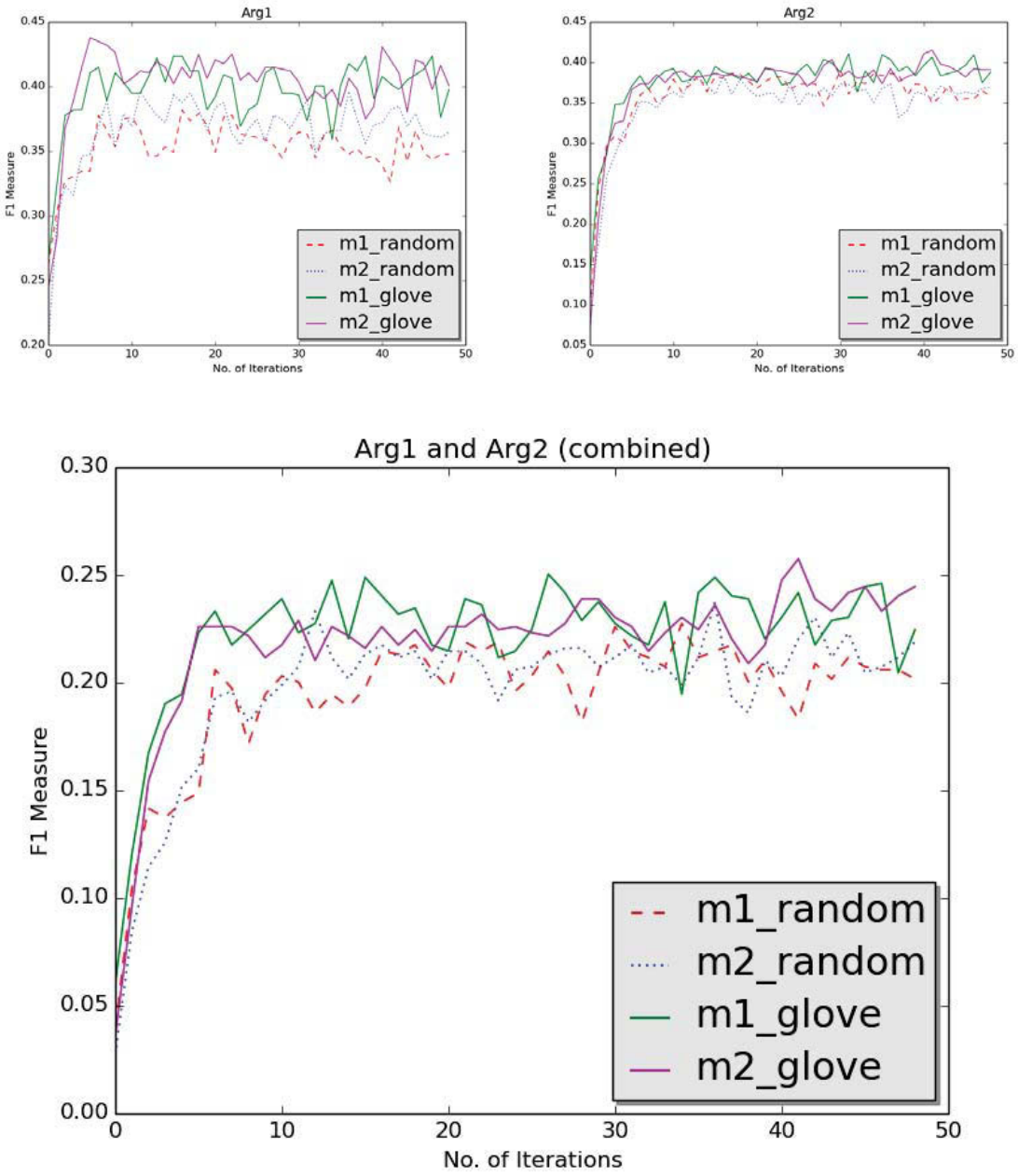

Figure 4.2: F1 score on the test set as a function of the number of iterations on the training set for `Arg1` (top left), `Arg2` (top right) and `Arg1+Arg2` (bottom) (Hooda & Kosseim, 2017)

As can be noticed in Figure 4.2, for all the models, the rate of learning is highest in the first 10 epochs where the accuracy increases significantly after each consecutive epoch. However, beyond

that point, increases in F1 measure are far less pronounced. Running these experiments for longer terms (up to 500 epochs) reveals the exact same trend. The point of saturation is reached early on after which the model stabilizes.

It is however important to note that for almost all the models, the increase in accuracy achieved before the saturation point can vary drastically. This can be noticed much more significantly for `Arg1` but not as much for `Arg1+Arg2` and very minutely for `Arg2`.

For `Arg1` it can be noticed that using the pre-computed GloVe embeddings can have a major improvement in the accuracy of roughly 5-7% just around the saturation point. For `Arg2` a similar conclusion can be drawn based on the top right of Figure 4.2. However, the improvement from using pre-computed GloVe embeddings is not as significant as for `Arg1`. Overall, as well, the improvements from the pre-computed embeddings are closer to that of `Arg2` than `Arg1`. This is most likely due to `Arg2` acting as the limiting factor as it has the lower F1 measures as compared to `Arg1`.

The combined F1 measure for `Arg1+Arg2` is also, significantly lower than either of the constituents separately. Again, this can be explained by the fact that the correctly identified `Arg1` and `Arg2` constituents mostly do not belong to the same discourse relations. In terms of F1 measure, as mentioned earlier, `Arg2` has a slight but significantly lower accuracy when compared with `Arg1`. We postulate that this is an artifact of the algorithm trying to decipher the more complicated segments of `Arg1` and then incorrectly classifying terms (especially those that are segment boundaries) in `Arg2` in the same way as that for `Arg1` which would be an overestimation of the complexity of `Arg2`. However, this favors the correct classification of `Arg1` at the expense of `Arg2`.

Another factor to consider is that the graphs for each of the runs are not smooth and fluctuate significantly (even before reaching the saturation point). This implies that increasing the accuracy based on some samples, results in a significant drop in correct classification of a considerable set of test samples. This implies that there are groups of segmentation mechanics or specific rules that can be used to explain segmentation in groups of discourse relations. However, these rules are either contradictory in different circumstances or require a lot more information to be learned. We hypothesize that the latter is more likely the problem here as our LSTM based neural network is unable to learn this information in a smoother fashion even over time. This hypothesis is reinforced

Table 4.1: F1 scores of our LSTM models for explicit relations compared to the best (hand-crafted) approaches and to (L. Wang et al., 2015)

Model	Arg1+Arg2	Arg1	Arg2	Method
J. Wang and Lan (2016)	55.11%	62.01%	81.26%	Linear classification
Schenk et al. (2016)	54.41%	61.97%	78.87%	CRF
Qin, Zhang, and Zhao (2016)	53.44%	60.99%	79.94%	SVM
Oepen et al. (2016)	51.37%	60.72%	75.83%	SVM
Laali, Cianflone, and Kosseim (2016)	48.67%	56.71%	75.95%	CRF
Kong, Li, Li, Zhu, and Zhou (2016)	46.37%	52.89%	74.81%	MaxEnt
m2_Glove	25.75%	42.06%	41.49%	Bidirectional LSTM
m1_Glove	24.89%	42.35%	39.48%	Bidirectional LSTM
m2_random	23.75%	39.63%	37.34%	Bidirectional LSTM
m1_random	22.75%	36.62%	38.63%	Bidirectional LSTM
L. Wang et al. (2015)	20.52%	28.55%	41.78%	RNN

by the m2_Glove runs which have a visually less erratic and smoother curve than their m1_Glove counterpart. This indicates that adding more neurons as well as Dropout layer (to enforce generalization) will allow learning this information. However, all models still exhibit many discreet peaks and valleys. This points to the lack of richness of the dataset itself with regards to learning without much prior information. This also suggests that a heavily architected rule-based algorithm can be used to emulate this specific dataset (however, that may not generalize over other datasets very well and would therefore be very restrictive in usage).

4.3 CoNLL 2015 and 2016

Table 4.1 shows how our models compare with state of the art systems from the CoNLL 2015 and CoNLL 2016 competitions. As Table 4.1 clearly shows, hand-engineered approaches or approaches that involve manually identifying and implementing features, vastly outperform our LSTM methodology with F1-measures as high as 55.11% for both Arg1+Arg2, 62.01% for Arg1 and 81.26% for Arg2. For Arg2 our models' F1-measures fall to almost half the value (and even lower for some variations) as compared to the state of the art model that uses linear classification (J. Wang & Lan, 2016). However, for Arg1, the LSTM based networks perform fairly well even without GloVe embeddings (in the case of m2_Glove where it is roughly equivalent to about two-thirds the F1-measures when compared to the state of the art model from (J. Wang & Lan, 2016)). This tends to suggest that, given enough data, and a more optimized network architecture,

a generic model can be constructed with acceptable accuracy scores for Arg1 that can be learned without any prior information about the dataset itself.

Also, when compared to (L. Wang et al., 2015), both our models outperform the RNN based approach. While (L. Wang et al., 2015) included hand-engineered features along with skip-gram word embeddings (Mikolov et al., 2013) of size 100, as well as 300 hidden nodes for the RNN layer, our model incorporates GloVe vectors of size 300 and only a 100 nodes at the LSTM layer. Unlike RNN nodes, the LSTM nodes provide the ability to control retaining the information seen earlier in a given discourse relation sequence. This information is stored within each LSTM cell as its internal memory. This seems to show that terms in a discourse relation sequence are unequally important in terms of their relation to the segment boundaries for Arg1 and Arg2. This also shows that retaining information from some terms and ignoring others might be a better optimization strategy when learning segment boundaries for Arg1 and Arg2. Finally, this also hints at the fact that some terms in a given discourse relation that are much further away from the segment boundaries might be more important than those that are closer.

A major dichotomy between the approaches from CoNLL 2015 and 2016 is that almost all of these approaches account for the differences between Arg1 and Arg2 and therefore, build different models to learn the structures of the respective arguments. This involves a comprehensive breakdown of the overall algorithm that require learning either of the boundaries of Arg1 or Arg2 first and then based on that information attempt to choose a strategy that will identify the boundaries of the other argument with respect to information resolved earlier. In all such cases, the primary information that these pipeline-based models depend on is the location of Arg1 which is known to be the hardest of the two to identify (Prasad et al., 2007). In comparison, our approach treats the entire structure as a black box and forces the network to learn the differences and similarities of the two arguments within the same model as opposed to separate ones. This leads to a formulation that would, in general, require significantly more amounts of training data that is not required by the aforementioned approaches. This serves as an important explanation for the differences between our model and those used in CoNLL 2015 and 2016. In essence, we leave it up to our model to understand and learn the differences between Arg1 and Arg2 and neither, rather than build separate models and account for overlaps and / or information produced independently by one of the models.

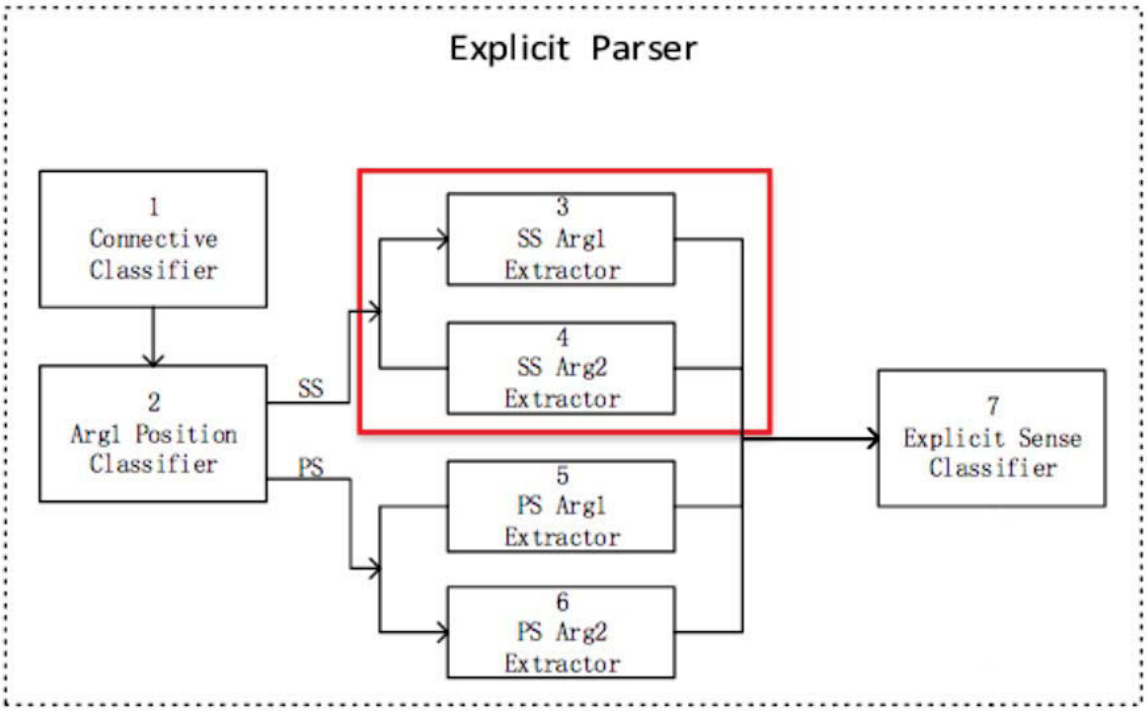

Figure 4.3: The pipeline based architecture used by J. Wang and Lan (2016) at CoNLL 2016. The image shows how the algorithm can be divided into multiple models, the use of which, depends heavily on identifying the location of `Arg1` within a given discourse relation.

It should also be mentioned that while almost all of the approaches seem to outperform our models, these approaches overlap significantly in the features and methodologies used. In fact, the methodology used by (Schenk et al., 2016) is to simply re-use the best models from CoNLL 2015 and output the final result using weighted accuracy estimates from the two models. Such similarities between the approaches given here explain the small space of F1 scores between which most of these approaches exist. Many of these methodologies also disregard one of the three classes for `Arg1` locations rather than include them as part of the learning dataset. Recall from Section 2.3, in a given discourse relation with respect to the discourse connective, an `Arg1` can appear either in the same sentence, preceding sentence(s) or a following sentence (Prasad et al., 2008). Since the PDTB dataset contains only 8 out of 18459 instances where `Arg1` can appear in a sentence following that where the discourse connective exists (see Table 3.1), those instances are disregarded by most systems. Most systems also disregard instances where `Arg1` occurs in non-adjacent preceding sentence(s). This is also justifiable with the same reasoning, as the PDTB dataset only contains

about 1674 of such instances making up about 9% of the PDTB dataset (see Table 3.1) (Prasad et al., 2008). Our system, in contrast, does not take into account the lack of instances for such cases and continues to learn from and identify it, as it would for any other instance in the training and test sets.

Finally, as discussed in Section 3.3.2, some discourse relation instances in the PDTB dataset, are known to be embedded within other discourse relations. The example in Figure 3.6 illustrates this in greater detail. Our models were not specialized to take into account the recursive nature of these discourse relations. While the models presented at CoNLL 2015 and 2016 do not explicitly account for this scenario either, they do attempt to classify a discourse relation by learning the location of `Arg1` within the relation and therefore implicitly account for these cases. In our case, however, this differs as the model has to learn based on word-embeddings (that are either seeded or learned during training) with no features to adjust for recursive cases.

4.4 Distance based F-measure

As mentioned earlier in Section 2.6.3, the LSTM based approach is able to deal with long-term dependencies better than a vanilla RNN based model. To verify this hypothesis, we separated the test dataset by the distance between `Arg1` and `Arg2`. This was done by counting the words between the closest words of `Arg1` and `Arg2` excluding the words labelled as constituents of the discourse connective. For most instances, this meant either counting the word tokens starting from the end of `Arg1` to the start of `Arg2` for relations where the `Arg1` was in a preceding sentence and from the end of `Arg2` to the start of `Arg1` in the case where `Arg1` was in a sentence following the discourse connective. The discourse connective tokens were excluded from this count. Due to the lack of samples in the test sets, we grouped the instances that had more than 10 word tokens between `Arg1` and `Arg2`. This allowed for a more even distribution between the test dataset in terms of distances.

We calculated the distance-based F1 measure for both our models initialized using random vectors that were learned during the training process (`m1_random` and `m2_random`). The calculations were performed at the last epoch of the training process. Figure 4.4 shows the results achieved. It can be easily noticed that the model `m1_random` seems to outperform `m2_random` for `Arg1` and

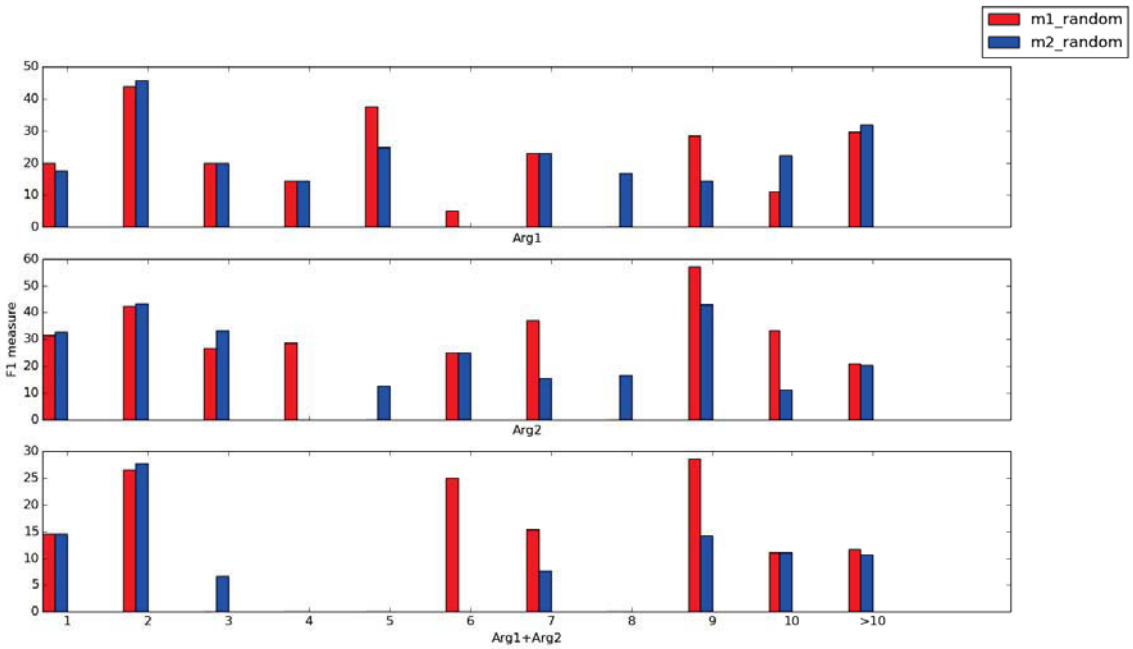

Figure 4.4: Plot of the distance-based F1 scores for `Arg1` (top), `Arg2` (middle) and `Arg1+Arg2` (bottom) (Hooda & Kosseim, 2017)

`Arg2`. However, model `m2_random` still gets better scores for `Arg1+Arg2` for longer distances (greater than 9 word tokens). In the case of both models, however, there is no correlation between the increase in distance and F1 measure that can be abstracted. We can therefore, conclude that these models are unaffected by the increased distances between the `Arg1` and `Arg2`.

As Figure 4.4 shows, there seems to be a significant parity in the F1 scores for most distances between the two models. This represents the restrictive nature of our dataset in terms of its size and quality. The lack of balance between different structures (such as those where `Arg1` appears in a preceding sentence as opposed to the same the next sentence than where the discourse connective is located, as well as the number of relations with a given distance between `Arg1` and `Arg2`) of a discourse relation mentioned in Section 3.3.1 and also the recursive nature within the relations shown in Figure 3.6. Thus increasing the number of learnable nodes in the model does not seem to have much effect in the F1 measures computed over distances between the `Arg1` and `Arg2`

An important phenomenon to note however, is that while the F1-measures do not seem to correlate with the increasing distances, they do however, decrease significantly for Arg2 and consequently Arg1+Arg2 for distances greater than 10. This reinforces the conclusion made earlier in Section 4.2 that the learning capacity of the model is much worse for Arg2 than Arg1. It is, however, interesting to note that between the distances of 2 and 10 words, Arg2 does perform better than Arg1 (especially at distances of 6 and 9). We posit that this is due to the fact that the model's learning capacity with the given dimensionality is limited and therefore, the model is optimized at a local minima that results in the provided output. Consequently, we believe, that with a larger and more balanced dataset, the model would perform significantly better on Arg1 than Arg2.

4.5 Analysis

As mentioned in Chapter 2, in the case of explicit relations, since Arg2 is structurally bound to the discourse connective, locating and marking the boundaries of Arg2 should be much easier once the discourse connective is successfully identified. Contrary to this idea, in our LSTM based approach, the opposite results can be observed. As explained earlier in Section 4.2, this is due to the complexity difference between Arg1 and Arg2. We also believe that a larger dataset would alleviate much of the issues faced here in the correct identification of the argument spans in a given discourse relation.

Another important factor to consider is that unlike the models presented at the CoNLL 2015 and 2016 competitions, our architecture attempts to identify both Arg1 and Arg2 using a single model (instead of separate models for each argument). As a result, due to the structural differences between Arg1 and Arg2, our model does not perform as well as those presented in the CoNLL 2015 and 2016 competitions. This shows that separating the learning for the two arguments maybe necessary for better performance in an LSTM based approach.

We also noticed that the pre-computed embeddings result in a higher accuracy than using randomly generated embeddings that are learned using the training dataset from 22.75% to 24.89% in the case of Model m1 and 23.75% to 25.75% in the case of Model m2. This is likely due to the high sparsity rates of some words present in the dataset. Due to the low of occurrence of these words, the

model is unable to learn the argument labels with respect to those words. As a result, during testing, the model tends to perform poorly with these words in a discourse relation.

As shown in Section 4.4, our model does, however, perform adequately over increasing distances between `Arg1` and `Arg2` for a given discourse relation. Our approach further leads to a promising conclusion that understanding argument segmentation in discourse relations is a task that can be solved without prior knowledge. While, injecting prior knowledge into the network (for example, using pre-computed GloVe embeddings) does indeed boost the accuracy of the model, we believe that it does so only due to the sparsity of some words in the dataset for which the model's learning is augmented by the prior knowledge. Therefore, a more balanced dataset would be helpful to solve this task.

4.6 Summary

In Chapter 4, we have shown the results of our models and compared them with those produced by other models at the CoNLL 2015 and 2016 competitions. We explained the differences in performance based on the dataset, as well as the model architecture. We also provided results based on word token distances between `Arg1` and `Arg2` (discounting the discourse connective) in a given relation. We showed that our approach does indeed work well against increasing distances between the two arguments in a discourse relation due to the internal mechanics of an LSTM. We also argue that the lack of a rich and more balanced dataset hinders the learning of the network.

In the next chapter, we will summarize our findings and discuss the implications and future work that can be done to extend our approach.

Chapter 5

Conclusion and Future Work

Chapter 5 provides a general discussion of the project explained in the previous chapters and focuses on insights gained as well as aspects of the study that warrant further research. We also outline a few possible options that could be used to improve upon our models presented in Chapter 3.

5.1 Advantages of our Approach

Recall that the goal of our research was to segment or identify the boundaries of `Arg1` and `Arg2` using deep learning approaches. To do so, we used an approach based on Bidirectional LSTMs. This model represents a novel approach towards argument labeling using deep learning approaches. However, we do fall short against the bench-marked accuracy scores provided by more conventional approaches. Indeed, our highest F-measure with the CoNLL 2015 and 2016 dataset reaches only 25.75% while others achieve an F-measure of up to 55.11%. Nonetheless, our approach does provide an insight into multiple aspects of argument labeling that were not highlighted before.

Applicability to other Languages Our approach promises a potential generalization of argument labeling over various languages. Since our model does not require hand-crafted features, it can be used to label arguments written in any other language as well. Assuming a possible structure exists between arguments within that language and that this structure loosely follows the shallow discourse structure explained in Section 2.3, then, given enough annotated data to serve as training set, we

expect to see similar accuracy scores as that provided by our system for the PDTB dataset (Prasad et al., 2008).

No Feature Engineering Another major advantage of our approach lies in the fact that no previously understood feature set was required to elicit the accuracy scores that were presented in Section 4.2. The most important disadvantage with feature based engineering is that there is always a danger for over-fitting the model to the dataset. This, in turn, is hard to provide proof for without having access to multiple complementary datasets. Computational linguistics in general and argument labeling in particular are specially prone to this dilemma since, even though the PDTB is complemented by the RST dataset, the two are based on fundamentally different approaches of understand discourse relations and therefore cannot be used to serve as comparison. Furthermore, the RST corpus uses the same underlying text for annotation thereby making the comparison impossible since the feature set that needs to be tested for specificity is still locked to the same domain.

Use of Pre-Trained Embeddings While our model does indeed perform better using pre-generated GloVe embeddings, there are multiple arguments that make the use of pre-generated embeddings more promising than a hard coded feature set. Firstly, the difference in accuracy is not sufficient to justify that the model performs poorly without it (which tends to be the case for many features for a given model). Secondly, the use of pre-generated embeddings is much more benign as it can be generated from raw text via unsupervised learning. This overhead, only adds to the training time for the model since if pre-generated embeddings do not exist, they have to be created before the model can be trained for argument labeling. The data required for generating these embeddings does not rely on a specific domain or any features, but rather only on a large body of provided text and the definition of word tokens which carries over into the model as well. Thus, this improves our confidence in generalizing the model over various datasets in the same language as well as different languages than those with embedded features that assist in de-structuring the given training set.

Easy to Expand Another advantage is that our model is simple and easy to extend. While there are multiple variations that can be built further from it, the original approach provides a simple starting point that can be used to test out multiple theories and architectures. For example, a pipeline like

architecture, reminiscent of conventional models can be applied on top of our approach. Similarly a more parallel graph can be built around it that passes information back and forth (instead of sequentially) to understand an argument using more than just one network. Furthermore, it is also easy to incorporate specific features or knowledge into the model. As an example, consider transfer learning using GloVe embeddings. Another way to provide more features to the model is by simply parsing the training instances using a specific feature function and then replacing the word tokens in the training set in such a way that it directs the model to learn the feature. For example, identifying the discourse connective and replacing it with an artificial word token "DC" would allow the model to rely heavily on the presence of "DC" to identify the start of an `Arg2` (since `Arg2` is structurally bound to the discourse connective. See Section 2.3 for more details).

Better Modeling Finally, our model allows better experimentation and understanding of the anaphoric discourse theory (Webber et al., 2003) utilized in the PDTB dataset. By synthetically altering various factors (such as distance between `Arg2` and the discourse connective, as well as switching the position of `Arg1` before, after or within `Arg2`), one can understand how the model abstracts the general structure of a discourse relation and guide the model towards a better understanding for a given discourse relation instance. This also enforces the idea that the D-LTAG (Polanyi et al., 2004; Webber, 2004) holds merit and a general validation of the framework can be derived independently of human bias. Finally, allowing for generalization over multiple languages, this technique can reveal further insights in discourse theory and help guide understanding towards language structure in general.

5.2 Shortcomings of our Approach

Even though our approach does open up new paradigms for argument labeling methodologies, it does have its shortcomings that must be improved upon.

Lack of data A significant problem with our models, as is the case with most deep learning approaches, is that the lack of rich training data prevents a more efficient optimization of the model which, in turn, would lead to better accuracy scores. As mentioned earlier, the training set used lacks

in size for what would be adequate for a large deep learning model. It also comprises of multiple discourse relations that are contained within other discourse relations making it harder for the model to understand the relationship between the arguments that comprise of multiple discourse relations within a given segment of text. Alternatively, this could also be seen as an area of improvement for our model where understanding of embedded discourse relations is vital. However, for such a requirement, the model needs to be heavily adjusted to account for the possible depth of a nested discourse relation leading towards a more RST (Mann & Thompson, 1988) based approach for discourse relations for which the RST-DT (Carlson et al., 2002) would be more appropriate as a training set.

Low Performance Another major drawback of our approach is the F-measure of the model. It is evident from Section 4.3 that, given the same amount and type of dataset, our model does not produce F-measure values in the same order as those presented at CoNLL 2015 and 2016. This suggests further fine tuning in the model parameters or a complete overhaul of the entire model architecture itself. One could argue, that the number of neurons within the model could be increased to allow for a more generalized learning. However, in that case, various steps must be taken to avoid over-fitting the dataset to the model. Another argument can be that learning the structures for `Arg1` and `Arg2` is not efficient as they are inherently different. When it comes to labeling `Arg2`, the problem is slightly easier as the discourse connective acts as a marker for the start of `Arg2`. However, `Arg1` can be present before or after the `Arg2` making it difficult to locate. Also, given that `Arg1` can be an arbitrarily far away from `Arg2` for a given discourse relation, it would be difficult to properly generate a correct and complete discourse relation as opposed to confusing `Arg1` of a given discourse relation with the `Arg2` of an adjacent discourse relation.

Architectural Constraints Another major challenge that we faced with our model is that, while the `Arg1` and `Arg2` F1-measure values were lower than those architectures presented at CoNLL 2015 and 2016, the `Arg2` F1-measure scores were lower in general when compared to `Arg1` F1-measure scores. This is counter intuitive as theoretically `Arg2` is easier to locate due to its structural attachment to the discourse connective. `Arg1` on the other hand is unbounded and therefore much

harder to locate and thus label. Furthermore, the theory that `Arg2` is easier to locate, is well demonstrated by almost all the projects submitted for the CoNLL 2015 and 2016 competitions. While this hints at the fact that our model has an architectural flaw, much work needs to be done in order to determine a more optimal model that can handle both `Arg1` and `Arg2` appropriately.

No smoothening over labels Another improvement that could be made to the model is the addition of a smoothening technique. It was noticed that in some of the cases where the model learned labeling word tokens as either `Arg1`, `Arg2` or neither, the model was not consistent in mapping out contiguous segments of text as either `Arg1` or `Arg2`. It could be helpful to test the effects of a generalizing rule where, for example, a word token classified as `Arg2` in the middle of 3 word tokens classified as `Arg1` on both sides, is remapped to be labelled as `Arg1` instead. This could improve the accuracy scores and it would be useful to know if a more complicated computation (such as linear programming) could be helpful in boosting the accuracy scores for the model again without providing extraneous information. Such computation could also be mapped as a tertiary neural network that could simply be learned on fake dataset that encodes the smoothening rules that need to be applied.

Test over multiple languages The last improvement that can made on our work is that, since our model is generalized to not require any features and still provide an accuracy score that is not due to chance, it posits a hypothesis that argument labeling in this manner can be generalized over languages other than English. This hypothesis needs further work such as testing it over other languages where datasets of a similar magnitude exists. While such scenarios do present themselves in other languages, experiments need to be performed to confirm or reject this hypothesis. An interesting improvement could be to test on model on a dataset that is induced using PDTB such as the one presented in (Laali & Kosseim, 2017). This will also allow for providing a baseline for how our model behaves on different languages within the same framework as opposed to a discourse relations framework that is not standardized over multiple languages.

5.3 Future Work

In our experiments, we attempted to take advantage of a bidirectional LSTM based model to understand discourse relations and perform argument labeling. However, other approaches can also be applied to test if different architectures could lead to a better result.

As an example, a recursive neural net would be best suited for embedded discourse relations presented in RST-DT (Carlson et al., 2002). However, for the PDTB, the dataset would have to be adjusted so that it fits well within the context of "deep" nesting since the discourse relations in PDTB are shallow. Another possibility could be the use of Gated Recurrent Units (GRUs) in place of LSTM cells in order to test the effectiveness of a memory based network. The GRU holds promise, since it has been known to perform better on smaller datasets (Chung, Gulcehre, Cho, & Bengio, 2014).

Another option could be to take advantage of even simpler Boltzmann machines (Hinton, Sejnowski, et al., 1986) or a Deep Belief Net (Hinton, Osindero, & Teh, 2006). However, due to the complex structures of discourse relations and the significant amount of variations within the relations, this approach is not expected to perform very well. One could also attempt to solve this issue using a convolutional neural network. This might prove useful and even have a better accuracy score for some discourse relations. However, we believe that, this approach would not work very well over discourse relations where `Arg1` is much further away from `Arg2`.

A final approach could be to use attention mechanisms with RNNs. With the recent rise of attention based techniques (Xu et al., 2015) and the ability to allow a network to learn the relevant parts to focus on based on the instance itself, this is a step above the bidirectional LSTM approach presented here. An architecture involving attention could prove useful in improving the accuracy scores much further even for cases where long distances between `Arg1` and `Arg2` could potentially be a concern.

Lastly, in general, the model could further benefit from a more defined technique for optimizing the hyper-parameters of the network. Algorithms such as Grid Search can be implemented in order to boost the efficiency further without involving trial and error based on numerous executions of the learning and testing process. Although, hyper-parameter optimization especially Grid Search can

require the model to be re-run several times and therefore prove to be computationally expensive to perform.

5.4 Summary

In this chapter, we provided a summary of the advantages of our approach, as well as its disadvantages and suggestions for further improvements. In short, we assert that while our approach and its results show promise, there is still much work to be done before we can consider this a viable alternative to feature-based learning for argument labeling in discourse relations.

Appendix A

Parsed Document

This appendix shows an example of a parsed section of the `wsj_0200` document from the PDTB below:


```
{
  "sentences": [
    {
      "dependencies": [
        [
          "prep",
          "attributed-45",
          "In-1"
        ],
        [
          "det",
          "review-5",
          "an-2"
        ],
        [
          "dep",
          "19-4",
          "Oct.-3"
        ],
        [
          "amod",
          "review-5",
          "19-4"
        ],
        [
          "pobj",
          "In-1",
```

```
            "review-5"
          ],
          [
            "prep",
            "review-5",
            "of-6"
          ],
          [
            "det",
            "Misanthrope-9",
            "The-8"
          ],
          [
            "pobj",
            "of-6",
            "Misanthrope-9"
          ],
          [
            "prep",
            "Misanthrope-9",
            "at-11"
          ],
          [
            "poss",
            "Theatre-15",
            "Chicago-12"
          ],
          [
            "possessive",
            "Chicago-12",
            "'s-13"
          ],
          [
            "nn",
            "Theatre-15",
            "Goodman-14"
          ],
          [
            "pobj",
            "at-11",
            "Theatre-15"
          ],
          [
            "nn",
            "Classics-19",
            "Revitalized-18"
```

```
        ],
        [
          "dep",
          "Theatre-15",
          "Classics-19"
        ],
        [
          "dep",
          "Classics-19",
          "Take-20"
        ],
        [
          "det",
          "Stage-22",
          "the-21"
        ],
        [
          "dobj",
          "Take-20",
          "Stage-22"
        ],
        [
          "prep",
          "Take-20",
          "in-23"
        ],
        [
          "nn",
          "City-25",
          "Windy-24"
        ],
        [
          "pobj",
          "in-23",
          "City-25"
        ],
        [
          "dep",
          "Classics-19",
          "Leisure-28"
        ],
        [
          "cc",
          "Leisure-28",
          "&-29"
        ],
```

```
        [
          "conj",
          "Leisure-28",
          "Arts-30"
        ],
        [
          "det",
          "role-34",
          "the-33"
        ],
        [
          "nsubjpass",
          "attributed-45",
          "role-34"
        ],
        [
          "prep",
          "role-34",
          "of-35"
        ],
        [
          "pobj",
          "of-35",
          "Celimene-36"
        ],
        [
          "vmod",
          "Celimene-36",
          "played-38"
        ],
        [
          "prep",
          "played-38",
          "by-39"
        ],
        [
          "nn",
          "Cattrall-41",
          "Kim-40"
        ],
        [
          "pobj",
          "by-39",
          "Cattrall-41"
        ],
        [
```

```
          "auxpass",
          "attributed-45",
          "was-43"
        ],
        [
          "advmod",
          "attributed-45",
          "mistakenly-44"
        ],
        [
          "root",
          "ROOT-0",
          "attributed-45"
        ],
        [
          "prep",
          "attributed-45",
          "to-46"
        ],
        [
          "nn",
          "Haag-48",
          "Christina-47"
        ],
        [
          "pobj",
          "to-46",
          "Haag-48"
        ]
      ],
      "parsetree": "( (S (PP (IN In) (NP (NP (DT an) (ADJP (NNP
         Oct.) (CD 19)) (NN review)) (PP (IN of) (NP (" ") (NP (
         DT The) (NN Misanthrope)) (" ") (PP (IN at) (NP (NP (NP
         (NNP Chicago) (POS 's)) (NNP Goodman) (NNP Theatre)) (
         PRN (-LRB- -LRB-) (NP (NP (" ") (NP (NP (NNP Revitalized
         ) (NNPS Classics)) (VP (VB Take) (NP (DT the) (NNP Stage
         )) (PP (IN in) (NP (NNP Windy) (NNP City))))) (, ,) ("
         ")) (NP (NNP Leisure) (CC &) (NNP Arts))) (-RRB- -RRB-))
         ))))) (, ,) (NP (NP (DT the) (NN role)) (PP (IN of) (NP
          (NP (NNP Celimene)) (, ,) (VP (VBN played) (PP (IN by)
         (NP (NNP Kim) (NNP Cattrall)))) (, ,)))) (VP (VBD was) (
         VP (ADVP (RB mistakenly)) (VBN attributed) (PP (TO to) (
         NP (NNP Christina) (NNP Haag))))) (. .)) )\n",
      "words": [
        [
          "In",
```

```
            {
              "CharacterOffsetBegin": 9,
              "CharacterOffsetEnd": 11,
              "Linkers": [
                "arg1_3173"
              ],
              "PartOfSpeech": "IN"
            }
          ],
          [
            "an",
            {
              "CharacterOffsetBegin": 12,
              "CharacterOffsetEnd": 14,
              "Linkers": [
                "arg1_3173"
              ],
              "PartOfSpeech": "DT"
            }
          ],
          [
            "Oct.",
            {
              "CharacterOffsetBegin": 15,
              "CharacterOffsetEnd": 19,
              "Linkers": [
                "arg1_3173"
              ],
              "PartOfSpeech": "NNP"
            }
          ],
          [
            "19",
            {
              "CharacterOffsetBegin": 20,
              "CharacterOffsetEnd": 22,
              "Linkers": [
                "arg1_3173"
              ],
              "PartOfSpeech": "CD"
            }
          ],
          [
            "review",
            {
              "CharacterOffsetBegin": 23,
```

```
            "CharacterOffsetEnd": 29,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "NN"
          }
        ],
        [
          "of",
          {
            "CharacterOffsetBegin": 30,
            "CharacterOffsetEnd": 32,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "IN"
          }
        ],
        [
          """,
          {
            "CharacterOffsetBegin": 33,
            "CharacterOffsetEnd": 34,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": """
          }
        ],
        [
          "The",
          {
            "CharacterOffsetBegin": 34,
            "CharacterOffsetEnd": 37,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "DT"
          }
        ],
        [
          "Misanthrope",
          {
            "CharacterOffsetBegin": 38,
            "CharacterOffsetEnd": 49,
            "Linkers": [
```

```
              "arg1_3173"
            ],
            "PartOfSpeech": "NN"
          }
        ],
        [
          """,
          {
            "CharacterOffsetBegin": 49,
            "CharacterOffsetEnd": 50,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": """
          }
        ],
        [
          "at",
          {
            "CharacterOffsetBegin": 51,
            "CharacterOffsetEnd": 53,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "IN"
          }
        ],
        [
          "Chicago",
          {
            "CharacterOffsetBegin": 54,
            "CharacterOffsetEnd": 61,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "NNP"
          }
        ],
        [
          "'s",
          {
            "CharacterOffsetBegin": 61,
            "CharacterOffsetEnd": 63,
            "Linkers": [
              "arg1_3173"
            ],
```

```
            "PartOfSpeech": "POS"
          }
        ],
        [
          "Goodman",
          {
            "CharacterOffsetBegin": 64,
            "CharacterOffsetEnd": 71,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "NNP"
          }
        ],
        [
          "Theatre",
          {
            "CharacterOffsetBegin": 72,
            "CharacterOffsetEnd": 79,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "NNP"
          }
        ],
        [
          "-LRB-",
          {
            "CharacterOffsetBegin": 80,
            "CharacterOffsetEnd": 81,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "-LRB-"
          }
        ],
        [
          """,
          {
            "CharacterOffsetBegin": 81,
            "CharacterOffsetEnd": 82,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": """
          }
```

```
        ],
        [
          "Revitalized",
          {
            "CharacterOffsetBegin": 82,
            "CharacterOffsetEnd": 93,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "NNP"
          }
        ],
        [
          "Classics",
          {
            "CharacterOffsetBegin": 94,
            "CharacterOffsetEnd": 102,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "NNPS"
          }
        ],
        [
          "Take",
          {
            "CharacterOffsetBegin": 103,
            "CharacterOffsetEnd": 107,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "VB"
          }
        ],
        [
          "the",
          {
            "CharacterOffsetBegin": 108,
            "CharacterOffsetEnd": 111,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "DT"
          }
        ],
        [
```

```
          "Stage",
          {
            "CharacterOffsetBegin": 112,
            "CharacterOffsetEnd": 117,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "NNP"
          }
        ],
        [
          "in",
          {
            "CharacterOffsetBegin": 118,
            "CharacterOffsetEnd": 120,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "IN"
          }
        ],
        [
          "Windy",
          {
            "CharacterOffsetBegin": 121,
            "CharacterOffsetEnd": 126,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "NNP"
          }
        ],
        [
          "City",
          {
            "CharacterOffsetBegin": 127,
            "CharacterOffsetEnd": 131,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "NNP"
          }
        ],
        [
          ",",
          {
```

```
            "CharacterOffsetBegin": 131,
            "CharacterOffsetEnd": 132,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": ","
          }
        ],
        [
          """,
          {
            "CharacterOffsetBegin": 132,
            "CharacterOffsetEnd": 133,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": """
          }
        ],
        [
          "Leisure",
          {
            "CharacterOffsetBegin": 134,
            "CharacterOffsetEnd": 141,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "NNP"
          }
        ],
        [
          "&",
          {
            "CharacterOffsetBegin": 142,
            "CharacterOffsetEnd": 143,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "CC"
          }
        ],
        [
          "Arts",
          {
            "CharacterOffsetBegin": 144,
            "CharacterOffsetEnd": 148,
```

```
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "NNP"
          }
        ],
        [
          "-RRB-",
          {
            "CharacterOffsetBegin": 148,
            "CharacterOffsetEnd": 149,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "-RRB-"
          }
        ],
        [
          ",",
          {
            "CharacterOffsetBegin": 149,
            "CharacterOffsetEnd": 150,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": ","
          }
        ],
        [
          "the",
          {
            "CharacterOffsetBegin": 151,
            "CharacterOffsetEnd": 154,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "DT"
          }
        ],
        [
          "role",
          {
            "CharacterOffsetBegin": 155,
            "CharacterOffsetEnd": 159,
            "Linkers": [
              "arg1_3173"
```

```
            ],
            "PartOfSpeech": "NN"
          }
        ],
        [
          "of",
          {
            "CharacterOffsetBegin": 160,
            "CharacterOffsetEnd": 162,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "IN"
          }
        ],
        [
          "Celimene",
          {
            "CharacterOffsetBegin": 163,
            "CharacterOffsetEnd": 171,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "NNP"
          }
        ],
        [
          ",",
          {
            "CharacterOffsetBegin": 171,
            "CharacterOffsetEnd": 172,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": ","
          }
        ],
        [
          "played",
          {
            "CharacterOffsetBegin": 173,
            "CharacterOffsetEnd": 179,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "VBN"
```

```
          }
        ],
        [
          "by",
          {
            "CharacterOffsetBegin": 180,
            "CharacterOffsetEnd": 182,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "IN"
          }
        ],
        [
          "Kim",
          {
            "CharacterOffsetBegin": 183,
            "CharacterOffsetEnd": 186,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "NNP"
          }
        ],
        [
          "Cattrall",
          {
            "CharacterOffsetBegin": 187,
            "CharacterOffsetEnd": 195,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "NNP"
          }
        ],
        [
          ",",
          {
            "CharacterOffsetBegin": 195,
            "CharacterOffsetEnd": 196,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": ","
          }
        ],
```

```
        [
          "was",
          {
            "CharacterOffsetBegin": 197,
            "CharacterOffsetEnd": 200,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "VBD"
          }
        ],
        [
          "mistakenly",
          {
            "CharacterOffsetBegin": 201,
            "CharacterOffsetEnd": 211,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "RB"
          }
        ],
        [
          "attributed",
          {
            "CharacterOffsetBegin": 212,
            "CharacterOffsetEnd": 222,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "VBN"
          }
        ],
        [
          "to",
          {
            "CharacterOffsetBegin": 223,
            "CharacterOffsetEnd": 225,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "TO"
          }
        ],
        [
          "Christina",
```

```
          {
            "CharacterOffsetBegin": 226,
            "CharacterOffsetEnd": 235,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "NNP"
          }
        ],
        [
          "Haag",
          {
            "CharacterOffsetBegin": 236,
            "CharacterOffsetEnd": 240,
            "Linkers": [
              "arg1_3173"
            ],
            "PartOfSpeech": "NNP"
          }
        ],
        [
          ".",
          {
            "CharacterOffsetBegin": 240,
            "CharacterOffsetEnd": 241,
            "Linkers": [],
            "PartOfSpeech": "."
          }
        ]
      ]
    },
    {
      "dependencies": [
        [
          "nn",
          "Haag-2",
          "Ms.-1"
        ],
        [
          "nsubj",
          "plays-3",
          "Haag-2"
        ],
        [
          "root",
          "ROOT-0",
```

```
          "plays-3"
        ],
        [
          "dobj",
          "plays-3",
          "Elianti-4"
        ]
      ],
      "parsetree": "( (S (NP (NNP Ms.) (NNP Haag)) (VP (VBZ plays
         ) (NP (NNP Elianti))) (. .)) )\n",
      "words": [
        [
          "Ms.",
          {
            "CharacterOffsetBegin": 242,
            "CharacterOffsetEnd": 245,
            "Linkers": [
              "arg2_3173"
            ],
            "PartOfSpeech": "NNP"
          }
        ],
        [
          "Haag",
          {
            "CharacterOffsetBegin": 246,
            "CharacterOffsetEnd": 250,
            "Linkers": [
              "arg2_3173"
            ],
            "PartOfSpeech": "NNP"
          }
        ],
        [
          "plays",
          {
            "CharacterOffsetBegin": 251,
            "CharacterOffsetEnd": 256,
            "Linkers": [
              "arg2_3173"
            ],
            "PartOfSpeech": "VBZ"
          }
        ],
        [
          "Elianti",
```

```
            {
              "CharacterOffsetBegin": 257,
              "CharacterOffsetEnd": 264,
              "Linkers": [
                "arg2_3173"
              ],
              "PartOfSpeech": "NNP"
            }
          ],
          [
            ".",
            {
              "CharacterOffsetBegin": 264,
              "CharacterOffsetEnd": 265,
              "Linkers": [],
              "PartOfSpeech": "."
            }
          ]
        ]
      }
    ]
}
```

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from http://tensorflow.org/ (Software available from tensorflow.org)

Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., ... Zhang, Y. (2016, May). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, *abs/1605.02688*. Retrieved from http://arxiv.org/abs/1605.02688

Andrew Kachites McCallum. (2002). *MALLET: A Machine Learning for Language Toolkit.* (http://mallet.cs.umass.edu)

Asher, N. (1993). Reference to abstract objects in English: A philosophical semantics for natural language metaphysics. *Studies in Linguistics and Philosophy.*.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, *3*(Feb), 1137–1155.

Carlson, L., & Marcu, D. (2001). Discourse tagging reference manual. *ISI Technical Report ISI-TR-545*, *54*, 56.

Carlson, L., Marcu, D., & Okurowski, M. E. (2003). Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue* (pp. 85–112). Springer.

Carlson, L., Okurowski, M. E., & Marcu, D. (2002). *RST discourse treebank.* Linguistic Data Consortium, University of Pennsylvania, Catalog LDC2002T07.

Chollet, F. (2015). *keras.* https://github.com/fchollet/keras. GitHub.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Nips 2014 workshop on deep learning, december 2014.*

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, *41*(6), 391–407.

Dines, N., Lee, A., Miltsakaki, E., Prasad, R., Joshi, A., & Webber, B. (2005, June). Attribution and the (non-) alignment of syntactic and discourse arguments of connectives. In *Proceedings of the Workshop on Frontiers in Corpus Annotations II: Pie in the Sky* (pp. 29–36). Ann Arbor, MI, USA.

Elwell, R., & Baldridge, J. (2008, August). Discourse connective argument identification with connective specific rankers. In *Proceedings of the IEEE International Conference on Semantic Computing (ICSC 2008)* (pp. 198–205). Santa Clara, CA, USA.

Ghosh, S., Johansson, R., & Tonelli, S. (2011, November). Shallow discourse parsing with conditional random fields. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP 2011).* Chiang Mai, Thailand.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249–256).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. Retrieved from http://goodfeli.github.io/dlbook/ (Book in preparation for MIT Press)

Goodfellow, I., Bulatov, Y., Ibarz, J., Arnoud, S., & Shet, V. (2013). Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*.

Graves, A., Jaitly, N., & Mohamed, A.-r. (2013). Hybrid speech recognition with deep bidirectional lstm. In *Automatic speech recognition and understanding (asru), 2013 ieee workshop on* (pp. 273–278).

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5-6), 602–610.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, *18*(7), 1527–1554.

Hinton, G. E., Sejnowski, T. J., et al. (1986). Learning and relearning in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, *1*, 282–317.

Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. *Science*, *349*(6245), 261–266.

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.* In J. Kolen and S. Kremer, editors, *A field guide to dynamical recurrent neural networks. IEEE Press*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hooda, S., & Kosseim, L. (2017). Argument labeling of explicit discourse relations using lstm neural networks. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, (RANLP 2017)* (pp. 309–315).

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), 359–366.

Jian, P., She, X., Zhang, C., Zhang, P., & Feng, J. (2016, August). Discourse Relation Sense Classification Systems for CoNLL-2016 Shared Task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)* (pp. 158–163). Berlin, Germany.

Jones, E., Oliphant, T., & Peterson, P. (2001). *SciPy: Open source scientific tools for Python.* Retrieved from http://www.scipy.org/

Jones, K. S. (2001). Natural Language Processing: A historical review. *Current Issues in Computational Linguistics*, 2–10.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kong, F., Li, S., Li, J., Zhu, M., & Zhou, G. (2016, August). SoNLP-DP System for ConLL-2016 English Shallow Discourse Parsing. In *Proceedings of the 20th Conference on Computational Natural Language Learning (CoNLL 2016)* (pp. 65–69). Berlin, Germany.

Kong, F., Ng, H. T., & Zhou, G. (2014, October). A Constituent-Based Approach to Argument Labeling with Joint Inference in Discourse Parsing. In *Proceedings of the 2014th Conference on Empirical Methods in Natural Language Processing. (EMNLP 2014)* (pp. 68–77). Doha, Qatar.

Kullback, S. (1997). *Information theory and statistics*. Courier Corporation.

Laali, M., Cianflone, A., & Kosseim, L. (2016, August). The CLaC Discourse Parser at CoNLL-2016. In *Proceedings of the 20th Conference on Computational Natural Language Learning (CoNLL 2016)* (pp. 92–99). Berlin, Germany.

Laali, M., & Kosseim, L. (2017). Improving discourse relation projection to build discourse annotated corpora. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, (RANLP 2017)* (pp. 407–416).

Lin, Z., Ng, H. T., & Kan, M.-Y. (2014). A PDTB-styled end-to-end discourse parser. *Natural Language Engineering*, *20*(02), 151–184.

Mann, W. C., & Thompson, S. A. (1988). Rhetorical Structure Theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, *8*(3), 243–281.

Manning, C. D., & Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT Press.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., & McClosky, D. (2014, June). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations (ACL 2014)* (pp. 55–60). Baltimore, Maryland, USA.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS 2013)* (pp. 3111–3119).

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective. Adaptive Computation and Machine Learning*. MIT press.

Oepen, S., Read, J., Scheffler, T., Sidarenka, U., Stede, M., Velldal, E., & Øvrelid, L. (2016, August). OPT: OsloPotsdamTeesside Pipelining Rules, Rankers, and Classifier Ensembles for Shallow Discourse Parsing. In *Proceedings of the 20th Conference on Computational*

Natural Language Learning (CoNLL 2016) (pp. 20–26). Berlin, Germany.

Olah, C. (2015). *Understanding LSTM Networks* (No. August 25). http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543).

Petrov, S., & Klein, D. (2007, April). Improved Inference for Unlexicalized Parsing. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL 2007)* (Vol. 7, pp. 404–411). Rochester, NY, USA.

Pitler, E., Raghupathy, M., Mehta, H., Nenkova, A., Lee, A., & Joshi, A. K. (2008). Easily identifiable discourse relations. *Technical Reports (CIS)*, 884.

Polanyi, L., Culy, C., Van Den Berg, M., Thione, G. L., & Ahn, D. (2004, April). A rule based approach to discourse parsing. In *Proceedings of the 5th Special Interest Group on Discourse and Dialogue (SIGDial)* (p. 108-117). Boston, MA, USA.

Prasad, R., Dinesh, N., Lee, A., Miltsakaki, E., Robaldo, J. A., Livio, & Webber, B. (2008, May). The Penn Discourse TreeBank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008).* Marrakech, Morocco.

Prasad, R., Joshi, A. K., & Webber, B. L. (2010, May). Exploiting Scope for Shallow Discourse Parsing. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)* (pp. 2076–2083). Valletta, Malta.

Prasad, R., Miltsakaki, E., Dinesh, N., Lee, A., Joshi, A., Robaldo, L., & Webber, B. L. (2007). *The Penn Discourse Treebank 2.0 Annotation manual.* Retrieved from https://www.seas.upenn.edu/~pdtb/PDTBAPI/pdtb-annotation-manual.pdf

Prasad, R., Webber, B., & Joshi, A. (2014). Reflections on the Penn discourse treebank, comparable corpora, and complementary annotation. *Computational Linguistics*, *40*(4), 921-950.

Qin, L., Zhang, Z., & Zhao, H. (2016, August). Shallow discourse parsing using convolutional neural network. In *Proceedings of the 20th Conference on Computational Natural Language Learning (CoNLL 2016)* (pp. 70–77). Berlin, Germany.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533–536.

Schenk, N., Chiarcos, C., Donandt, K., Rönnqvist, S., Stepanov, E. A., & Riccardi, G. (2016, August). Do We Really Need All Those Rich Linguistic Features? A Neural Network-Based Approach to Implicit Sense Labeling. In *Proceedings of the 20th Conference on Computational Natural Language Learning (CoNLL 2016)* (pp. 41–49). Berlin, Germany.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117.

Seide, F., & Agarwal, A. (2016). CNTK: Microsoft's open-source deep-learning toolkit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 2135–2135).

Turian, J., Ratinov, L., & Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics* (pp. 384–394).

Wang, J., & Lan, M. (2015, July). A refined end-to-end discourse parser. In *Proceedings of the 19th Conference on Computational Natural Language Learning (CoNLL 2015)* (pp. 17–24). Beijing, China.

Wang, J., & Lan, M. (2016, August). Two End-to-End Shallow Discourse Parsers for English and Chinese in CoNLL-2016 Shared Task. In *Proceedings of the 20th Conference on Computational Natural Language Learning (CoNLL 2016)* (pp. 33–40). Berlin, Germany.

Wang, L., Hokamp, C., Okita, T., Zhang, X., & Liu, Q. (2015, July). The DCU discourse parser for connective, argument identification and explicit sense classification. In *Proceedings of the 19th Conference on Computational Natural Language Learning (CoNLL 2015)* (p. 89-94). Beijing, China.

Wang, W., Su, J., & Tan, C. L. (2010, July). Kernel based discourse relation recognition with temporal ordering information. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)* (pp. 710–719). Uppsala, Sweden.

Webber, B. (2004). D-LTAG: Extending lexicalized TAG to discourse. *Cognitive Science*, *28*(5), 751–779.

Webber, B., Stone, M., Joshi, A., & Knott, A. (2003). Anaphora and discourse structure. *Computational Linguistics*, *29*(4), 545–587.

Wellner, B., & Pustejovsky, J. (2007, June). Automatically Identifying the Arguments of Discourse Connectives. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)* (pp. 92–101). Prague, Czech Republic.

Wöllmer, M., Metallinou, A., Eyben, F., Schuller, B., & Narayanan, S. (2010). Context-sensitive multimodal emotion recognition from speech and facial expression using bidirectional lstm modeling. In *Proc. interspeech 2010, makuhari, japan* (pp. 2362–2365).

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048–2057).

Xue, N., Ng, H. T., Pradhan, S., Bryant, R. P. C., & Rutherford, A. T. (2015, July). The CoNLL-2015 shared task on shallow discourse parsing. In *Proceedings of the 19th Conference on Computational Natural Language Learning (CoNLL 2015)* (pp. 1–16). Beijing, China.

Xue, N., Ng, H. T., Rutherford, A., Webber, B., Wang, C., & Wang, H. (2016, August). The CoNLL 2016 Shared Task on Multilingual Shallow Discourse Parsing. In *Proceedings of the 20th Conference on Computational Natural Language Learning (CoNLL 2016)* (pp. 1–19). Berlin, Germany.

Zeldes, A. (2016). rstWeb-A Browser-based Annotation Interface for Rhetorical Structure Theory and Discourse Relations. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations* (pp. 1–5).