EVENT STRUCTURE DETECTION
AND REPRESENTATION USING
DEEP LEARNING

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF SCIENCE AND ENGINEERING

2020

By
Kurt Junshean Pelayo Espinosa
School of Computer Science
Contents

Abstract 14

Declaration 15

Copyright 16

Acknowledgements 18

Dedication 20

Publications 21

1 Introduction 22

1.1 Motivation 22

1.2 Research Questions, Hypotheses and Objectives 26

1.3 Contributions and Outline of This Thesis 28

2 Deep Learning for NLP 31

2.1 Overview 31

2.2 Neural Networks: An Overview 33

2.2.1 Representation 34

2.2.2 Learning 39

2.2.3 Training Neural Networks 43
2.3 Neural Network Architectures ........................................... 44
  2.3.1 LSTM ................................................................. 44
  2.3.2 CNN ................................................................. 47
2.4 Modelling Considerations .................................................. 48
  2.4.1 Parameter Initialisation and Pre-trained Embeddings ....... 49
  2.4.2 Activation Functions ............................................... 51
  2.4.3 Hyper-parameter Tuning ........................................... 55
  2.4.4 Training Considerations ............................................ 57
  2.4.5 Evaluation ........................................................... 60
2.5 Conclusion ................................................................. 62

3 Event Structure Detection .................................................. 63
  3.1 Event Structures ........................................................ 63
    3.1.1 What is an Event? .................................................. 63
    3.1.2 Event Structure Representation ............................... 67
  3.2 Event Structure Extraction ............................................ 68
  3.3 Event Detection Approaches .......................................... 72
    3.3.1 Domains and Datasets .......................................... 72
    3.3.2 General Approaches ............................................. 76
    3.3.3 Analysis of the Current Approaches ........................... 78
  3.4 Proposed Event Detection Approach ................................. 80
    3.4.1 Common Model Processes ...................................... 81
    3.4.2 Search-based Model ............................................. 86
    3.4.3 Baseline Model .................................................. 91
  3.5 Conclusion ............................................................. 93

4 Biomedical Event Detection Experiments .................................. 94
  4.1 Pipeline Experiments .................................................. 94

3
5 Sentence Representation

5.1 Overview

5.1.1 Distributed Representations and Distributional Semantics

5.1.2 Sentence Encoding

5.2 Natural Language Inference Tasks

5.2.1 Semantic Similarity Measurement vs Paraphrase Detection

5.2.2 Benchmark Datasets

5.2.3 State-of-the-Art Methods for Measuring Semantic Similarity

5.3 Classes of Methods: A Closer Look

5.3.1 Representation Granularity

5.3.2 NN Architectures’ Complexity

5.3.3 NN Architectures for Semantic Similarity

5.4 Conclusion

6 Sentence Representation Preliminary Experiments

6.1 Experiments Settings

6.1.1 Model Overview

6.1.2 Training Details

6.1.3 Dataset and Evaluation
6.2 Results and Discussion .................................................. 134
6.3 Error Analysis ............................................................. 137
6.4 Discussion ................................................................. 139
6.5 Further Investigation on Noisy Data ................................. 141
   6.5.1 Methodology ......................................................... 142
   6.5.2 Results and Discussion ............................................ 146
   6.5.3 Conclusion .......................................................... 147

7 Event Representation Applied to Biomedical Semantic Similarity Task 149
   7.1 Overview ............................................................... 149
   7.2 Introduction ........................................................... 150
   7.3 Methodology ........................................................... 151
      7.3.1 Event Detection ................................................... 151
      7.3.2 Sentence Representation ......................................... 152
      7.3.3 Semantic Similarity Evaluation ............................... 153
   7.4 Experimental Settings ............................................... 154
      7.4.1 Baseline ........................................................... 156
   7.5 Results and Discussion ............................................... 158
      7.5.1 Baseline Results .................................................. 158
      7.5.2 Event Extraction .................................................. 159
      7.5.3 Semantic Similarity Evaluation ............................... 160
      7.5.4 Analysis .......................................................... 163
   7.6 Conclusion and Future Work ........................................ 169

8 Conclusion 171
   8.1 Evaluation of Research Objectives ................................. 171
   8.2 Future Work .......................................................... 176
A Computation of Partial Derivatives

A.1 Derivative of the cost function $J$ w.r.t the activation function $a$ . . . . . 178
A.2 Derivative of the activation function $a$ . . . . . . . . . . . . . . . . . 179
A.3 Derivative of the cost function $J$ w.r.t the linear function $z$ . . . . . . 180
A.4 Derivative of the linear function $z$ w.r.t $\Theta$ . . . . . . . . . . . . . . . 181
A.5 Derivative of the linear function $z$ w.r.t $b$ . . . . . . . . . . . . . . . . . 181
A.6 Derivative of the cost function $J$ w.r.t the weights $\Theta$ . . . . . . . . . 181
A.7 Derivative of the cost function $J$ w.r.t the bias $b$ . . . . . . . . . . . . . 182

Word Count: 31,576
List of Tables

3.1 Datasets on the general (ACE2005) and biomedical domain (BioNLP-STs) on event detection and their respective details. The BioNLP-ST datasets are sorted in descending order according to the number of event types in each domain. Acronym definitions: Automatic Content Extraction (ACE), Gene Regulatory Ontology (GRO), Cancer Genetics (CG), Pathway Curation (PC), Epigenetics and Post-translational Modification (EPI), Gene Regulation Network (GRN), Infectious Diseases (ID), Genia (GE), Bacteria Biotopes (BB), Entity Relations (REL). 75

3.2 Top 5 systems for each dataset in the general and biomedical domains in terms of F1-score (along with the human baseline performance if available) and their respective approach. 77

4.1 Model performance comparison on the development using Scenario 1. 100
4.2 Model performance comparison on the development using Scenario 2. 100
4.3 Model performance comparison on the development using Scenario 3. 101
4.4 Hyper-parameters used in our experiments. 107
4.5 Comparison of event detection performance. 107
4.6 Comparison on computational efficiency based on the number of classifications. 108
4.7 Comparison among models in nested and overlapping event detection F1 score performance. ............................. 108

4.8 Event detection performance on CG task 2013 development set for SBNN with the top three performing k-values and when k = 1. ...... 108

5.1 NLI tasks and corresponding benchmark datasets ....................... 116

5.2 Details on the benchmark datasets with five (5) datasets from general domain and one (2) datasets from the biomedical/clinical domain. Acronym definitions: Microsoft Paraphrase (MSR Par 2004), Sentences Involving Compositional Knowledge (SICK) 2014, Paraphrase Identification in Twitter (PIT) 2015, Semantic Textual Similarity (STS) 2016, and Intepretable Semantic Textual Similarity (iSTS) 2016, Biomedical Sentence Similarity Estimation (BIOSSES) 2017, Medical Semantic Textual Similarity (MedSTS) 2018. ................................. 119

6.1 Model comparison between MaLSTM and our implementation ...... 133

6.2 Pearson Correlation. Performance comparison across input representations and composition models. Baseline method uses cosine similarity measure to predict similarity between sentences. ................................. 134

6.3 Percentage of Out-of-Embedding-Vocabulary (OOEV) words .......... 134

6.4 Proposed mapping between W-NUT entity types and DBpedia categories 145

6.5 Effect on the F-scores after adding weakly labelled data to sparse entity types ................................................. 147

6.6 Performance of our NER system, akora, compared to other top-ranking W-NUT 2016 systems ............................................. 147

6.7 Named entity recognition performance on the official evaluation data set 148
7.1 Comparison of Pearson correlation scores as reported vs the actual performance on the BIOSSES dataset.

7.2 Number of events predicted at different event detection model thresholds. Legend: $P_2$ number of sentence pairs with both having predicted events, $P_1$ number of sentence pairs with only one of the pair having predicted events, $P_0$ number of sentence pairs with no events predicted, $U_E$ number of unpaired sentences with events predicted, $U_0$ number of unpaired sentences with no events predicted.

7.3 The distribution of the sentence pairs along different gold similarity scores. Legend: $P_n$ number of sentence pairs, the scores are grouped into the following bins: 0 : [0,0.5), 1 : [0.5,1.5), 2 : [1.5,2.5), 3 : [2.5,3.5), 4 : [3.5,4.5].

7.4 The distribution of the sentence pairs along different gold similarity scores after converting the task into a binary classification. Legend: $P_n$ number of sentence pairs, the scores are grouped into the following bins: 0 : [0,2.5), 1 : [2.5,4]

7.5 Pearson correlation scores at different thresholds and different number of bins using different models: Event, BioBERT and BlueBERT. Legend: $P_n$ number of sentence pairs, $Model_2$ Pearson correlation with two bins, $Models$ Pearson correlation with five bins.

7.6 Comparison of the gold (at different bins) and predicted cosine similarity scores of the sentence pairs for the Event model at threshold=0.5. Shown also are the corresponding cosine similarity scores using the BioBERT and BlueBERT models.
List of Figures

1.1 Top: A DAG-structured relation graph (topmost) from the sentence “Looking for mechanisms linking Brn-3a to carcinogenesis, we discuss the role of this transcription factor in influencing Bcl-2/VEGF induction of tumor angiogenesis, ...” from BioNLP’13 CG Shared Task (Pyysalo et al., 2015). Bottom: A pair of overlapping and nested events \((E_2, E_3)\) extracted from the graph with their shared argument event, a flat event \((E_1)\). ................................................. 25

2.1 A schematic diagram of a single-layer perceptron .......................... 35
2.2 Multi-layer perceptron ................................................................. 37
2.3 The cross-entropy loss function in Equation 2.12 ......................... 40
2.4 Sigmoid function .................................................................. 52
2.5 Tanh function ........................................................................ 53
2.6 ReLu function ........................................................................ 55
2.7 Underfitting vs Overfitting (Bias-Variance) trade-off ..................... 58

3.1 An example event structure of type election. .............................. 64
3.2 An example event of type election with structure different from Figure 3.1. ................................................................. 65
3.3 Another example event of type election with structure different from Figure 3.1. ................................................................. 65
3.4 Dependency structure of sentence A. .......................... 66
3.5 Dependency structure of sentence B. .......................... 66
3.6 Event structure of sentence A. ................................. 66
3.7 Event structure of sentence B. ................................. 67
3.8 A tree representation of a flat event structure. .................. 67
3.9 Top: A DAG-structured relation graph (topmost) from the sentence
“Looking for mechanisms linking Brn-3a to carcinogenesis, we discuss the role of this transcription factor in influencing Bcl-2/VEGF induction of tumor angiogenesis,” from BioNLP’13 CG Shared Task (Pyysalo et al., 2015). Bottom: A pair of overlapping and nested events \((E2, E3)\) extracted from the graph with their shared argument event, a flat event \((E1)\). ........................................ 69
3.10 Candidate generation process given pairwise relations ............. 81
3.11 An illustration of the graph created for sub-events with multiple event predictions. Top: The DAG structure relation graph. Bottom: The expanded relation graph considering the number of event predictions for each sub-event argument. ........................................ 85
3.12 An illustration of the search procedure with the sequence of actions taken to construct an event structure. .......................... 87
3.13 The proposed neural model (SBNN) for nested and overlapping event detection ........................................ 89
3.14 Baseline model architecture ........................................ 91
4.1 Abstract schema for the pipeline approach of event extraction ........ 95
4.2 Scenario 2, each module is trained separately on gold data ........... 96
4.3 Scenario 3, each module is trained using predictions from previous module ........................................ 96
4.4 Learning curve analysis on the development set using Scenario 1 with
Threshold = 0.5 ................................................................. 97

4.5 Learning curve analysis on the development set using Scenario 2 with
Threshold = 0.5 ................................................................. 98

4.6 Learning curve analysis on the development set using Scenario 3 with
Threshold = 0.5 ................................................................. 99

4.7 Distribution of event structures according to number of arguments in
the gold training set (gold relations) ....................................... 103

4.8 Distribution of event structures according to number of arguments in
the predicted training set (predicted relations) ......................... 104

4.9 Distribution of event structures according to number of arguments in
the gold development set (gold relations) ................................ 105

4.10 Distribution of event structures according to number of arguments in
the predicted development set (predicted relations) ................. 106

5.1 Sample manual alignment of two sentences according to the schema
proposed by Agirre et al. (2016c): “Trump’s win caused the shares
of the parent company in the UK to drop unexpectedly.” and “The
stocks of the London-based organisation slumped unexpectedly after
the US election.”. Each aligned pair of chunks includes information
on alignment type and score ............................................... 118
5.2 Top 5 performing systems on each dataset (in descending order of F1-score or Pearson correlation). **MSR Par 2004** (Wang et al., 2019; Devlin et al., 2019a; Ji and Eisenstein, 2013; Cheng and Kartsaklis, 2015; Yin et al., 2015); **SICK 2014** (Mueller and Thyagarajan, 2016a; He and Lin, 2016; Hashimoto et al., 2016; Zhou et al., 2016; He et al., 2015); **PIT 2015-PI** (Vosoughi et al., 2016; Eyecioglu and Keller, 2015a,b; Zarrella et al., 2015a; Zhao et al., 2014); **PIT 2015-SS** (Zarrella et al., 2015b,c; Bicici, 2015a,b; Bertero and Fung, 2015); **STS 2016** (Yang et al., 2019; Devlin et al., 2019a; Rychalska et al., 2016; Brychcin and Svoboda, 2016; Afzal et al., 2016); **iSTS 2016** (Banjade et al., 2016a,b,c; Magnolini et al., 2016a,b); **BIOSSES 2017** (Peng et al., 2019a,b; Blagec et al., 2019; Peng et al., 2019c; Chen et al., 2019); **MedSTS 2018** (Peng et al., 2019a,c,b; Lee et al., 2019; Chen et al., 2019).

5.3 Deep learning methods for sentence modelling (top-performing systems highlighted).

5.4 Typical neural network architecture for the semantic similarity measurement task.

6.1 Pearson correlation in STS 2016 evaluation sets (Key: L=LSTM, V=Vector Sum, C=char, W=word2vec, G=GloVe).

6.2 The convolutional neural network for extracting character-level representation of words. (Ma and Hovy, 2016)

6.3 The main architecture of the neural network for named-entity recognition. (Ma and Hovy, 2016)
Abstract

EVENT STRUCTURE DETECTION AND REPRESENTATION USING DEEP LEARNING
Kurt Junshean Pelayo Espinosa
A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy, 2020

As the amount of unstructured text generated every day grows, so does the need to process them automatically. The goal of this thesis is to develop models using deep learning that can automatically detect and generate event structure representations and to investigate their effectiveness as sentence semantic representations. We provide an overview of deep learning for NLP in Chapter 2 as background for subsequent chapters. We define in Chapter 3 the properties of an event structure and present the state-of-the-art approaches for both general and biomedical domains. We then introduce our proposed approach on biomedical event detection. In Chapter 4, we discuss the experiments settings in which we evaluate our neural models and present the results and analyses. We then present in Chapter 5 a review of the approaches in sentence representation in both general and biomedical domains. We discuss in Chapter 6 the results of our preliminary experiments on a state-of-the-art approach in the general domain and examine its limitations. Finally, we investigate the effectiveness of event representations in a biomedical semantic similarity task in Chapter 7.
Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property
and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=487), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University’s policy on presentation of Theses.
Acknowledgements

First of all, I am deeply grateful to my supervisor Sophia for all her guidance, encouragement and support throughout my PhD. I also thank Riza for co-supervising me and providing very helpful feedback throughout.

My special gratitude also to Prof. Makoto Miwa (Miwa-sensei) for sharing his invaluable expertise and patience in those countless hours of discussion. Similarly, I also thank Prof. Junichi Tsujii for hosting me (along with others) on an internship in AIST, Japan. It was a memorable experience, academically and culturally.

I also thank Dr. Tingting Mu and Dr. Angus Roberts for being on my committee and for their constructive comments.

I would not be able to pursue the PhD without the financial support from University of the Philippines so I cannot thank you enough for giving me this opportunity of a lifetime.

To the research fellows at NaCTeM - Chryssa, Piotr, Austin, Axel, Matthew, Sunil, Nhung, Paul; thank you for all the invaluable support and patience with me through all those years. To my PhD mates: Fenia, Mei, Maolin, Yifei, Thy and Hassan; thanks for the time and peer support all throughout.

To my friends who were there to share some pints when I needed a little break: Phil, Nuno, Alessio, Adam, Jake. Of course, to my family who have given me all the support and encouragement even without realising it, thank you!

I also like to thank those people whom I might have missed to acknowledge here
(sorry, my memory sometimes miserably fails me).

Of course, I thank myself (this may sound a little unconventional) for carrying through to the end of this PhD!

And last but not the least, all my gratitude goes to the supreme being for guiding me in the search for the Truth.
Dedication

To Mama and late Papa
Publications


Chapter 1

Introduction

1.1 Motivation

A sentence is defined as a group of words that expresses a complete thought. In other words, a sentence is the realisation of some specific thought. Furthermore, we know that it is possible to generate two different sentences yet expressing the same thought. Consider the following sentence pair (A,B) below from the general domain (Espinosa, 2017):

A: The shares of the company dropped.
B: The organisation’s stocks slumped.

The meaning (or semantics) of the sentences above is the same but neither the words used nor the word order (or syntax) are much the same. If we use a naive model such as bag-of-words to represent each sentence and the count of word occurrence to determine the similarity between sentences, the sentence pair above would have very low similarity score. In fact, only one word ‘The’ is shared by both sentences. However, we understand that the two sentences are semantically similar since the words used in one sentence are synonyms of the words in the other sentence: ‘shares’ vs ‘stocks’, ‘dropped’ vs ‘slumped’, ‘company’ vs ‘organisation’.
Consider another sentence pair from a biomedical domain (Soğancıoğlu et al., 2017):

A: *Only concomitant ablation of ERK1 and ERK2 impairs tumor growth.*

B: *Furthermore, ablation of both Erk1 and Erk2 impaired tumor development, whereas inactivation of either one alone had no effect.*

The sentence pair above is labelled as having the same meaning. Though the two sentences share a number of words, their sentence structure are different. The idea expressed in the phrase “only concomitant” in sentence A is stated differently in sentence B by using “both” and “inactivation of either one alone had no effect”.

The examples we just considered capture some of the inherent difficulties in determining the semantic similarity between sentences. The main question then is, how can we represent the meaning of text that is invariant to its structure? This is the main motivation of this thesis. Humans are quite adept at doing this, however, today’s computers are far from it. Indeed, structure-invariant meaning representation remains an open problem in the field of natural language understanding.

In tackling the meaning representation problem, an unavoidable question presents itself: what constitutes sentence meaning? Perhaps unsurprisingly, due to this subject’s interdisciplinary (Davidson, 1967; Johnson-Laird and Byrne, 2002; Uexkull, 1982; Fillmore, 1975) nature tackled in different areas from philosophy to neuroscience, coming up with a single encompassing definition is nearly impossible. In this work, we pursue an empirical investigation based on the Principle of Compositionality (Frege, 1892) which argues that the meaning of a sentence is a function of the meaning of its constituents (e.g., words, phrases, morphemes).

A naive approach is to represent each constituent usually based on its position in a vocabulary. However, this kind of representation suffers from the curse of dimensionality (Bellman, 1966) and does not consider the syntactic relationship of one constituent
with the others. Recent approaches address these shortcomings using distributed representations (Hinton, 1984). With distributed representations, each constituent’s representation is based on its co-occurrence with other constituents in a specific context and typically represented in a dense \(n\)-dimensional vector. There have been many proposed approaches that estimate these representations such as \textit{word2vec} (Mikolov et al., 2013a), \textit{GloVe} (Pennington et al., 2014) and more recently \textit{BERT} (Devlin et al., 2019b).

While there has been much research into constructing representations for constituents specifically words, representing larger/complex semantic structures such as event structures remain under-explored (Blacoe and Lapata, 2012a; Espinosa, 2017). In this thesis, we investigate how to detect and represent event structures particularly in the biomedical domain. Furthermore, we investigate the effectiveness of event representation as sentence representation to determine semantic similarity between sentences.

We follow the conventional view of an event structure (Ananiadou et al., 2014, 2010; Hogenboom et al., 2011; Wattarujeekrit et al., 2004a) that it is a structured representation composed of a verb (or its nominalised form) (also called a \textit{trigger}) and a set of arguments with associated roles indicating how they participate in the event.

Let us consider an example of an event structure from the BioNLP shared task (Pyysalo et al., 2015). Given the sentence: \textit{Looking for mechanisms linking Brn-3a to carcinogenesis, we discuss the role of this transcription factor in influencing Bcl-2/VEGF induction of tumor angiogenesis.}, Figure 1.1 shows the extracted events in a directed acyclic graph (DAG) structured relation graph. Events are usually defined as follows: \textit{ID} : \textit{trigger} \textit{Role} : \textit{Argument}_1 \textit{Role} : \textit{Argument}_2 \cdots \textit{Role} : \textit{Argument}_n. An example is: \textit{E1} : \textit{angiogenesis AtLoc} : \textit{tumor}.

One property of events that make the modelling and representation complex is that they can appear as nested and overlapping structures in text. In nested events such as \(E_2\)
1.1. MOTIVATION

and $E_3$, some of their arguments are events themselves. To detect these nested events, we need to detect and represent the argument event $E_1$ first. Another characteristic of events is that they can occur as overlapping structures wherein multiple events share the same argument. For example, $E_2$ and $E_3$ contain the same argument $E_1$. In this work, we investigate how to detect and represent nested and overlapping event structures.

Current approaches (Zerva et al., 2017; Miwa and Ananiadou, 2015; Van Landeghem et al., 2013b; Miwa et al., 2013a; Björne and Salakoski, 2018a) on event detection do not explicitly model nested structures in learning. Other approaches (Rao
et al., 2017; Riedel and McCallum, 2011b; Vlachos and Craven, 2012; Venugopal et al., 2014a) focus on finding relation graphs and detect events with rules. McClosky et al. (2011) treats events as dependency structures by constraining event structures to trees, thus their method cannot represent overlapping event structures. Other event detection approaches are in the general domain (Feng et al., 2016; Nguyen and Grishman, 2015; Chen et al., 2015; Nguyen et al., 2016) but the ACE2005 corpus does not contain nested events (Miwa et al., 2014; Liu et al., 2019), therefore, these approaches cannot handle nested and overlapping event structures.

In summary, the research goal of this thesis is to develop learning models that can automatically detect biomedical events with nested and overlapping structures. Furthermore, we investigate the effectiveness of event-based sentence representations in semantic similarity task. Concretely, we focus on the use of deep learning models over feature-based methods to detect nested and overlapping structures as well as generate semantic representations from these structures in text. Methods that rely on hand-crafted features require comprehensive domain knowledge and thus take time to develop. Thus, as a result these feature-based methods are more difficult to adapt to new domains.

In the next sections, we formulate the research questions, hypotheses and objectives (1.2) and outline the main contributions of this thesis (1.3).

## 1.2 Research Questions, Hypotheses and Objectives

To address the gaps mentioned in the previous section, we have raised the following research questions:

$Q_1$: What is the state-of-the-art in event detection and its limitations?

$Q_2$: Is it possible to develop a deep learning-based method for biomedical event structure detection and representation without relying on hand-crafted features?
1.2. RESEARCH QUESTIONS, HYPOTHESES AND OBJECTIVES

Q3: What is the state-of-the-art in sentence representation and its limitations?

Q4: Can event structure representation be used as sentence representation on biomedical semantic textual similarity task and to what extent is it effective?

Guided by the research questions, we formulate the following research hypotheses:

H1: Existing methods in event detection mostly require hand-crafted features and do not consider nested and overlapping event structures simultaneously during learning.

H2: It is possible to develop a deep learning method without reliance on hand-crafted features in order to detect biomedical nested and overlapping event structures.

H3: Existing methods in text semantics representation can be described in a framework which considers the input representation granularity and composition model complexity and this framework can point out the areas that are under-explored.

H4: Event representations can be used as sentence representations in biomedical semantic similarity task.

Founded on these research questions and hypotheses, we establish the following research objectives:

O1: To conduct a review of existing work on event detection covering general and biomedical domains and analyse the limitations of the state-of-the-art

O2: To develop a neural-based method for detecting nested and overlapping events in the biomedical domain

O3: To conduct a review of existing work on text semantic representation in both general and biomedical domain and classify them into a framework that considers input granularity and model complexity and then investigate the limitations of the state-of-the-art

O4: To generate sentence representation from event representations and evaluate them on semantic textual similarity task in the biomedical domain
1.3 Contributions and Outline of This Thesis

In this section, we summarise the contributions we make in this thesis.

Chapter 2 is an introductory chapter about neural networks as applied to natural language processing (NLP). We describe the representation and learning aspects of neural networks. Furthermore, we describe the common neural network architectures which are considered building blocks in many bigger architectures used across the NLP community. Lastly, we consider the various modelling aspects when using neural networks.

Summary Chapter 3: Event Structure Detection

We first define event structures and its properties. We also discuss how to represent events. We then conduct a survey of the different approaches on event extraction, particularly event detection. We examine the various datasets in different domains and analyse the state-of-the-art approaches in each and discuss the limitations. In this chapter, we also introduce our event detection approach. Concretely, we describe how we detect event structures from given input relations considering their nested and overlapping properties. For the baseline model, we present a neural Tree-LSTM model to classify event structures in a bottom-up manner. This model enumerates exhaustively all the possible event structures that can be formed from input relations. As this can lead to computational efficiency problems, we propose a search-based method that constructs event structures by performing actions at each time step and using a neural network to score each action.

Summary Chapter 4: Biomedical Event Detection Experiments

In this chapter, we discuss the experiments using the models presented in the previous chapter. We evaluate the models in two settings, namely, as part of a pipeline approach
on event extraction and separately as an event detection model against the state-of-the-art event detection approach. We present the experiments and analyses on event structure detection in the biomedical domain. We find that the proposed search-based method performs better than the state-of-the-art event detection method in both evaluation settings. In particular, it yields higher F1-score in event detection, requires less computational cost and does not use any hand-crafted features.

**Summary Chapter 5: Sentence Representation**

We first conduct a survey of the different approaches on sentence modelling in an effort to determine the research gaps. We propose a framework that categorises each method along two dimensions: input representation granularity and model compositional complexity. Our framework shows that there are under-explored input representation granularities such as event structures. We observe that most of the approaches rely on word representations and less research has been conducted on how to construct representations from higher level semantic structures such as events.

**Summary Chapter 6: Sentence Representation Preliminary Experiments**

We replicate a state-of-the-art model for semantic similarity task in the general domain. By conducting experiments with a state-of-the-art model, we discover some of its limitations. Furthermore, we investigate the effect of noisy data in a named-entity recognition task and propose a method to address it.

**Summary Chapter 7: Event Representation Applied to Biomedical Semantic Similarity Task**

We investigate in this chapter if sentence representations based on events can be effective in semantic similarity tasks and to what extent. Since there can be multiple events predicted in a sentence, we describe a simple method to generate a sentence
representation and evaluate it on the semantic similarity task. While there are a num-
ber of limitations and shortcomings in applying the event representations to semantic
similarity, results indicate that they can be used as sentence representations for textual
semantic similarity task. To the best of our knowledge, our work is the first one that
applies event representations to semantic similarity tasks. We conclude by enumerat-
ing potential avenues for extending this work.

In the next chapter, we provide the necessary mathematical background on deep
learning as well as training and modelling considerations when using this method.
Chapter 2

Deep Learning for NLP

In this chapter we will introduce the background concepts on deep learning for natural language processing (NLP) which will be useful in understanding the succeeding chapters of this thesis. We shall present an overview of neural networks by discussing its representation and learning aspects. We shall also discuss some common neural network architectures used in the literature. Lastly, we will examine the different modelling considerations when using neural networks.

2.1 Overview

Broadly, the term deep learning refers to training neural network, sometimes very large neural networks. Neural networks belong to the class of supervised learning algorithms, where we have some input $x$ and we want to learn a function mapping to some output $y$. While neural networks are an old algorithm, it has not gained much attention until recently in the age of big data and increased computational capabilities in terms of hardware. All these developments have given rise naturally to algorithmic innovations that greatly impact the scale of computation. The explosive growth of data, the availability of computational infrastructure supporting large scale computations of
efficient algorithms, contribute to the ever-increasing possibilities of applying deep learning to different applications.

A natural question to ask is: what is the advantage of neural networks over other classifiers or algorithms? In contrast to other supervised learning algorithms, an important characteristic of neural networks is their ability to learn complex non-linear hypotheses $h(x)$ that map input $x$ to output $y$ very well given enough data $m$, even when the input feature space $n$ is large.

Let us consider text classification which is one of the main tasks in NLP. To create a text classifier, one can design features based on bag-of-words model using $n$-grams which are sequences of $n$ characters. Given the English alphabet with 26 letters and limiting the longest n-gram length to just 10, the total number of $n$-gram features where $n = 1, 2, 3, \ldots, 10$ would already be $\sum_{i=1}^{10} 26^i$ which is equal to $1.47 \times 10^{14}$ features. This is a massive number and many of those n-grams do not correspond to words in the English language.

This explosion of feature space also occurs in computer vision. To illustrate the point, given a $50 \times 50$ pixel image where each pixel intensity $x$ is represented as an integer between 0 and 255, we find that even by merely using the quadratic features $(x_i \times x_j)$ as input, the number of features would already reach around 3 million.

Simple logistic regression is simply not the most efficient way to learn a complex non-linear hypothesis as it requires the creation and representation of many features. A logistic regression classifier does not have the capacity to learn a complex hypothesis when the input feature space is large due to the linearity of its decision boundaries. It turns out neural networks are able to learn complex non-linear hypothesis even when the input feature space $n$ is large.

Furthermore, since the neural networks are able to learn new features, the need for extensive feature engineering is lessened or not needed at all. This makes the model derived from neural networks domain-independent and therefore applicable to
new datasets more easily. In practice, what needs to be changed are only the input representations (such as word embeddings) that are pre-trained on the domain.

Neural networks have become the state-of-the-art method in many tasks such as speech recognition (Hinton et al., 2012; Graves et al., 2013) and computer vision (Krizhevsky et al., 2012). Furthermore, neural networks have also performed similarly well in many natural language understanding tasks (Wang et al., 2018a) at par with human-level performance, although, in some tasks the state-of-the-art performance is still below the human performance, indicating that much work are still needed in this area.

2.2 Neural Networks: An Overview

Neural networks were originally motivated by the goal of having machines that can mimic the brain. Roe et al. (1992) posed the so-called “one learning algorithm” hypothesis which states that the human brain uses only a single learning algorithm to do different functions such as sight, hearing, speech, etc. Roe et al. (1992) performed neural rewiring experiments by cutting the wire that connects the ear to the auditory cortex, which is the part of the brain that allows us to hear, and rewire the signal from the eyes such that it gets routed to the auditory cortex. It turned that the auditory cortex learned to see. M´etin and Frost (1989) also performed a similar experiment by rewiring the signals from the eyes to the somatosensory cortex, the part of the brain responsible for processing the sense of touch and it turned out that the brain tissue learned to see.

These neural rewiring experiments have thus provided some evidence that if the same piece of physical brain tissue can process sight, sound or touch, then perhaps there is one learning algorithm that can process sight, sound or touch. This would mean that instead of needing to implement thousands of different programs or algorithms, we might just need to figure out some approximation of the learning algorithm of the
brain and this algorithm will learn to process different kinds of data (Marblestone et al., 2016).

While there have been significant advances in neural networks, the goal of creating this “one learning algorithm” has proved to be challenging and still remains an open problem.

In the following sections, we present the standard approximation of the representation and the learning algorithm used in neural networks as inspired by how the neurons in the brain work.

## 2.2.1 Representation

In this section, we will discuss the issues of representation when using neural networks to model our research hypotheses.

Neural networks were developed simulating neurons or network of neurons in the brain. McCulloch and Pitts (1943) drew an analogy between biological neurons and simple logic gates with binary outputs. A simplified model of a biological neuron was then formalised by Rosenblatt (1957) where he proposed the perceptron. The perceptron is the simplest kind of neural network. It is also referred to as a single-layer perceptron network as it consists of a single layer of output nodes. The inputs are fed directly to the outputs via a series of weights.

Figure 2.1 shows a schematic diagram of the perceptron model. The perceptron belongs to the class of supervised learning algorithms for binary classification which maps input $x$ to a binary output value $y$. Formally, let $x = [x_1, \ldots, x_m] \in \mathbb{R}^d$, $b \in \mathbb{R}$ the bias\(^1\) (shown in Figure 2.1 with value $b = 1$ for simplicity), $w \in \mathbb{R}^d$ be the parameters (also called weights) of the model, and $g(z)$ as the activation function. We compute $z$ as the linear combination of the input values $x$ and weights $w$ as shown in Equation

\[^1\text{The bias is the y-intercept value of the linear function and serves to shift the decision boundary from the origin.}\]
2.1. In Equation 2.2 we apply the activation function \( g(z) \) on the linear combination and this gives us the output \( y \) (Equation 2.3) or the hypothesis \( h_w(x) \).

\[
z = w_1x_1 + \cdots + w_mx_m = \sum_{i=1}^{m} w_ix_i = w^T x
\]  

(2.1)

\[
g(z) = \begin{cases} 
1 & \text{if } z \geq 0 \\
-1 & \text{otherwise} 
\end{cases}
\]  

(2.2)

\[
y = h_w(x) = g(z)
\]  

(2.3)

Since the perceptron uses the threshold step function \( g(z) \) as the activation function, it is not differentiable which means that \( g'(z) \) does not exist at zero and is equal to zero elsewhere. Therefore, the perceptron cannot use the gradient descent algorithm...
which is an optimisation algorithm for finding the minimum of a function based on the gradients or derivatives.

To address this problem, Widrow (1960) proposed a linear activation function called ADALINE (Adaptive Linear Neuron). The linear activation function is a continuous function and therefore it is differentiable. And being a differentiable function, the gradient descent can be used to update the weights in the network. Widrow (1960) also proposed the delta rule which is a gradient descent learning rule for updating the weights of the inputs in neural networks. The delta rule is a special case of the backpropagation algorithm which we will discuss in more detail in the next section. We will also discuss in §2.4.2 the different activation functions used in neural networks.

The main limitation of the single-layer perceptron is that it was only capable of learning linearly separable patterns. Minsky (1969) showed that it was impossible for the single-layer perceptron network to learn an XOR function. However, Minsky (1969) already knew that this is not true for multi-layer perceptron, which we discuss next.

A multi-layer perceptron (MLP) is a natural generalisation of the single-layer perceptron. A MLP contains at least three layers, namely, the input layer, the hidden layer and the output layer. Figure 2.2 shows a MLP with one hidden layer but there could be many hidden layers adding to the complexity of the network. A MLP is usually referred to as “vanilla” neural networks (Hastie et al., 2009). Hence, in the following paragraphs we use the terms interchangeably.

We now examine the mathematical definition to compute the hypothesis, $h_\Theta(x)$, used by the neural networks in Figure 2.2. Note that from hereon we use $\Theta$ instead of $w$ to refer to the weights of the model to follow the convention.

In Equations 2.4 to 2.7 we derive the computations involved in the forward pass of the neural networks in Figure 2.2. We call this as the forward pass since the computations are done in one direction, from the input layer (leftmost) to the output layer.
We first introduce the notations used in the equations.

Let $a_i^j$ be the activation of unit $i$ in layer $j$, $\Theta^j$ be the matrix of weights controlling function mapping from layer $j$ to layer $j+1$ and the sigmoid function $\sigma(x)$ (see §2.4.2 for detailed discussion) be the activation function. By activation, we mean the value computed and outputted by a neuron. To compute the hidden unit $a_i^j$, we apply the sigmoid function on the linear combination of input $x$. 

The dimensions of $\Theta^j$ is determined in the following manner. If the network has $s_j$ units in layer $j$, $s_{j+1}$ units in layer $j+1$, then $\Theta^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$. For instance, layer 1 (input layer) has 3 units and layer 2 (hidden layer) has 3 units, then $\Theta^{(1)} \in \mathbb{R}^{3 \times 4}$. The notation $\theta_{ik}^{(j)}$ represents the weight of input $x_k$ (or $a_i^j k^l$ in multiple hidden layers) on activation unit $a_l^j$.

$$a_1^2 = \sigma(\Theta_{10}(1)x_0 + \Theta_{11}(1)x_1 + \Theta_{12}(1)x_2 + \Theta_{13}(1)x_3) \quad (2.4)$$

$$a_2^2 = \sigma(\Theta_{20}(1)x_0 + \Theta_{21}(1)x_1 + \Theta_{22}(1)x_2 + \Theta_{23}(1)x_3) \quad (2.5)$$
\[ a_3^2 = \sigma(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3) \quad (2.6) \]

\[ h_{\Theta}(x) = a_1^3 = \sigma(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}) \quad (2.7) \]

The units \(a_1^{(2)}, a_2^{(2)}, a_3^{(2)}\) in Equations 2.4 to 2.6 comprise the hidden layer of the neural networks (as shown in Figure 2.2). These units are effectively new features learned by the neural networks since they differ from the raw input features \(x\). These new features enable the neural networks to learn complex hypotheses \(h_{\Theta}(x)\).

To support vector operations, we can further simplify Equations 2.4 to 2.7 by rewriting them in vectorised form. To do that, we introduce some notations. Let \(x = [x_0, x_1, x_2, x_3]\) be the concatenation of \(x_k\) elements and \(z_i^{(j)}\) be the linear combination of \(\Theta_{ik}^j\) and \(x_k\) before the sigmoid function is applied. We can then write \(z^{(2)} = [z_1^{(2)}, z_2^{(2)}, z_3^{(2)}]\) as the concatenation of \(z_i^{(j)}\) elements. Equations 2.4 to 2.7 can then be rewritten in vectorised form as:

\[ z^{(2)} = \Theta^{(1)} x \quad (2.8) \]

\[ a^{(2)} = \sigma(z^{(2)}) \quad (2.9) \]

\[ z^{(3)} = \Theta^{(2)} a^{(2)} \quad (2.10) \]

\[ h_{\Theta}(x) = a^{(3)} = \sigma(z^{(3)}) \quad (2.11) \]

In the previous paragraphs we have derived the hypothesis representation of a neural network with only one hidden layer but the derivation process is the same for multiple hidden layers. Furthermore, we have shown a specific class of neural network architecture called feed-forward neural network. It is called feed-forward network (Goodfellow et al., 2016) because information flows through the function being evaluated from \(x\), through the intermediate computations used to define \(h_{\Theta}(x)\) and finally to the output \(y\). This feed-forward network has a fully-connected layer since every input
neuron is connected to every neuron in the next layer. We define \textit{neural network architecture} as the mechanism where neurons are connected in the different layers in the neural networks. We discuss in §2.3 the most common neural network architectures used in this thesis.

\subsection{Learning}

In this section, we explain how the parameters ($\Theta$) are learned in neural networks. This is also called the \textit{backward pass} of the network. We will now introduce the concept of loss function and the back-propagation algorithm.

To put this into context, let us consider the setting in a classification task (Bishop, 2006; Murphy, 2012) where we classify an input $x$ into one of the $k$ classes. Suppose we have a training set of $(x_i, y_i)$ training pairs of $m$ training examples: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)})$, $\ldots, (x^{(m)}, y^{(m)})$, $L$ as the total number of layers in the network and $s_l$ as the number of units (not counting the bias unit) in layer $l$. In a binary classification task, the labels $y$ are either 0 or 1 with $k$ output unit equals 1, the hypothesis as $h_{\Theta}(x) \in \mathbb{R}$. In a multi-class classification task where there are $k$ classes, the labels are $y \in \mathbb{R}^K$ and the hypothesis $h_{\Theta}(x) \in \mathbb{R}^K$.

For the network to be able to learn, we need to compare the predictions $h_{\Theta}(x)$ with the gold labels $y$. The difference between the predictions and the gold labels is quantified using a loss function (Hastie et al., 2009; Murphy, 2012). The cost function is the sum of all the losses on the entire training set plus some model complexity penalty such as regularisation (Goodfellow et al., 2016). The objective function is the function that we want to minimise during training (Bishop, 2006).

For the classification task defined in the previous paragraph, we can compute the loss by using cross-entropy loss function (Murphy, 2012). The cross-entropy loss function is used to measure the performance of a classification model whose output is a
CHAPTER 2. DEEP LEARNING FOR NLP

probability value. The cross-entropy $H$ between the gold label $y$ and the hypothesis $h_{\Theta}(x)$ for $k = 2$ (i.e. binary classification task) can be calculated as shown in Equation 2.12:

$$H(y, h_{\Theta}(x)) = - (y \log(h_{\Theta}(x)) + (1 - y) \log(1 - h_{\Theta}(x)))$$ (2.12)

The log function refers to the natural logarithm function. The negative sign in the expression makes the predictions $h_{\Theta}(x)$ less than 1 grow to positive infinity instead of negative infinity for the natural logarithm functions. The behaviour of Equation 2.12 is depicted in Figure 2.3. The intuition of using the cross-entropy loss is that when the prediction $h_{\Theta}(x)$ approaches 1 when the actual label is 1, the loss slowly decreases and approaches zero. As the prediction $h_{\Theta}(x)$ moves towards 0, the loss increases rapidly towards infinity. Thus, cross-entropy loss penalises predictions especially those far from the actual labels.

![Figure 2.3: The cross-entropy loss function in Equation 2.12](image)

Equation 2.13 shows the objective function that we want to minimise with respect to the parameters $\Theta$ using the cross-entropy loss. The loss is the same as in Equation 2.12 but is summed over all $k$ classes and for $m$ training samples. The loss is averaged
2.2. NEURAL NETWORKS: AN OVERVIEW

over \( m \) samples so that it is not dependent on the number of training examples.

Equation 2.13 also includes a regularisation term which prevents the network from over-fitting the training data. If a model over-fits, it will not be able to generalise well to unseen instances (to be discussed in more detail in §2.4.4). The regularisation used in Equation 2.13 is called \( L2 \) regularisation or “weight decay” (Goodfellow et al., 2016) as it is the sum of the squares of parameters \( \Theta \). The regularisation term has a lambda (\( \lambda \)) parameter. Reasonable values of the lambda parameter range between 0 and 0.1 to adjust the amount of regularisation applied to all \( \Theta \) parameters in all layers (Kuhn and Johnson, 2013).

\[
\min_{\Theta} J(\Theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} y^{(i)}_{k} \log(h_{\Theta}(x^{(i)}))_{k} + (1 - y^{(i)}_{k})\log(1 - (h_{\Theta}(x^{(i)}))_{k}) + \\
\frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{s_{l}} \sum_{s_{l+1}} (\Theta^{(l)}_{j_{l}j_{l+1}})^2
\]

(2.13)

We have defined in the previous paragraph the objective function that we want to minimise. Learning in neural networks happens when this objective function is minimised (Goodfellow et al., 2016).

To minimise the objective function, we need to be able to quantify the changes or gradients in the parameters \( \Theta \) that will be needed so that the prediction \( h_{\Theta}(x) \) will be closer to the target \( y \). The neural networks use the backpropagation algorithm (Bryson et al., 1963; Werbos, 1974; Rumelhart et al., 1988) to compute the gradients of the network with respect to the \( \theta \) parameters. In other words, we compute the partial derivatives \( \frac{\partial}{\partial \Theta^{(l)}_{j_{l}j_{l+1}}} J(\Theta) \) of Equation 2.13. Then optimisation methods such as stochastic gradient descent (Robbins and Monro, 1951) use the computed gradients to minimise the errors of the network. The name backpropagation comes from the fact that the computed gradients or “error” are backpropagated from the output layer to the first hidden layer as we will discuss in more detail in the next paragraphs.
The overview of the backpropagation algorithm as applied to the whole training set and performed on node $j$ is shown in Algorithm 1. Suppose we have the training set described at the start of this section, the next step we need to do is set the $\Delta_{ij}^{(l)}$ value to zero. $\Delta_{ij}^{(l)}$ accumulates the gradients to be able to compute the partial derivatives. Next, we iterate through each item $i$ in the training set. In the loop, we set the activation of the first layer $a^{(1)}$ equal to the input $x^{(i)}$. We then compute the activation for the other layers using the Equations 2.8 to 2.11. Using the gold or target label $y^{(i)}$ we compute the gradient of the output layer $\delta^{(L)}$. We then compute the gradients of the previous layers until the $\delta^{(2)}$. In Line 8, we accumulate the gradients into the $\Delta_{ij}^{(l)}$. We will provide a detailed discussion in the succeeding paragraphs for Lines 6-8. Finally, the partial derivatives (since we want to find out how the $\delta_{ij}^{(l)}$ modifies the output) for the whole training set is computed in Lines 9-10. Line 10 means that regularisation is not applied to the bias term (i.e. node $j = 0$). The bias is only a single term and it will not introduce too much variance by leaving it unregularised.

**Algorithm 1:** Back-propagation algorithm

1. Training set $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$
2. Set $\Delta_{ij}^{(l)} = 0$ for all $l, i, j$
3. For $i = 1$ to $m$
   4. Set $a^{(1)} = x^{(i)}$
   5. Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \ldots, L$
   6. Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$
   7. Compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$
   8. $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_{i}^{(l+1)}$
   9. $D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)}$ if $j \neq 0$
   10. $D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)}$ if $j = 0$

The partial derivatives $D_{ij}^{(l)}$ in Line 9-10 are then used by the optimisation algorithms such as gradient descent to minimise the cost function. We compute the partial derivatives because we want to know how much each variable or component in the
network contributes to the overall error. This process is repeated until the loss is small enough while constraining the model to be able to predict well on unseen instances (more details in § 2.4.4).

We show in Appendix A the detailed derivation of the partial derivatives of the cost function with respect to each component in the neural network.

The goal of backpropagation is to adjust the weights of the network in proportion to how much it contributes to the overall error. By iteratively reducing each weight’s error, we will have a final set of weights that produce good predictions.

2.2.3 Training Neural Networks

We now enumerate the steps on how to train a neural network. Firstly, we decide on the neural network architecture to use (i.e., number of input units, number of output units, number of hidden layers). Depending on the task, some architectures are known to work better in particular cases. For example, recurrent neural networks (RNNs) (Rumelhart, 1986) and its variant Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks are typically used to model sequential data while convolutional neural networks (CNNs) (Fukushima, 1980; LeCun et al., 1999) are known to capture spatial features. Furthermore, attention (Bahdanau et al., 2014), tree (Socher et al., 2011b; Tai et al., 2015a) and graph (Scarselli et al., 2008) neural network architectures can model more complex hypothesis.

Once the neural network architecture is decided, we train the neural network following these steps:

1. Randomly initialise the parameters/weights (Θ) of the network (will be discussed in detail in § 2.4.1)

2. Perform forward propagation to compute \( h_\Theta(x^{(i)}) \) for any \( x^{(i)} \) (§ 2.2.1)
3. Compute the loss using the cost function \( J(\Theta) \) (see § 2.2.2)

4. Perform back-propagation to compute partial derivatives \( \frac{\partial}{\partial \Theta} J(\Theta) \) (see § 2.2.2)

5. Minimise \( J(\Theta) \) as a function of parameters \( \Theta \) using optimisation algorithms with back-propagation (see § 2.2.2)

We have described above the representation and learning algorithm used in neural networks. In the next section, we provide an overview of the two most common neural network architectures.

### 2.3 Neural Network Architectures

In this section, we will describe two main neural network architectures that have been developed in the past 20 years which are now the building blocks of deeper architectures. Concretely, we will define the following two building blocks of deep learning architectures today: Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Convolutional Neural Network CNN (Fukushima, 1980; LeCun et al., 1999). Generally, LSTM is well-suited for sequential information while CNN for spatial information.

#### 2.3.1 LSTM

Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks are a special kind of recurrent neural networks (RNNs) that addressed the shortcomings of the latter, that is being unable to capture long-term dependencies as explored by Bengio et al. (1994). The problem of long-term dependencies happens in sequential information such as in text. In sequences, we want the neural networks to be able to connect the previous information to perform the present task. For example, consider a language
model trying to predict the next word based on the previous ones. When the gap in the sequence between the relevant information and the point where it is needed becomes large, RNNs are unable to learn to connect the information. It turns out that LSTM are capable of learning long-term dependencies.

To address the problem of long-term dependencies, an LSTM cell has four (4) interacting layers (in contrast to an RNN cell which has only one): input layer \((i_t)\), forget layer \((f_t)\), cell update layer \((\tilde{C}_t, C_t)\) and output layer \((o_t)\). Let \(\ast\) be element-wise multiplication, \((\cdot)\) be dot product, \((\oplus)\) be concatenation, \(\sigma\) be the sigmoid function, \(W\) and \(b\) be the parameters and biases, Equations 2.14-2.18 show all the interacting layers with \(h_t\) representing the hidden state at time step \(t\). The cell update layer \((C_t)\) which is a linear combination of the previous state \(C_{t-1}\) and the signal from the input \(i_t\) addresses the problem of vanishing gradients inherent in RNNs.

\[
f_t = \sigma(W_f \cdot [h_{t-1} \oplus x_t] + b_f)
\]

\[
i_t = \sigma(W_i \cdot [h_{t-1} \oplus x_t] + b_i)
\]

\[
\tilde{C}_t = \tanh(W_c \cdot [h_{t-1} \oplus x_t] + b_c)
\]

\[
C_t = f_t \ast C_{t-1} + i_t \ast \tilde{C}_t
\]

\[
o_t = \sigma(W_o \cdot [h_{t-1} \oplus x_t] + b_o)
\]

\[
h_t = o_t \ast \tanh(C_t)
\]

Vanishing gradients happen when the change (or error signal) in the neural network weights become very small (Hochreiter and Schmidhuber, 1997). When the value are very small, the neural network stops learning because it would appear that there are no changes in the values. The values of the weights become very small as an effect of multiplying small numbers to compute the gradients of an \(n\)-layer network. As the gradient decreases exponentially with \(n\), the front layers in the neural network train
very slowly. Specifically, the vanishing gradient problem is caused by activation functions that have gradients in the range \([0,1]\). Conversely, when the activation function used has derivatives that can take on larger values, the problem becomes an exploding gradient problem.

Extensions to this LSTM architecture have been explored for sequence analysis and proven to be effective at capturing contextual information. Dyer et al. (2015a) proposed a bi-directional LSTM which provides access to both past (left) and future (right) contexts in a sequence. In this case, each sequence is passed forward and backward to capture the past and future information, respectively, and then the two hidden states \(h\) are concatenated to form the final output: \(h_{\text{final}} = [h_{\text{forward}} \oplus h_{\text{backward}}]\).

Furthermore, Tai et al. (2015b) proposed Tree-LSTMs to capture information in tree structures such as in constituency trees and dependency parse trees. The difference between the standard LSTM unit and the Tree-LSTM unit is that in the latter the gating vectors (i.e., the weights corresponding to the layers in an LSTM unit) and memory cell updates are dependent on the number of child units.

In this thesis, we have used a specific type of Tree-LSTM called Child-Sum Tree-LSTMs, which is well-suited for trees whose children are unordered. For example, in the case of event structures, the order of arguments does not matter. The Child-Sum Tree-LSTM transition equations are the following:

\[
\tilde{h}_j = \sum_{k \in C(j)} h_k, \quad (2.19)
\]

\[
i_j = \sigma(W^{(i)}x_j + U^{(i)}\tilde{h}_j + b^{(i)}), \quad (2.20)
\]

\[
f_{jk} = \sigma(W^{(f)}x_j + U^{(f)}h_k + b^{(f)}), \quad (2.21)
\]

\[
o_j = \sigma(W^{(o)}x_j + U^{(o)}\tilde{h}_j + b^{(o)}), \quad (2.22)
\]
2.3. NEURAL NETWORK ARCHITECTURES

\[ u_j = \tanh(W^{(u)}x_j + U^{(u)}\tilde{h}_j + b^{(u)}) , \]  
\[ (2.23) \]

\[ c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k, \]  
\[ (2.24) \]

\[ h_j = o_j \odot \tanh(c_j), \]  
\[ (2.25) \]

where \( C(j) \) denote the set of children of node \( j \), \( k \in C(j) \) in Equation 2.21, \( x_j \) is the input at the current node \( j \), \( \odot \) denotes elementwise multiplication, \( W \) and \( U \) are trainable weight matrices, \( b \) the bias, \( \sigma \) the sigmoid function. Similar to standard LSTM units, each Tree-LSTM unit (indexed by \( j \)) contains input and output gates \( i_j \) and \( o_j \), a memory cell \( c_j \) and hidden state \( h_j \). As shown in Equations 2.20, 2.22, 2.24, the gating vectors and memory cell updates are dependent on the \( k \) number of child units. Furthermore, the Tree-LSTM unit contains one forget gate \( f_{jk} \) for each child \( k \). This allows the Tree-LSTM unit to selectively incorporate information from each child (Tai et al., 2015a).

2.3.2 CNN

Convolutional neural networks (CNNs), were originally used for computer vision applications (LeCun et al., 1998) and they have subsequently been proven to be effective in many NLP tasks. We describe in this section the typical CNN architecture following (Kim, 2014; Collobert et al., 2011).

CNNs typically consist of three (3) layers: a convolutional layer, a pooling layer and a fully-connected layer. Let \( x_i \in \mathbb{R}^k \) be the \( k \)-dimensional word vector input corresponding to the \( i \)-th word in a sentence of length \( n \), represented as:

\[ x_{\text{sentence}} = x_1 \oplus x_2 \oplus \cdots \oplus x_n, \]  
\[ (2.26) \]
where $\oplus$ is the concatenation operator. We apply a convolution operation, which involves a filter $W \in \mathbb{R}^{x \times k}$, applied to a window of $h$ words to produce a new feature, $c_i$, as follows:

$$c_i = \Psi(W \cdot x_{i:i+h-1} + b)$$ (2.27)

Here $\Psi$ is a non-linear function such as a hyperbolic tangent and $b \in \mathbb{R}$ is a bias term. By applying this operation to each possible window of words in the sentence as shown,

$$x_{1:h}, x_{2:h+1}, \ldots, x_{n-h+1:n}$$ (2.28)

we produce a feature map:

$$c = [c_1, c_2, \ldots, c_{n-h+1}]$$ (2.29)

where $c \in \mathbb{R}^{n-h+1}$.

The next layer is called the pooling layer applied over the feature map. Typically, a max pooling (Schmidhuber et al., 2012) is used where we take the maximum value: $\tilde{c} = \max\{c\}$ to get the feature of this particular filter. The idea behind using max function is to capture the most important feature for each feature map. To obtain multiple features, we use multiple filters.

The final layer, which is the fully-connected softmax layer, takes these features and outputs probability distribution over labels.

### 2.4 Modelling Considerations

In this section, we discuss our modelling considerations when using neural networks which have affected the performance of the model. We describe how we typically initialise the learning parameters of the neural networks. Another important consideration
is the selection of model hyper-parameters and training considerations. Finally, we explain the evaluation measures we used in this thesis and elaborate on the significance test to determine if the performance of the model is significantly better or comparable to the performance of other state-of-the-art techniques.

2.4.1 Parameter Initialisation and Pre-trained Embeddings

There are many ways to initialise the parameters of the network. One way is to initialise it with a normal distribution where each element of the array is initialised by a value drawn independently from the Gaussian distribution (Gauss, 1832) (also known as normal distribution) whose mean is 0 and standard deviation is according to some scale between $[0, 1]$. Since the initial weights can affect the performance of the model, we determine empirically which parameter initialiser works better for a specific problem. These parameters are initialised when the model is invoked for the first time. To achieve reproducible results (Hutson, 2018), these parameters need to be the same every time. This can be attained by setting a seed value which will be used by the number generators used in the experiments.

Parameter initialisation can be done also for input representations such as for words. In this case, there are many available pre-trained vectors (embeddings) in the general domain such as those learned using word2vec (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014). Word representations trained using word2vec are also available for the biomedical domain (Moen and Ananiadou, 2013a).

Skip-gram and CBOW (Continuous Bag of Words) are the two ways to learn the word representations using word2vec. In CBOW, the input to the word2vec model could be $w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}$, the preceding and the following words of $w_i$ and the output will be $w_i$. Hence, we can think of the task as predicting the word given its context. The number of words to the left and to the right is a parameter of the model.
Skip-gram works the other way around. The input to the model is $w_i$ and the output could be $w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}$. Hence, the task is predicting the context given a word. Mikolov et al. (2013a) found that Skip-gram works well with small amount of training data and represents well even rare words or phrases while CBOW needs more data but is faster to train and has a slightly better accuracy for frequent words.

In contrast to word2vec which is a predictive model, GloVe (Global Vectors) is a count-based model. GloVe is trained on the non-zero entries of a global word-word co-occurrence matrix, which tabulates how frequently words co-occur with one another in a given corpus. Constructing the matrix requires a single pass through the entire corpus which can be computationally expensive but this is only done once. Subsequent training iterations are much faster because it only considers the non-zero matrix entries which are typically much smaller than the total number of words in a corpus.

Baroni et al. (2014) demonstrated that, in nearly all tasks, the predictive models consistently outperform count-based models. However, Levy et al. (2015) found that there is insignificant performance difference between the methods and demonstrated that much of the performance gains are due to certain design choices and hyperparameter optimisations, rather than the embedding algorithm themselves.

More recently, sub-word representations that are pre-trained on large amount of unlabeled text are available. For example, ELMo (Embeddings from Language Models) (Peters et al., 2018) and BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019b) and its variations (Wang et al., 2019; Yang et al., 2019) which have been successful in many natural language understanding task (Wang et al., 2018a).
2.4. MODELLING CONSIDERATIONS

2.4.2 Activation Functions

Activation functions play a central role in neural networks. They determine whether a neuron has fired (or has been activated) or not. This activation signal is then transmitted to other neurons in the network. In the following section, we will justify the need for certain activation functions.

A neuron is simply the weighted sum of its input and with a bias:

\[ y = \sum (W \cdot x) + b \]  \hspace{1cm} (2.30)

The value of \( y \) can range from \(-\infty\) to \(+\infty\). Since there are no specific bounds to the values it can take, then it is difficult to decide whether the neuron has “fired” or not.

This is also called a linear activation function, which can be expressed as: \( A = cx \). Another reason why this function is not suitable is because its first derivative is a constant \( c \) with respect to the input \( x \). In this case, if there are errors in the prediction, with the back-propagation step, the changes will be constant and will not depend on the input \( x \). Furthermore, even stacking many linear layers will still have the effect a linear function since the combination of linear functions in a linear manner still results in a linear function.

What we are looking for are activation functions which have the following two properties:

1. non-linear properties (that is, having non-constant derivatives)
2. bounded prediction values

While this continues to be an active research area in the deep learning community (Carlile et al., 2017; Maas et al., 2013; Klambauer et al., 2017), there are mathematical functions that can serve that purpose albeit with some limitations. In the following sections, we will describe a number of non-linear activation functions that are widely...
used in the deep learning community.

**Sigmoid function**

The sigmoid function (Pierre-François, 1838), is defined in Equation 2.31 and its derivative in Equation 2.32:

\[
\sigma(x) = \frac{1}{1 + e^x} \tag{2.31}
\]

\[
\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x)) \tag{2.32}
\]

![Figure 2.4: Sigmoid function](image)

Figure 2.4 shows the plot of the sigmoid function. It shows that the \(y\) values are in the range \([0, 1]\). We observe that with values of \(x\) between -2 and 2, the \(y\) values have very steep slope. This means that any small changes in the value of \(x\) will have a significant change to the values of \(y\). In other words, this function has a tendency to bring values of \(y\) to either 0 or 1, which is a good property for a classifier as it makes clear distinctions on the predictions.
However, this function also has some problems. We can observe that the values of $y$ tend to respond less to changes in $x$ towards either end of the function. This means that the gradients at these values of $y$ are going to be very small. It gives rise to the problem of what they call as “vanishing gradients” since the gradients are extremely small at this $y$ values that it prevents the network from learning further or drastically slowing the learning process.

**Tanh function**

The hyperbolic tangent (or tanh) function (Becker and Orstrand, 1909) is defined in Equation 2.33 and its derivative in Equation 2.34:

\[
f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \quad (2.33)
\]

\[
f'(x) = 1 - f(x)^2 \quad (2.34)
\]

Figure 2.5 shows the tanh function. Its $y$ values are bound to the range $[-1, 1]$. 
CHAPTER 2. DEEP LEARNING FOR NLP

Note that tanh \(x\) is just a scaled sigmoid function as shown in Equation 2.35:

\[
tanh(x) = 2 \cdot \sigma(2 \cdot x) - 1 \tag{2.35}
\]

Hence, it shares the same characteristics as the sigmoid function. But since the derivatives are steeper in tanh than in sigmoid function, this leads to stronger gradients and thus pushes the predictions towards 1 and -1. However, it also shares the same problem of vanishing gradients as in sigmoid function.

**ReLu function**

The rectified linear unit (ReLu) function (Nair and Hinton, 2010) is defined as a step function in Equation 2.36 and its derivative in Equation 2.37:

\[
f(x) = \begin{cases} 
0 & \text{for } x > 0 \\
x & \text{for } x \geq 0 
\end{cases} \tag{2.36}
\]

\[
f'(x) = \begin{cases} 
0 & \text{for } x > 0 \\
1 & \text{for } x \geq 0 
\end{cases} \tag{2.37}
\]

Figure 2.6 shows the ReLu function. Its range of \(y\) values is \([0, +\infty]\). Since the values are not bounded on the positive values for \(y\), this can possibly blow up the activation.

There are two advantages of ReLu over sigmoid or tanh function. The first is the sparsity of activation. With sigmoid or tanh function, almost all neurons would fire and thus have to be processed to compute the output of the network. This makes it computationally expensive. Ideally, we want fewer neurons to activate and have sparse
activations. With ReLu, for all negative values of \( x \), the neurons will not be activated. This means fewer neurons are firing.

The second one is related to the first one. The ReLu function involves simpler mathematical operations and thus less computationally expensive.

The limitations of the ReLu function comes also from its definition. We see that with values of \( x < 0 \) the gradient can go towards 0. This means that neurons at this state will stop responding to changes in the \( x \). This is called the “dead ReLu problem” (Lau and Lim, 2017).

### 2.4.3 Hyper-parameter Tuning

Neural networks use optimisation algorithms to minimise the objective or cost functions. These optimisers have parameters, we call them as “hyper-parameters” such as the learning rate and weight decay rate (Krogh and Hertz, 1992) among others. Learning rate (or step size) is a parameter in gradient descent (Robbins and Monro, 1951) algorithms that determines how much we change the weights of the network with respect to the gradient in order to reach a (local) minimum. If the learning rate value is
too small, convergence may take very long and if the value is too large, the network weights may not converge to the local minimum. The weight decay rate is a regularisation technique which improves generalisation by preventing the weights in the network from growing too large over the course of the training. These are determined empirically in a process called tuning. Tuning is simply finding the best values of these hyper-parameters that gives the best performance of the model.

Other than the optimisers, the number of epochs on which to train the neural networks is also a hyper-parameter. Epoch is described as one forward and backward pass on the entire dataset. To determine the best number of epochs, usually an early stopping (Prechelt, 1998) condition is set based on the performance on the validation set. With the early stopping condition, the training stops when the performance on the validation set drops \( n \) times consecutively. The parameter \( n \) is also called the patience parameter.

There are more hyper-parameters in the neural networks such as the batch size and learning parameter sizes. The batch size is the number of samples processed before the model is updated. Common batch sizes range between 50 and 256, but can vary for different applications (Ruder, 2016).

Since there are a number of hyper-parameters, another challenge is how to select the combination that gives the best performance of the model. If there are few hyper-parameters, grid search may be practical. It involves enumerating through all the possible combinations of hyper-parameters and running the model on them. The best combination is the one that gives the best performance on the task.

If there are too many hyper-parameters, this may become impractical. The usual alternatives in these cases would be to use random search, which is basically a random selection of combinations of hyper-parameters. Bergstra and Bengio (2012) have shown that random search are more efficient empirically and theoretically for hyper-parameter optimisation than grid search. In contrast to random or grid search, bayesian
hyper-parameter optimisation approaches (Snoek et al., 2012; Bergstra et al., 2013) keep track of the past evaluation results to inform the search. Bergstra et al. (2013) showed that bayesian hyper-parameter optimisation approach is able to find better hyper-parameters than random search in the same number of trials.

2.4.4 Training Considerations

When training a neural network on a specific task, the objective is to minimise the cost or loss of an objective function. Depending on the network’s capacity, the loss on the training set could reach almost zero, however, the performance on the validation set could also dramatically decrease. This condition is called overfitting (Hawkins, 2004). The model fits on the training set such that it fails to generalise to the validation set or generally unseen instances.

The opposite behaviour could happen. The model could struggle to fit to the training set as well as the validation set. This is called underfitting. It may occur due to model simplicity. In other words, the features learned by the model are not enough to make accurate predictions.

Ideally, we want to choose a model that exhibits a trade-off between overfitting and underfitting. We do this by carrying out some diagnostics. Concretely, we compare the training and validation error. It is important to note that these performance comparison should be conducted on the validation set. Hence, to be able to do model selection, we must prepare three sets of our data: training set, validation set and the test set.

In addition, we can perform $k$-fold cross validation (Fushiki, 2011). In this method we combine the training and validation set and split the dataset into $k$-folds. We use the $k - 1$ folds as the training set and the remaining one fold as the validation set. We do this for all the $k$ folds such that each fold is used as a validation set.

Once we have the performance on the training and validation set, we compare the
errors or accuracy of the model in both data sets. Underfitting (or high bias) problem exhibits both high training and validation error, while overfitting (high variance) problem exhibits low training error but high validation error. Figure 2.7 illustrates the trade-off between underfitting and overfitting. This dilemma is also called the bias-variance trade-off (Geman et al., 1992).

![Figure 2.7: Underfitting vs Overfitting (Bias-Variance) trade-off](image)

Once we have determined the problem, we can try to investigate if doing any of the following will resolve the problem. To remedy underfitting, we can investigate the effect of any of following (Ng, 2011):

1. **Train in more epochs.** Finding the best model is an optimisation problem where the training error is minimised. It is possible that the training is very slow and needs more epochs to reach the global optimum. We can also adjust the hyper-parameter *learning rate* so that the convergence will be faster.
2.4. MODELLING CONSIDERATIONS

2. **Obtain more features.** This means that the model does not have enough or does not have distinguishing features to be able to do prediction. This is another avenue to improve by designing or adding new input features.

3. **Increase model complexity.** In the case of neural networks, model complexity can be increased by adding more layers or logistics units. This can be achieved by adding linear followed by non-linear activation functions.

4. **Decrease regularisation.** Regularisation is a technique to help the algorithm generalise better (Loone and Irwin, 2001). Regularisation techniques can be applied such as $L_1$ and $L_2$ regularisation, in which a regularisation term is added to the cost function $J$, shown in Equation 2.38 and 2.39, respectively.

$$J = loss + \frac{\lambda}{2m} \sum ||w||$$

$$J = loss + \frac{\lambda}{2m} \sum ||w||^2$$

However, the **lambda** is a hyper-parameter between $[0.0, 1.0]$ that can be tuned as well. Hence, if there is underfitting, we decrease the value of this hyper-parameter. Another regularisation technique is called **dropout** (Srivastava et al., 2014). The main idea of dropout is that it randomly drop units from the network (along with their connections) during training. The dropout rate refers to the probability of retaining an input. This value is set between $[0.0, 1.0]$ and therefore can also be tuned. To reduce underfitting, we **increase** the dropout rate value.

For overfitting scenarios, we can try any of the following (Ng, 2011):

1. **Apply or increase regularisation.** We can apply or increase the regularisation values of $L_1$ and $L_2$ regularisation or **decrease** the dropout rate value.
2. **Obtain more data.** If it is possible to get more data, this will solve the overfitting problem as more instances will possibly cover a wider data distribution. This leads to more updates and more precise values for the parameters in the network. One way to check if this improves the performance is by plotting the learning curve over data set size. If the performance increases as data set size is increased, this means that adding more data will likely improve the model performance.

3. **Decrease number of features.** This may involve simplifying or reducing the input representation. This means removing some features or decreasing the dimensionality of the input features.

4. **Decrease model complexity.** This can be done by removing some hidden layers or logistic units in the network.

While addressing the underfitting or overfitting problem, it is helpful to plot the learning curve to see the effect of each of the changes to the model.

### 2.4.5 Evaluation

We evaluate the performance of the model based on how well it is able to predict the gold labels correctly (true positives). However, we also have to account the false predictions of the model (false positives) as well as those missed by the model (false negatives). We can describe these information via the precision, recall and F1 score measures (Chinchor, 1992). Precision and recall are defined as follows:

\[
Precision = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \quad (2.40)
\]

\[
Recall = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \quad (2.41)
\]
2.4. MODELLING CONSIDERATIONS

Precision is the ratio of the instances correctly labelled as belonging to the positive class (or true positives) over the total number of instances labelled as belonging to the positive class. The positive class is the sum of the true positives and the false positives, which are the instances incorrectly labelled as belonging to that class. Recall is the ratio of the true positives over the total number of instances that actually belong to that class, which is the sum of the true positives and false negatives. False negatives are the instances which were not labelled as belonging to the positive class but should have been. Depending on the task requirements, either of these two measures are used. However, F1-score is used to capture the trade-off between precision and recall. It is defined as a harmonic mean between the two:

\[
F = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]  

(2.42)

When we have to compare the performance of two systems, we want to quantify the probability that the difference between the two systems is due to luck. If the probability is low, then the difference is real. Otherwise, either the systems are not different or the data are insufficient to show that they are different. In this work, we use the approximate randomisation test (Noreen, 1989; Yeh, 2000) to compute this probability as summarised in Algorithm 2.

**Algorithm 2: Approximate randomisation test**

1. Let \( o_A = o_A^1, \ldots, o_A^n \) and \( o_B = o_B^1, \ldots, o_B^n \) be the output of the two systems on the same input.
2. Start with \( X = o_A \) and \( Y = o_B \).
3. Repeat \( R \) times the following: randomly flip each \( o_A^i, o_B^j \) between \( X \) and \( Y \) with probability = 0.5. Calculate \( t(X, Y) \) where \( t \) is the evaluation measure such as F1-score.
4. Let \( r \) be the number of times that \( t(X, Y) \geq t(o_A, o_B) \).
5. As \( R \to \infty \), \( \frac{r + 1}{R+1} \) approaches the significance level, where \( R = 1000 \) as the typically case.

The null hypothesis is that there is no difference between the two systems \( A \) and \( B \).
Values of $R$ (also known as p-value) below 0.05 (the significance level decided a priori) suggest that the null hypothesis is false and therefore there is a significant statistical difference between the two systems.

2.5 Conclusion

In this chapter, we started with some motivation why deep learning with neural networks is reasonable to explore. We then introduced neural networks - its representation and learning algorithm. We also discussed some common neural network architectures used in NLP which form the building blocks of neural networks in the literature and are used in this thesis. Finally we explain some of the different modelling aspects that must be considered when using neural networks.

In the next chapter, we will introduce event structures, present a survey of the state-of-the-art techniques in event detection and introduce our biomedical event detection approach.
Chapter 3

Event Structure Detection

(This chapter is an extended version of the author’s work accepted to the Empirical Methods for Natural Language Processing (EMNLP) Conference 2019. (Espinosa et al., 2019))

In this chapter, we define event structures and discuss its properties. We then present a survey of the state-of-the-art approaches in both general and biomedical domains. Finally, we introduce our proposed approach.

3.1 Event Structures

We begin by defining in this section what is an event structure. Then, we discuss some properties of an event structure and how to represent events in a data structure.

3.1.1 What is an Event?

An event is a semantic structure consisting of a trigger usually a verb or its nominalised form and its set of unordered arguments which are usually entities with their corresponding roles (Ananiadou et al., 2014, 2010; Hogenboom et al., 2011; Wattarujeeekrit et al., 2004a).
Figure 3.1 shows an example of an event structure. The event type is \textbf{election} represented by the trigger \textit{elected}. In this example, the event has three arguments which are entities. One argument is a \textbf{PER} (or Person) entity type with role \textbf{Official} and the actual mention is \textit{Boris Johnson}. Another argument is also a \textbf{PER} entity type with role \textbf{Position} and the actual mention is \textit{prime minister}. Lastly, the event has another argument of \textbf{GPE} (or GeoPoliticalEntity) entity type with role \textbf{Place} with the actual mention \textit{UK}.

To extract an event structure involves a number of steps such as named-entity recognition, trigger detection, relation extraction and event detection. We will discuss them in more detail in §3.2.

![Figure 3.1: An example event structure of type election.](image)

In the current example (Figure 3.1), the \textbf{election} event has the following structure template and the corresponding instance:

\textbf{Template}: \textit{Election}: \{ \textbf{Official}: \textit{PER}, \textbf{Position}: \textit{PER}, \textbf{Place}: \textit{GPE} \}

\textbf{Instance}: \textit{Election}: \{ \textbf{Official}: \textit{Boris Johnson}, \textbf{Position}: \textit{prime minister}, \textbf{Place}: \textit{UK} \}

However, an \textbf{election} event can have structural variations. Figures 3.2 and 3.3 shows some examples of such structural variations of the \textbf{election} event. Figure 3.2 shows
an election event with only two arguments, that is, with no PER argument of role Official. Figure 3.3 shows an election event with also two arguments but unlike Figure 3.1, it does not have GPE argument of role Place. In both cases, the event detection algorithm should be able to handle these structural variants.

Figure 3.2: An example event of type election with structure different from Figure 3.1.

Figure 3.3: Another example event of type election with structure different from Figure 3.1.

Another property of event structure which makes it different from dependency
structure is that it is word order invariant. Dependency structures represent the syntactic structure while event structures represent the semantic structure which makes it invariant to word order. For example, given the following sentences A and B:

**A:** Boris Johnson elected as prime minister of the UK

**B:** The UK elected Boris Johnson as prime minister

If we generate the dependency structures of sentences A and B, it would result in distinct dependency structures as the syntax are different. The dependency structures for sentences A and B are shown in Figures 3.4 and 3.5, respectively.

Figure 3.4: Dependency structure of sentence A.

Figure 3.5: Dependency structure of sentence B.

However, if we look at the event structures for sentences A and B, we find in Figures 3.6 and 3.7 that they are the same. Both Figures 3.6 and 3.7 have three arguments
3.1. EVENT STRUCTURES

3.1.2 Event Structure Representation

In section 3.1.1, we have defined what is an event and discuss some of its properties. In this section, we discuss how we represent events.

The events we have explored thus far are considered simple or flat events since all their arguments are entities. Flat events can be represented using a tree data structure as shown in Figure 3.8.

However, events can have events as well as their arguments. These type of events are called nested events. Furthermore, these nested events can overlap with other
events as they share common arguments with other events, and thus are called overlapping events. We will define these different types of event in the following paragraphs.

Nested and overlapping event structures, which occur widely in text, are important because they can capture relations between events such as causality, e.g., a “production” event is a consequence of a “discovery” event, which in turn is a result of an “exploration” event.

Figure 3.9 shows an example of a nested and overlapping event structure in the biomedical domain. The relation graph (topmost) forms a directed acyclic graph (DAG) structure (McClosky et al., 2011) in contrast to trees in the flat events. The relation graph encapsulates 15 event structures. It contains nested event structures such as $E_2, E_3$ because one of their arguments, in this case $E_1$, is an event. Specifically, $E_1$ is a flat event since its argument is an entity. Moreover, $E_2$ and $E_3$ are also overlapping events (explicitly shown in the relation graph having two induction triggers) because they share a common argument, $E_1$.

The DAG-structured representation of event structures makes it more complex than tree-structured dependency structures. Thus, dependency parsing methods cannot be directly applied to event extraction. In the following section, we examine the different approaches to event extraction.

\subsection{Event Structure Extraction}

In this section, we first define the overall task of event extraction and specifically the sub-task of event detection.

We have seen in the previous section what is an event structure. We saw that events are composed of a trigger and its set of arguments with their corresponding roles. If we break apart the event extraction process, it involves a number of sub-tasks which we will discuss in the succeeding paragraphs.
3.2. EVENT STRUCTURE EXTRACTION

The first (or one of the tasks if we perform this task simultaneously with the other sub-tasks) task involves detecting the trigger and the arguments of the trigger. This task can be formulated as an entity or trigger detection problem. Concretely, this involves determining which words and phrases in a sentence potentially constitute as arguments or participants of an event. If we take Figure 3.1 as an example, this first task involves detecting the trigger elected and its corresponding type which is election. Furthermore, we need to detect the individual arguments and their types. In Figure 3.1, this means detecting Boris Johnson as PER entity type, prime minister as PER entity...
type and *UK* as *GPE* entity type.

Secondly, the pairwise relations between the triggers and arguments are detected or extracted. This task falls into the relation extraction problem. If the trigger or argument has been extracted in a previous step, then at this point the task only involves detecting the relationship type. In Figure 3.1 the relationship type are the roles. For example, the role or relationship type between the trigger *elected* and the argument *Boris Johnson* is *Official*. In some approaches, as we will see later in §3.3.1, the argument and the type, the trigger and type and the relationship type or role is extracted in a joint manner as a triplet.

Finally, to extract the event structure, we need to combine the pairwise relations into complete event structures. To determine which combination of relations would form a valid structure is called event structure detection. However, this naive process of generating combinations and classifying if they are event or not can lead to exponential running time complexity especially when there are many arguments. We will examine this in more detail later in this chapter. Furthermore, as we have seen in Section 3.1.1, an event type can have many variants and therefore the event extraction algorithm should be able to handle this. Finally, events can be nested and overlapping which adds to the further complexity of this task.

In this work, we focus on event detection since the current methods do not explicitly model nested and overlapping structures in learning as we will see in the next section (§3.3). This task is essential to capture relations between events such as causality, e.g., a “production” event is a consequence of a “discovery” event, which in turn is a result of an “exploration” event.

There are two main lines of approaches to extract events: the pipeline approach (Björne and Salakoski, 2018a; Miwa et al., 2013b) and the joint approach (Rao et al., 2017; Riedel and McCallum, 2011b; Vlachos and Craven, 2012; Venugopal et al., 2014a; McClosky et al., 2011). In the pipeline approach, the three sub-tasks mentioned above
3.2. EVENT STRUCTURE EXTRACTION

are extracted performed in sequence. It means that the entities and triggers are detected first and based on the predictions, the relationships between entities and triggers are detected next and finally, the event structures are detected from the pairwise relations. In contrast, the joint approach performs all the different subtasks and detects the entities, triggers, relationships and events together.

At the sentence-level extraction, some studies (Wattarujeekrit et al., 2004b; KOKKINAKIS, 2013; Spiliopoulou et al., 2017; Judea and Strube, 2015) have viewed event extraction and frame-semantic parsing or semantic role labelling (SRL) as identical tasks. In frame-semantic parsing, the frames (which are also called events or predicate-argument structures) include a predicate, which determines the events, and arguments. A number of resources containing frames have been developed over the years, e.g. FrameNet (Baker et al., 1998; Ruppenhofer et al., 2006), PropBank (Kingsbury and Palmer, 2002), VerbNet (Schuler, 2005) and NomBank (Meyers et al., 2004). However, there are some important differences between frames (or events) in SRL and events in information extraction (IE). Firstly, not all predicates in frames invoke or correspond to events in IE (Judea and Strube, 2015). A predicate may be detected in a sentence but this does not always invoke an event since the latter requires specific arguments to be present. The kind of arguments that an event can take are defined in each respective event dataset. Furthermore, in some cases, the event trigger does not always correspond to a predicate as it can be a noun. For instance, the word fatal in the phrase fatal accident can be a trigger for a Die event. Secondly, the event arguments in IE are defined based on entity types. In ACE 2005 event dataset In general, event extraction in IE is a higher level of semantic abstraction than frames (or events) in SRL.

In this work, we focus on sentence-level event detection since the current methods do not explicitly model nested and overlapping structures in learning as we will see in

1https://catalog.ldc.upenn.edu/LDC2006T06
CHAPTER 3. EVENT STRUCTURE DETECTION

the next section (§3.3). This task is essential to capture relations between events such as causality, e.g., a “production” event is a consequence of a “discovery” event, which in turn is a result of an “exploration” event.

3.3 Event Detection Approaches

In this section, we survey the various event detection approaches. We examine the domains and datasets that has been explored and developed, respectively, for event extraction. We then take a closer look at the current state-of-the-art approaches for event detection on these datasets. Finally, we present an analysis of the current approaches on which we build our approach on.

3.3.1 Domains and Datasets

The research on event detection has been driven largely by the availability of benchmark datasets in various domains. In this work, we focus our discussion to the two most widely studied domains specifically the general domain and the biomedical domain.

In the general domain, the progress of information extraction in the field has been driven by the shared tasks based on carefully curated resources such as those organised in the Message Understanding Conferences (MUC) \(^2\) (Grishman and Sundheim, 1996) and continued with the Automatic Content Extraction (ACE) program \(^3\) (Doddington et al., 2004).

The Linguistic Data Consortium (LDC) developed the annotation guidelines for the ACE program which consists of three primary ACE annotation tasks: Entity Detection

\(^2\)https://cs.nyu.edu/cs/faculty/grishman/muc6.html
\(^3\)https://catalog.ldc.upenn.edu/ldc2006t06
and Tracking (EDT), Relation Detection and Characterisation (RDC) and Event Detection and Characterisation (EDC). The proposed tasks by ACE are more challenging than their MUC forerunners because of the increased complexity of the dataset and the introduction of more fine-grained entity types. Furthermore, the EDC was introduced in the Text Analysis Conferences (TAC)\footnote{https://tac.nist.gov/about/index.html} evaluation workshops where the focus is on the extraction of events from text along with their participants and relations. The above-mentioned initiatives have provided the research community a way to evaluate and compare information extraction systems and approaches.

In this work, we are interested in EDC where the task is to identify the type of event in which entities participate. Concretely, the textual mention or anchor for each event is categorised to its type and subtype. Furthermore, the arguments and attributes for each event are also identified according to a type-specific template. The ACE dataset contains datasets in English, Chinese and Arabic and gathered from various information sources such as broadcast transcripts, newswire and newspaper data. For event detection (or EDC), the ACE dataset has eight (8) event types and thirty-three (33) sub-types. The annotations are tagged on the document-level. The training set contains 529 documents which the development and test set contain 30 and 40 documents, respectively. It is to be noted that the ACE dataset does not contain nested events.

This has also been the case in biomedical event detection. The first few curated resources focused on information retrieval (IR) such as the TREC Genomics track (Hersh et al., 2006) and named entity recognition such as the JNLPBA (Kim et al., 2004) and then later on the focus started to shift to information extraction (IE) such as LLL (Nédellec, 2005) and BioCreative (Hirschman and Valencia, 2007) which seeks to extracts the relations between bio-molecules.

More recently, the BioNLP shared task (ST) 2009 (Kim et al., 2009a) took a definitive step further toward finer-grained information extraction. In contrast to LLL and
CHAPTER 3. EVENT STRUCTURE DETECTION

BioCreative, the BioNLP-ST 2009 concerns the detailed behaviour of bio-molecules, characterised as bio-molecular events. The succeeding BioNLP-ST 2011 (Kim et al., 2011a) preserves the general design and goals of the previous shared task but adds a new focus of variability to address a limitation of the BioNLP-ST 2009 which was based only on the Genia corpus (Kim et al., 2008) and thus restricting the community-wide effort to resources developed by a single group for a small subdomain of molecular biology. The BioNLP-ST 2011 (Kim et al., 2011a) included four main tracks (with five tasks) representing fine-grained bio-IE: Genia task (GE), Epigenetics and post-translational modification task (EPI), Infectious diseases task (ID) and Bacteria track (Bacteria biotope task (BB), Bacteria interaction task (BI)).

Similar to the previous two editions of BioNLP-ST, the BioNLP-ST 2013 (Nédellec et al., 2013) aimed to measure the progress accomplished by the community on complex text-bound event extraction and shared the common goal with other tasks. It consisted of six tasks: Genia Event Extraction for NFkB knowledge base (GE), Cancer Genetics (GE), Pathway Curation (PC), Corpus Annotation with Gene Regulation Ontology (GRO), Gene Regulation Network in Bacteria (GRN) and Bacteria Biotopes (BB). This edition covered many new hot topics that reflected the evolving needs of biologists and broadened the scope of the text-mining application domains in biology by introducing new issues on cancer genetics and pathway curation while building on the well-known previous datasets GENIA, LLL, BI and BB to propose tasks closer to the actual needs of biological data integration.

BioNLP-ST 2013 shared the common goal of providing a common framework for the comparative evaluation of information extraction (IE) methods in the biomedical domain with other tasks, such as BioCreative (Critical Assessment of Information Extraction in Biology) (Arighi et al., 2011), DDIExtraction (Extraction of Drug-Drug Interactions from biomedical texts) (Segura Bedmar et al., 2011) and i2b2 (Informatics for Integrating Biology and the Bedside) Shared-Tasks (Sun et al., 2013). However, we
3.3. EVENT DETECTION APPROACHES

<table>
<thead>
<tr>
<th>Corpus/Dataset</th>
<th>Num Event Types</th>
<th>Reported Dataset Size</th>
<th>Inter-Annotator Agreement (Kappa (%))</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACE 2005</td>
<td>34</td>
<td>599 transcripts / news items</td>
<td>73 (Ji and Grishman, 2008)</td>
</tr>
<tr>
<td>GRO 2013</td>
<td>126</td>
<td>300 abstracts</td>
<td>43-56 (Kim et al., 2011c)</td>
</tr>
<tr>
<td>CG 2013</td>
<td>42</td>
<td>5938 sentences / 600 abstracts</td>
<td>70-80 (Pyysalo et al., 2013a)</td>
</tr>
<tr>
<td>PC 2013</td>
<td>24</td>
<td>5040 sentences / 525 abstracts</td>
<td>61 (Ohta et al., 2013)</td>
</tr>
<tr>
<td>EPI 2011</td>
<td>16</td>
<td>11772 sentences</td>
<td>82-89 (Ohta et al., 2011)</td>
</tr>
<tr>
<td>GE 2013</td>
<td>15</td>
<td>8369 sentences / 34 full papers</td>
<td>Not reported (Kim et al., 2013)</td>
</tr>
<tr>
<td>GRN 2013</td>
<td>12</td>
<td>174 abstracts</td>
<td>Not reported (Bossy et al., 2013a)</td>
</tr>
<tr>
<td>ID 2011</td>
<td>11</td>
<td>5118 sentences</td>
<td>75 (Pyysalo et al., 2011a)</td>
</tr>
<tr>
<td>GE 2009</td>
<td>10</td>
<td>11380 sentences / 1210 abstracts</td>
<td>Not reported (Kim et al., 2009b)</td>
</tr>
<tr>
<td>GE 2011</td>
<td>10</td>
<td>14958 sentences / 1210 abstracts and 14 full papers</td>
<td>Not reported (Kim et al., 2011b)</td>
</tr>
<tr>
<td>BB 2013</td>
<td>2</td>
<td>563 web pages</td>
<td>Not reported (Bossy et al., 2013b)</td>
</tr>
<tr>
<td>REL 2011</td>
<td>1</td>
<td>11351 sentences</td>
<td>Not reported (Pyysalo et al., 2011b)</td>
</tr>
</tbody>
</table>

Table 3.1: Datasets on the general (ACE2005) and biomedical domain (BioNLP-STs) on event detection and their respective details. The BioNLP-ST datasets are sorted in descending order according to the number of event types in each domain. Acronym definitions: Automatic Content Extraction (ACE), Gene Regulatory Ontology (GRO), Cancer Genetics (CG), Pathway Curation (PC), Epigenetics and Post-translational Modification (EPI), Gene Regulation Network (GRN), Infectious Diseases (ID), Genia (GE), Bacteria Biotopes (BB), Entity Relations (REL).

are mainly interested with the BioNLP-ST series because compared to the other tasks it proposed a linguistically motivated approach to event representation that enables the evaluation of the participating methods in a unifying computer science framework.

Table 3.1 shows the summary of the event detection datasets in both the general and biomedical domains. In Section §3.4 where we discuss our approach, we will also explain the choice of datasets in which we evaluate our proposed approach based on some issues such as comparability with the state-of-the-art systems and accessibility of online evaluation servers.

At this point, it is important to know that among the datasets in Table 3.1, only the ACE 2005 dataset does not contain nested events and thus BioNLP datasets are considered more complex in terms of event structures. Table 3.1 is sorted in descending order by the number of event types each dataset has in the general and biomedical domain, respectively. Some datasets provided information about the source of the documents (e.g., abstract, full paper, web pages, etc) with the corresponding number of extracted sentences, if available. All the tasks (except GRN which uses Slot Error Rate) use the F1-score as the standard evaluation measure (see §2.4.5 for details).
The last column reports the inter-annotator agreement (IAA) in terms of Kappa score (if reported). Although some papers in Table 3.1 did not report IAA scores, which makes it difficult to know the upperbound performance expected of proposed systems, they have employed alternative techniques to ensure the quality of the annotations as described in their respective papers.

Another important thing to point out is that all the datasets in Table 3.1 are annotated on the document level and thus they require anaphora and coreference resolution techniques to extract inter-sentence events. In this thesis, we only focus on sentence-level events which comprise most of the sentences, as will see in detail in the next sections.

3.3.2 General Approaches

In this section, we survey the different approaches on event detection in the general and biomedical domains. For the general domain, we report the the top five systems for ACE2005. For the biomedical domain, we first identified datasets from Table 3.1 which have been widely evaluated on in the literature. This turned out to be the main tasks in the BioNLP STs, which are the following: GE 2009, GE 2011, GE 2013, CG 2013, PC 2013. This can be attributed to the size of datasets, number of event types and accessibility of datasets. These five datasets will enable us to make a comparison of the different approaches as we report the top five systems in each.

We report in Table 3.2 the top five best performing–thus, state-of-the-art–event detection systems, based on the performance that has been reported in the literature. In the following paragraphs, we discuss the main approaches used by the top five systems in each dataset. We can observe that for the ACE2005 dataset, all the top five systems used a neural-based approach. The recency of the publication dates shows the popularity and thus importance of this dataset in the research community. In fact, Table
3.3. EVENT DETECTION APPROACHES

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Top 5 Systems</th>
<th>F1 Score</th>
<th>Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACE 2005</td>
<td>DEEB-RNN3 (Zhao et al., 2018)</td>
<td>74.00</td>
<td>Neural-based</td>
</tr>
<tr>
<td></td>
<td>DEEB-RNN1 (Zhao et al., 2018)</td>
<td>73.70</td>
<td>Neural-based</td>
</tr>
<tr>
<td></td>
<td>DEEB-RNN2 (Zhao et al., 2018)</td>
<td>73.40</td>
<td>Neural-based</td>
</tr>
<tr>
<td></td>
<td>GCN (Nguyen and Grishman, 2018)</td>
<td>73.40</td>
<td>Neural-based</td>
</tr>
<tr>
<td></td>
<td>Human baseline (Ji and Grishman, 2008)</td>
<td>73.00</td>
<td>Neural-based</td>
</tr>
<tr>
<td></td>
<td>Gated Multilingual Attention (Liu et al., 2018)</td>
<td>72.40</td>
<td>Neural-based</td>
</tr>
<tr>
<td>GE 2009</td>
<td>SVM (Miwa et al., 2012)</td>
<td>58.27</td>
<td>Feature-based</td>
</tr>
<tr>
<td></td>
<td>Heuristics + learning-based method (Venugopal et al., 2014b)</td>
<td>58.16</td>
<td>Rule-based + Feature-based</td>
</tr>
<tr>
<td></td>
<td>CNN mixed 5 x ensemble (Björne and Salakoski, 2018b)</td>
<td>57.84</td>
<td>Neural-based</td>
</tr>
<tr>
<td></td>
<td>Heuristics (Riedel and McCallum, 2011a)</td>
<td>57.40</td>
<td>Rule-based</td>
</tr>
<tr>
<td></td>
<td>CNN 5 x ensemble (Björne and Salakoski, 2018b)</td>
<td>56.10</td>
<td>Neural-based</td>
</tr>
<tr>
<td>GE 2011</td>
<td>CNN mixed 5 x ensemble (Björne and Salakoski, 2018b)</td>
<td>58.10</td>
<td>Neural-based</td>
</tr>
<tr>
<td></td>
<td>Heuristics + learning-based method (Venugopal et al., 2014b)</td>
<td>58.07</td>
<td>Rule-based + Feature-based</td>
</tr>
<tr>
<td></td>
<td>SVM (Miwa et al., 2012)</td>
<td>57.98</td>
<td>Feature-based</td>
</tr>
<tr>
<td></td>
<td>CNN 5 x ensemble (Björne and Salakoski, 2018b)</td>
<td>57.87</td>
<td>Neural-based</td>
</tr>
<tr>
<td></td>
<td>CNN Single (Björne and Salakoski, 2018b)</td>
<td>56.80</td>
<td>Neural-based</td>
</tr>
<tr>
<td>GE 2013</td>
<td>Heuristics + learning-based method (Venugopal et al., 2014b)</td>
<td>53.61</td>
<td>Rule-based + Feature-based</td>
</tr>
<tr>
<td></td>
<td>CNN mixed 5 x ensemble (Björne and Salakoski, 2018b)</td>
<td>53.00</td>
<td>Neural-based</td>
</tr>
<tr>
<td></td>
<td>TEES + SVM Reranking (Hakala et al., 2013)</td>
<td>50.97</td>
<td>Feature-based</td>
</tr>
<tr>
<td></td>
<td>SVM (Björne and Salakoski, 2013)</td>
<td>50.74</td>
<td>Feature-based</td>
</tr>
<tr>
<td></td>
<td>CNN 5 x ensemble (Björne and Salakoski, 2018b)</td>
<td>50.34</td>
<td>Neural-based</td>
</tr>
<tr>
<td>CG 2013</td>
<td>Human baseline (Pyysalo et al., 2013a)</td>
<td>70.80</td>
<td>Neural-based</td>
</tr>
<tr>
<td></td>
<td>CNN mixed 5 x ensemble (Björne and Salakoski, 2018b)</td>
<td>57.60</td>
<td>Neural-based</td>
</tr>
<tr>
<td></td>
<td>CNN 5 x ensemble (Björne and Salakoski, 2018b)</td>
<td>56.74</td>
<td>Neural-based</td>
</tr>
<tr>
<td></td>
<td>CNN Single (Björne and Salakoski, 2018b)</td>
<td>55.52</td>
<td>Neural-based</td>
</tr>
<tr>
<td></td>
<td>SVM (Björne and Salakoski, 2013)</td>
<td>55.41</td>
<td>Feature-based</td>
</tr>
<tr>
<td></td>
<td>SVM (Miwa and Ananiadou, 2013)</td>
<td>52.09</td>
<td>Feature-based</td>
</tr>
<tr>
<td>PC 2013</td>
<td>Human baseline (Ohta et al., 2013)</td>
<td>61.00</td>
<td>Neural-based</td>
</tr>
<tr>
<td></td>
<td>CNN mixed 5 x ensemble (Björne and Salakoski, 2018b)</td>
<td>55.62</td>
<td>Neural-based</td>
</tr>
<tr>
<td></td>
<td>CNN 5 x ensemble (Björne and Salakoski, 2018b)</td>
<td>53.09</td>
<td>Neural-based</td>
</tr>
<tr>
<td></td>
<td>SVM (Miwa and Ananiadou, 2013)</td>
<td>52.10</td>
<td>Feature-based</td>
</tr>
<tr>
<td></td>
<td>CNN Single (Björne and Salakoski, 2018b)</td>
<td>52.84</td>
<td>Neural-based</td>
</tr>
<tr>
<td></td>
<td>SVM (Björne and Salakoski, 2013)</td>
<td>51.10</td>
<td>Feature-based</td>
</tr>
</tbody>
</table>

Table 3.2: Top 5 systems for each dataset in the general and biomedical domains in terms of F1-score (along with the human baseline performance if available) and their respective approach.

3.2 shows that the human baseline performance has already been surpassed. Another popular approach is the feature-based method (McClosky et al., 2011; Hong et al., 2011; Li et al., 2014; Ji and Grishman, 2008; Liao and Grishman, 2010; Huang and Riloff, 2012; Ahn, 2006) which rely mainly on a set of hand-designed discriminative features such as lexical or syntactic features to build statistical models. Some of these approaches (Chen et al., 2017; Liu et al., 2016, 2017) have tackled data sparsity problems by exploiting additional information and data such as FreeBase and FrameNet.

The main disadvantage of this approach is that it requires domain expertise and it is
CHAPTER 3. EVENT STRUCTURE DETECTION

For the BioNLP datasets, we can observe that there are three main approaches in the top five systems, namely: rule-based, feature-based and neural-based approaches. The neural-based approach yielded the best performance in three of the five datasets, thus indicating the effectiveness of this approach. Moreover, close to half (i.e., thirteen) of the all the top five systems (i.e., twenty-five) use neural-based approaches. The second most prevalent approach is the feature-based approach (i.e., eight), then by a mix of rule-based and feature-based (i.e., three) approaches and finally the rule-based (i.e., one) approach. Compared to the human baseline performance (for those with reported human baseline/IAA score), we can see in Table 3.2 that the performance of the proposed systems still fall below the human performance. This indicates the difficulty of the task.

3.3.3 Analysis of the Current Approaches

We now discuss the advantages and limitations of each approach in general. The advantage of rule-based approach is that it allows the explicit integration of domain knowledge or patterns about event structures into the system. This is especially useful when the event structure pattern distribution is rather skewed. However, the main drawback for this approach is that it needs domain expertise. Furthermore, it is a time-consuming and a labour-intensive process. Because of this limitation, rule-based approaches cannot be easily applied to new domains or datasets. This drawback is addressed by the feature-based approach where features are hand-engineered and machine learning classifiers are trained to do the classification of the event structures. Some of these features are domain-independent and hence some of the feature-based approaches can be relatively easy to apply to other domains or datasets. However, this approach requires the domain expertise and labour-intensive hand-engineering process
3.3. EVENT DETECTION APPROACHES

to determine the relevant features. The neural-based approach aims to address this limitation as it learns the features automatically from the dataset. This approach learns vector representations that implicitly captures the important features that otherwise have to be hand-engineered in the feature-based approach. This allows the possibility of easily adopting this approach to other domains and datasets. However, this approach also has its limitations. Compared to the predictions using the rule-based approach, the explainability of the predictions in the neural-based approach is not straightforward and still proves to be a challenge.

As shown in Table 3.2, the state-of-the-art approaches for event extraction in the biomedical domain are pipeline systems (Björne and Salakoski, 2018a; Miwa et al., 2013b). Other joint approaches have also been explored (Rao et al., 2017; Riedel and McCallum, 2011b; Vlachos and Craven, 2012; Venugopal et al., 2014a), but they focus on finding relation graphs and detect events with rules. McClosky et al. (2011) treats events as dependency structures by constraining event structures to map to trees, thus their method cannot represent overlapping event structures.

Table 3.2 also shows other neural models in event extraction in the general domain (Zhao et al., 2018; Feng et al., 2018; Nguyen and Grishman, 2018; Liu et al., 2018; Feng et al., 2016; Nguyen et al., 2016; Nguyen and Grishman, 2015; Chen et al., 2015), but they used the ACE2005 corpus which does not have nested events (Miwa et al., 2014). Furthermore, there are some efforts on applying transition-based methods on DAG structures in dependency parsing, e.g., (Sagae and Tsujii, 2008; Wang et al., 2018c), however, they do not consider overlapping and nested structures.

In this work, we focus on biomedical nested and overlapping event detection using the neural-based approach to address the limitations of the current approaches as mentioned above. In the next section, we discuss the details of our proposed approach for event detection.
3.4 Proposed Event Detection Approach

In this work, we focus on event detection using the pipeline approach for the biomedical domain. Concretely, we formulate biomedical nested and overlapping event detection as a structured prediction task for DAG structures. We propose a novel neural search-based model that detects overlapping and nested events with beam search. Given a relation graph of trigger-argument relations, our model detects events by searching a sequence of actions that construct event structures incrementally in a bottom-up manner.

Our model maintains multiple beams and detects events from all the beams to detect overlapping and nested events unlike existing transition-based methods (Nivre, 2003, 2006; Chen and Manning, 2014; Dyer et al., 2015b; Andor et al., 2016). We define an LSTM-based neural network model that can represent nested event structure to choose actions.

Finally, we evaluated our model on the BioNLP Cancer Genetics (CG) Shared Task 2013 (Pyysalo et al., 2013b) using the predicted relations of the state-of-the-art TEES system (Björne and Salakoski, 2018a).

The main contributions of this work are:

1. Search-based method that performs better than the state-of-the-art TEES system in detection of nested and overlapping events represented in DAG structures

2. Novel LSTM-based neural model for nested and overlapping events

3. State-of-the-art performance on the BioNLP CG Shared Task 2013 without using any syntactic and hand-designed features, which are used in TEES, and with almost half the computational cost compared to TEES
3.4. PROPOSED EVENT DETECTION APPROACH

3.4.1 Common Model Processes

In this section, we discuss the common model processes that are shared by the two models which we will describe in the next sections (§3.4.2 and §3.4.3). Firstly, we describe the preprocessing involved to generate the candidate event structures from the relations. Since the previous process will produce irrelevant candidate structures, we developed a filter creation algorithm and template matching process to reduce the number of candidate structures. We also explain the bottom-up classification process for each event structures.

Preprocessing

We describe in this section the preprocessing we performed from the given pairwise relations. Figure 3.10 shows the series of preprocessing steps performed from the pairwise relations from the subset of the relation graph (i.e., involving E1 and E2 events).
in Figure 3.9. The first step involves the preparation of the input relations from gold events or the predicted relations from a relation extraction system or from the pipeline systems. To construct them from the gold events, we break apart each event structure into all the possible pairwise relations. The relations are pairwise relations composed of triples (trigger, role, argument). In Step 1 of Figure 3.10, we show the IDs of each trigger (i.e., starting with TR) and entities (i.e., starting with T), respectively. We then prepare the pairwise relations. For example, the first relation triple contains the following: TR1 which represents Positive Regulation (PR) as the trigger, Cause as the role, and TR which represents Gene-or-Gene-Product (GGP) as the argument. In the second relation, the argument BVD is a trigger itself, as shown in Figure 3.9. This shows an example of an event argument as represented by the trigger that is nested in another event structure.

The second step involves the creation of adjacency list representations. We create a list for each trigger and add to this list all the arguments that are associated with this trigger. For example, the list associated with TR1 trigger has as members all its arguments, concretely the three different arguments in Step 1, while the list for TR1 contains only one member.

The third step is the construction of the DAG structure (step 3 in Figure 3.10). We create a topological sorting of the given trigger-argument pairwise relations. A topological sort or topological ordering of a directed graph is a linear ordering of its vertices such that for every directed edge \(uv\) from vertex \(u\) to vertex \(v\), \(u\) comes before \(v\) in the ordering. The pairwise relation represents the directed edge and the trigger or argument represents the vertices or nodes. In the resulting DAG structure shown in Step 3, some arguments are triggers such as TR2.

The last step involves the template extraction and matching process. We extract from the training data all unique event structure patterns and we use them as templates
of valid event structures. The uniqueness of event structures is based on multiset representation because an event argument (i.e., (Role, Argument Type)) can appear multiple times in an event structure. For example, for events E1, E2 and E3 in Figure 3.9, we create the respective multiset representation under PR and BVD shown in Step 4. We extract templates for each event type. Thus, each event type \( T \) has its own set of templates \( F_T \). To perform template matching, we create a multiset representation \( A_m \) for each candidate event structure \( A \) of type \( T \). Then, we compare the multiset representation \( A_m \) to each \( F_T \) representation in the templates of type \( T \). If \( A_m \) does not correspond to any of the \( F_T \) structures, then it is discarded, otherwise, it is considered a valid event structure.

**Bottom-Up Event Structure Classification Process**

We describe in this section the event detection process which is a crucial part of our method. We detect events from the bottom level of the DAG structure going up. In this way, the representations of events in the lower level are used as arguments in the event at the upper levels.

**Algorithm 3: Bottom-up Event Structure Classification Process**

1. For each group \( g \) in groups \( G \) of the event structures of a sentence.
2. For each trigger-structures pair \( TS \) in group \( g \).
3. For each event structure \( A \) in structures \( TS \):
   4. Generate a combination of \( m \) new structures \( S \) for each of the sub-events in \( A \) that were predicted as events.
   5. If \( A \) has \( k \) sub-events and if \( \text{Sub}_{ij} \) is one of the sub-events which has \( j \) event predictions from previous bottom groups, then \( A \) is replicated \( j \) times wherein each time it has different sub-event \( \text{Sub}_{ij} \) prediction.
   6. For each structure \( S_m \) in \( S \), predict it using the THRESHOLD.
   7. If \( S_m \) prediction \( > \) THRESHOLD, then it is an EVENT, otherwise, it is a NON-EVENT.
   8. Else, set early update \( \text{flag} = True \) and proceed to next \( g_{TS} \).
   9. If \( \text{flag} = True \) at this group \( g \), then stop bottom-up prediction; else, continue to the next group \( g \).
Algorithm 3 describes the procedure for bottom-up event structure classification. We apply this to the output of the candidate generation process described in the previous section (§3.4.1). We start predicting the events at the bottom of the DAG structure, with $g = 0$ in Line 1. Each group can contain many trigger-structures pair $TS$ since this is a DAG structure. In turn, each pair $TS$ can contain many event structures. This is the case where there are many events associated with the same trigger and we refer to this as overlapping events.

Then for each event structure $A$, we check if any of its arguments are events. If there is an event argument which is represented by a trigger, then we check how many events have been predicted previously for this trigger. We then generate as many new structures equivalent to the number of event predictions for that trigger or event argument, as stated in Lines 4-5. In effect, the event structures at the higher levels of the DAG structure use the event representations of the predicted event structures at the lower levels. For example, in Figure 3.9, if we are currently predicting events under the trigger *influencing*, we check if any of its $k$ arguments are events. If at particular point in our algorithm we are considering both triggers arguments represented by *induction* and *angiogenesis*, we check how many events have been predicted for each of these triggers. Since in this case, there is only one event associated for *angiogenesis* trigger, which is $E1$, therefore, for *angiogenesis* argument event only one structure is generated. On the contrary, if for example the trigger *induction* has $j$ event predictions, then *influencing* has to be replicated $j$ times since each sub-event prediction is unique. This is what is described in Line 5 where we do this for each argument event of a prospective event structure $A$.

To make this idea clearer, let us consider another example as shown in Figure 3.11. The nodes in boxes are triggers represented by letters $a$, $b$, $c$. The leaves of the tree represented by $T_i$ are entities. Suppose we are predicting events on trigger $c$ and we are considering both $a$ and $b$ as the argument for the current event structure. We mention
3.4. PROPOSED EVENT DETECTION APPROACH

Figure 3.11: An illustration of the graph created for sub-events with multiple event predictions. Top: The DAG structure relation graph. Bottom: The expanded relation graph considering the number of event predictions for each sub-event argument.

this because it can be that at a particular point we are only considering one of the arguments. Moving on, suppose that for trigger $a$, two events were previously predicted which is shown in the bottom relation graph as $a_1$ and $a_2$ containing one entity respectively. Similarly, for trigger $b$, suppose that three events were previously predicted which is also shown in the bottom graph as $b_1, b_2, b_3$ containing one entity respectively. Each of the $a_i$s and $b_i$s is a unique event, hence the number of event structures when
constructing events with $a$ and $b$ arguments under trigger $c$ now totals to six with the following combinations $c = \{\{a_1, b_1\}, \{a_1, b_2\}, \{a_1, b_3\}, \{a_2, b_1\}, \{a_2, b_2\}, \{a_2, b_3\}\}$. However, if there was only one event predicted for each trigger $a$ and $b$ such as shown in the top relation graph, then there would only be one event structure to consider under trigger $c$. In general, the number of event structures to create under a trigger $t$ is the product of the number of events $j$ for $k$ sub-events: $\prod_{i=1}^{k} Sub_{ij}$.

For each generated structure $S_m$, we score it using a neural network. This part is slightly different in the search-based model as will be explained in §3.4.2 as what will be scored by the neural network is the partially-built event structure. If the prediction score for the current event structure is below a specified threshold, the prediction process stops, otherwise, the process continues until the last group $g$ of the event structures in the sentence.

3.4.2 Search-based Model

We describe our search-based neural network (SBNN) model that constitutes events from a relation graph by structured prediction. SBNN resembles an incremental transition-based parser (Nivre, 2006), but the search order, actions and representations are defined for DAG structures.

Structured Prediction for DAGs

We represent event structures with DAG structures and find them in a relation graph. A DAG structure, unlike a tree structure, can have multiple paths between two nodes. Our model performs beam search on relation graphs by choosing actions to construct events. Unlike existing beam search usage where they only choose the best path in the beam, e.g., Nivre (2006), we use all the beams to predict event structures, which enables the model to predict overlapping and nested events.
Event structures are searched and fixed for each trigger of the relation graph in a bottom-up manner. The model predicts the flat events first and then the representations of the flat events become the argument of the nested events. If the flat event is not detected, its nested event will not be detected consequently. When this happens, the search process stops. For example, in Figure 3.9, events $E_1$ and $E_2$ are flat events while $E_3$ is a nested event. With our proposed algorithm above, the flat events need to be predicted before the nested event is predicted.

To do the search, the model maintains two data structures: a buffer $B$ which holds a queue of arguments to be processed and a structure $S$ which contains the partially built event structure. A special NONE argument marks the first argument in the buffer.
to enable detection of no-argument events. The initial state is composed of the buffer $B$ with all the arguments and a structure $S$ empty. At each succeeding time step, the model applies a set of predefined actions to each argument and uses a neural network to score those actions. Processing completes when $B$ is empty and $S$ contains all the arguments for the trigger along with the history of actions taken by the model.

The concept of actions in this work is similar to the actions defined in the arc-standard dependency parsing algorithm (Nivre, 2008), however, we define a completely different set of actions for this work. The choice of our set of actions is based on what actions will enable the model to construct events.

We now define the three actions that the model applies at each time step to each argument, namely: add the argument (ADD), ignore the argument (IGNORE) and add the argument and construct an event candidate (CONSTRUCT). The only exception is for the NONE argument where only two actions are applied: IGNORE and CONSTRUCT. We have chosen only three actions for simplicity. The event candidates are fixed as events if the scores of the CONSTRUCT actions are above a certain threshold. The resulting state after a CONSTRUCT action is removed from the beams.

Figure 3.12 shows an illustration of the search procedure applied to a relation graph with a trigger induction. We omit in Figure 3.12 no-argument event handling, the initial state and the role types in $B$ for simplicity. Initially, the buffer contains all the arguments to be added and the structure is empty. At time step 1, we remove the argument TRAF2 from the buffer and apply the set of actions to the argument as shown in Figure 3.12. In the Figure 3.12 we only show two actions ADD and IGNORE. For each action applied a different structure is formed as shown in Figure 3.12. We score each structure using the network and we choose the $k$-best structures based on the score. At each succeeding timestep, we expand each of the $k$-best structures following the same procedure and so until there are no more arguments in the buffer. Figure 3.12 shows that after time step 2, we have detected an event structure after the CONSTRUCT
action (indicated by the dotted box). Any structure which is formed using the CON-STRUCT action is removed from the beam. If the removed structure has a score above a predefined threshold, it is added to the list of predicted event structures. Our model allows the construction of multiple events at every time step. This behaviour allows the construction of overlapping event structures, wherein multiple events share the same trigger.

**Neural Network as Scoring Function**

We use a neural network as the action scoring function \( \sigma(a_t|S_{t-1}, B_{t-1}) \). The state at any time step \( t \) is composed of the buffer \( B_t \) and the partially built structure \( S_t \). Each of \( S_t \) and \( B_t \) contains a set of relations \( \{r_1, r_2, r_3, \ldots, r_n\} \). We model this function using \( S_t \)
and \( B_t \), which are composed by adding an action \( a_t \) for a relation \( r_t \) to \( S_{t-1} \) and moving \( r_i \) from \( B_{t-1} \) to \( S_{t-1} \).

Figure 3.13 shows the proposed neural model. We employ a BiLSTM network to generate the word representations from pre-trained word embeddings. To represent phrases, we averaged the word representations. The LSTM network is shared among the states during search in the sentence. We build a relation embedding for each argument, which concatenates the information of the trigger \( t \), the role \( o \), the argument \( a \) and the action \( c \). We include both the type information \( p \) and the word or phrase representation \( w \) of the trigger or entity argument. Formally, each relation \( r_i \) is represented as a relation embedding:

\[
 r_i = [t_p; t_w; o_p; a_p; a_w; c] \tag{3.1}
\]

where \( t_p \) is the representation of the type of the trigger and so on.

Each \( r_i \) is passed to a linear hidden layer, then to a rectified linear unit (ReLU) non-linearity and summed to produce the structure and buffer embeddings: \( S_t \) and \( B_t \). \( S_t \) and \( B_t \) are then concatenated to form the event embedding. The event embedding has the same dimension as the sum of argument type and word dimensions so that it can be used as argument representation in nested events as shown in Figure 3.13. Then, we passed the event embedding into a linear hidden layer and output \( z_t \). Finally, the scoring function \( \sigma \) is calculated as follows:

\[
 \sigma(a_t|S_{t-1}, B_{t-1}) = \text{sigmoid}(z_t). \tag{3.2}
\]

**Training**

We train our model using the same input as the state-of-the-art system TEES (Björne and Salakoski, 2018a), where we decompose the gold events into pairwise relations and merged them with the predicted relations of TEES. We then generate a relation
graph from the merged relations and calculate gold action sequences that construct the
gold event structures on the graph. The loss is summed over all actions $A$ and for all
the events $E$ during the beam search and thus the objective function is to minimise
their negative log-likelihood. We employ early updates (Collins and Roark, 2004): if
the gold falls out of the beam, we stop searching and update the model immediately.

### 3.4.3 Baseline Model

We developed a neural network model using tree-LSTM as our baseline without search.
Figure 3.14 shows the model architecture. The input to this model are the candidate
structures associated to each trigger as generated in the previous step described in
§3.10.

![Diagram of Baseline Model Architecture](image_url)

Figure 3.14: Baseline model architecture
For each sentence, we generate the word representations using BiLSTM which is shared among all the event structures in the sentence. Then for each event structure, we represent each relation similar to the search-based model. We use relation representation $r_i$ without actions as its input. Concretely, we learn an IN or OUT embedding concatenated to each relation representation to indicate that the particular relation is part of the event structure. We concatenate the relation representations of an event structure.

Since this model does not consider actions and search order including the order of arguments, we employ a Child-Sum Tree-LSTM (Tai et al., 2015b) to represent the candidate structures. The Child-Sum Tree-LSTM transition equations are the following:

\[ \tilde{h}_j = \sum_{k \in C(j)} h_k, \quad (3.3) \]
\[ i_j = \sigma(W^{(i)}x_j + U^{(i)}\tilde{h}_j + b^{(i)}), \quad (3.4) \]
\[ f_{jk} = \sigma(W^{(f)}x_j + U^{(f)}h_k + b^{(f)}), \quad (3.5) \]
\[ o_j = \sigma(W^{(o)}x_j + U^{(o)}\tilde{h}_j + b^{(o)}), \quad (3.6) \]
\[ u_j = \tanh(W^{(u)}x_j + U^{(u)}\tilde{h}_j + b^{(u)}), \quad (3.7) \]
\[ c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k, \quad (3.8) \]
\[ h_j = o_j \odot \tanh(c_j), \quad (3.9) \]

where $C(j)$ denote the set of children of node $j$, $k \in C(j)$ in Equation 3.5, $x_j$ is the input at the current node $j$, $\odot$ denotes elementwise multiplication, $W$ and $U$ are trainable weight matrices, $b$ the bias, $\sigma$ the sigmoid function. Similar to standard LSTM units, each Tree-LSTM unit (indexed by $j$) contains input and output gates $i_j$ and $o_j$. 
a memory cell $c_j$ and hidden state $h_j$. As shown in Equations 3.4, 3.6, 3.8, the gating vectors and memory cell updates are dependent on the $k$ number of child units. Furthermore, the Tree-LSTM unit contains one forget gate $f_{jk}$ for each child $k$. This allows the Tree-LSTM unit to selectively incorporate information from each child (Tai et al., 2015a).

We create an event representation from the output of the Tree-LSTM. This event representation is passed through a multi-layer perceptron network with a non-linearity to produce a reduced representation. We then apply softmax to classify the resulting reduced representation. The structures are classified as event or not.

### 3.5 Conclusion

In this chapter, we have defined event structure and discussed its properties. We also examined the state-of-the-art approaches in event detection and analysed its limitations. We also introduced the novelty of our approach over the existing approaches.

In the next chapter, we will discuss the pipeline and model evaluation experiments and present the respective results and analysis.
Chapter 4

Biomedical Event Detection

Experiments

In this chapter, we discuss the experimental settings, namely, as part of the pipeline approach and as compared to the state-of-the-art approach and present the experimental results and analyses.

4.1 Pipeline Experiments

We describe in this section the experiments which we conducted to evaluate the performance of the models described in the previous chapter. In particular, we present the results of the pipeline approach. In the pipeline approach, the output of each component serves as input for the next component as shown in Figure 4.1. As we can see in Figure 4.1, event structure detection relies on the output of the relation detection and entity and trigger detection components.
4.1. PIPELINE EXPERIMENTS

4.1.1 Evaluation Scenarios

In this section, we explain the different scenarios on evaluating the event detection module. Concretely, in the next paragraphs, we describe the three (3) scenarios.

Scenario 1 is when we train and predict events using gold relations and entities. This is the scenario we will use to evaluate the model performance itself. However, if we want to compare our model performance with other models, we use Scenario 2.

Figure 4.2 shows the Scenario 2 where each module is trained on the gold dataset. In particular, event detection is trained using gold relations and entities. In contrast to Scenario 1 where we detect events using gold relations and entities, in Scenario 2 we predict the events using the predicted relations and entities. For example, Event-Mine (Miwa et al., 2013b) or TEES (Björne and Salakoski, 2018a) produces predicted relations which we use to evaluate our model performance by predicted events using these relations. This is the scenario when we want to compare the performance of the event detection model developed in this work with the state-of-the-art event detection models.

Figure 4.3 shows the Scenario 3 where each module is trained using the predictions from the previous model. As expected the event detection module on this scenario will have a lower performance since it will be trained using the predicted relations and entities. While this does not directly measure the performance of the event detection
module, it gives the lower bound performance of the event extraction process as only predictions are passed to the next module.

**Figure 4.3**: Scenario 3, each module is trained using predictions from previous module

### 4.1.2 Learning Curve Analysis on the Different Scenarios

We plot the learning curves of the baseline model on the different scenarios over number of epochs. This will give us an idea of how many epochs are needed to train the model which obtains its peak performance on the development set.

Figure 4.4 shows the learning curve on the development set. In this scenario (Scenario 1), the model is trained on gold relations and predicted on gold relations. We can see that the model started to overfit at around 3 epochs. The training F-score and
the test (or development) F1-score plateaus after 3 epochs. Using the gold relations to train the model, the performance on the development set gold relations is around 67.5 F1-score.

![Learning Curve: 0.5](image)

Figure 4.4: Learning curve analysis on the development set using Scenario 1 with Threshold = 0.5

Figure 4.5 shows the learning curve of the model on Scenario 2. In this scenario, the model is trained on gold relations and evaluated on predicted relations. We use the predicted relations of Eventmine (Miwa et al., 2013b). EventMine is a machine learning-based pipeline system, which extracts events from documents that already contain named entity annotations (e.g., genes/proteins, etc.). The core system consists of four (4) detection modules, which operate on the output of syntactic parsers. The four modules are Trigger/Entity Detection, Argument Detection, Multi-argument

---

Event Detection and Modification Detection.

We can see that the performance is similar to Scenario 1. Since in both scenarios (1 and 2) the model is trained on gold relations, the prediction performance is expected to be the same.

Figure 4.6 shows the learning curve for Scenario 3 where the model is trained on predicted relations and evaluated on the predicted relations as well. We use the predicted relations of Eventmine (Miwa et al., 2013b) in both the training and development set. We can see that the development set F1-score (test_fscore in Figure 4.6) that the scores are comparatively lower than the previous two scenarios. This is expected because the model is trained on predicted relations. The model takes 6 (six) epochs before the model’s performance on the training set becomes greater than the performance...
4.1. PIPELINE EXPERIMENTS

Figure 4.6: Learning curve analysis on the development set using Scenario 3 with Threshold = 0.5 on the development set, which is a little longer than the previous two scenarios. Most likely this is because the model does not have enough event structure information in order to learn from the predicted relations.

4.1.3 Results and Analysis

In this section, we present the performance of the event detection models on a pipeline approach. We use the models of Ju et al. (2018) and Christopoulou et al. (2018) to extract the entities and relations, respectively. We perform the experiments on the CG dataset (Pyysalo et al., 2015). The prediction threshold for the baseline and search-based model is set to 0.5.

Table 4.1 shows the performance of the event detection models on the gold relations
of the development set. We can see that the scores are high because the models are trained on gold relations and evaluated on gold relations as well. Between the models, the search-based model performed better in F1-score than the baseline. In particular, the search-based model has a higher recall.

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>97.62</td>
<td>87.45</td>
<td>92.25</td>
</tr>
<tr>
<td>Search-based</td>
<td>97</td>
<td>89.79</td>
<td>93.26</td>
</tr>
</tbody>
</table>

Table 4.1: Model performance comparison on the development using Scenario 1.

Table 4.2 shows the performance comparison of the models when they are trained on gold relations and evaluated on predicted relations of the development set. These predicted relations were output of a relation extraction system (Christopoulou et al., 2018) trained on gold entities. We can see that the scores are lower than in Scenario 1 because in the scenario the event detection models rely on predicted relations.

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>56.9</td>
<td>53.31</td>
<td>56.28</td>
</tr>
<tr>
<td>Search-based</td>
<td>59.26</td>
<td>53.84</td>
<td>56.42</td>
</tr>
</tbody>
</table>

Table 4.2: Model performance comparison on the development using Scenario 2.

Table 4.3 shows the performance of the event detection models on Scenario 3 where the pipeline components are trained on the predictions of previous components. This scenario results in the lowest scores on the event detection component as expected. Between the models, the search-based model yields a higher F1-score.
4.2. **MODEL EVALUATION EXPERIMENTS**

In this section, we describe the experimental settings as we compare our event detection models with the state-of-the-art. We applied our model to the BioNLP CG shared task 2013 (Pyysalo et al., 2015). We used the original data partition by the shared task and employed the official evaluation metrics, i.e., precision, recall, and F1 scores, after removing illegal structures using the rules in the official evaluation script. To make a fair comparison, we run our model on the predicted relations of the best performing system TEES (Björne and Salakoski, 2018a), which is a pipeline system that uses convolutional neural network and depends on a dependency parser and hand-designed features for event detection. We used their published single models. In contrast to the ensemble methods, this enables us to make a direct comparison with TEES in a minimal setting.

### 4.2.1 Dataset

We apply our models on the CG dataset (Pyysalo et al., 2015). We extract the predicted relations using TEES (Björne and Salakoski, 2018a). We train our models using both the gold and predicted relations to follow TEES (Björne and Salakoski, 2018a) training setting. The training set contains 9,422 gold events and of which 384 (4.07%) are intersentence events which we removed since we focus our work on sentence-level event detection. The development set contains 3,219 gold events and of which 100 (3.10%) are intersentence events which we also removed.

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>56.21</td>
<td>33.32</td>
<td>41.84</td>
</tr>
<tr>
<td>Search-based</td>
<td>56.56</td>
<td>34.03</td>
<td>42.5</td>
</tr>
</tbody>
</table>

Table 4.3: Model performance comparison on the development using Scenario 3.
The development set contains 3,217 events of which 36.46% are nested events, 43.05% are overlapping events and 44.07% are flat events. Note that the total does not equal to 100% because nested and overlapping events may have intersection, meaning, a nested event can be overlapping and vice versa.

We also plot the distribution of event types with particular number of arguments since our search-based model applies actions to each argument at every time step. The more number of arguments, the more number of time steps the search-based model needs to learn an event structure. Figure 4.7 shows a heatmap indicating the number of event type instances in the gold training set with a particular number of argument. Darker shades mean that it has relatively high number of instances which is indicated by the number.

For example, Figure 4.7 shows that Positive Regulation event type has 1,047 instances which can be broken down into the following: zero (0) instances with zero (0) argument, 526 instances having one argument and 521 instances having 2 arguments. We can observe that most of the event types have at least one argument. Furthermore, there are specific event types that have both one and two arguments such Positive Regulation, Negative Regulation, Regulation. Some event types mostly appear as one-argument events such as Gene Expression, Growth, Cell Proliferation, Cell Transformation, etc. Figure 4.7 also shows an outlier event structure of type Gene Expression which has one occurrence with 5 arguments.

Figure 4.8 shows the distribution of event structures generated from predicted relations on the training set. We can observe that compared to the event structures from the gold relations in Figure 4.7 the event structures distribution in Figure 4.8 follows the same pattern in terms of the distribution of event types occurrences but in a slightly more diffused manner, that is, over many argument counts. For example, while in Figure 4.7 the event structure instances of Positive Regulation appear only as having one or two arguments, in Figure 4.8 Positive Regulation has instances with zero (0) to seven
### 4.2. MODEL EVALUATION EXPERIMENTS

<table>
<thead>
<tr>
<th>Argument count</th>
<th>Urubituation</th>
<th>Translation</th>
<th>Transcription</th>
<th>Synthesis</th>
<th>Reproduction</th>
<th>Remodeling</th>
<th>Regulation</th>
<th>Protein_processing</th>
<th>Positive_regulation</th>
<th>Planned_process</th>
<th>Pathway</th>
<th>Phosphorylation</th>
<th>Processed_pathway</th>
<th>Protein_processing</th>
<th>Regulation</th>
<th>Remodeling</th>
<th>Reproduction</th>
<th>Synthesis</th>
<th>Transcription</th>
<th>Translation</th>
<th>Ubiquitination</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 4.7: Distribution of event structures according to number of arguments in the gold training set (gold relations).

(7) arguments though most of them are still in the 1-2 arguments, which is expected. The reason for this deviation from the gold relations is that this dataset contains false positive relations.

Figure 4.9 shows the distribution of event structure instances on the development set using the gold relations. We can observe the training (Figure 4.7) and development set (Figure 4.9) have a relatively similar distribution of event types and the occurrences of event structures with a particular number of arguments. However, we can clearly see that the event type distribution is imbalanced in both data partitions. Figure 4.10 shows the distribution of event structures using the predicted relations on the development set. We can make the same observation as Figure 4.8 on the training set using the predicted relations.

We employ template matching (described in §3.4.1), built from training data, as preprocessing to reduce the number of candidate structures. The number is huge even
after filtering as shown in §4.2.3. The training set contains 65,757 unique tokens and we have determined that the development set contains 10.79% of unknown tokens.

### 4.2.2 Training Details and Model Parameters

We implemented our model using the Chainer library (Tokui et al., 2015). We initialised the word embeddings using pre-trained embeddings (Chiu et al., 2016a) while other embeddings are initialised using the normal distribution. All the embeddings and weight parameters were updated with mini-batch using the AMSGrad optimiser (Reddi et al., 2018). We also incorporated early stopping to choose the number of training epochs and tuned hyper-parameters (dropout, learning rate and weight decay rate) using grid search. The table 4.4 shows the hyper-parameters used in our experiments.
4.2. MODEL EVALUATION EXPERIMENTS

4.2.3 Results and Analysis

In this section, we present the results of our experiments. Table 4.5 shows the performance comparison. SBNN yields the state-of-the-art performance in both the development and test sets without using any syntactic and hand-designed features. We validated that it has no significant statistical difference with the TEES model (the Approximate Randomisation test (Yeh, 2000; Noreen, 1989)).

Table 4.6 shows the number of classifications (or action scoring function calls in our model) performed by each model on the development set. SBNN requires almost half the computational cost than TEES and 6 times less than the baseline. SBNN is thus more computationally efficient than TEES and the baseline.

Table 4.7 shows the performance comparison of the models on nested and overlapping events. We used the official evaluation script to measure the performance of
Figure 4.10: Distribution of event structures according to number of arguments in the predicted development set (predicted relations).

the model on nested, overlapping and flat events. We first separated the nested, overlapping and flat events, respectively. Then we compute the precision and recall for each category in the following way. For example, for nested events, to compute the precision, we compare the predicted nested events with all gold events and to compute recall, we compare gold nested events with all predicted events. The evaluation script detects nested events by comparing the whole tree structure down to its sub-events until it reaches the flat events. Hence, the performance scores of the nested events inevitably include the performance on flat events.

Results show that both SBNN and the baseline outperformed TEES consistently, confirming that our neural-based models can efficiently capture nested and overlapping event structures better. The baseline performs slightly better than SBNN as expected since it is an exhaustive method, however, it requires more computation (Table 4.6) and its overall performance is less than SBNN (Table 4.5).
4.2. MODEL EVALUATION EXPERIMENTS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value/Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Common Parameters</strong></td>
<td></td>
</tr>
<tr>
<td>Mini-Batch Size</td>
<td>100</td>
</tr>
<tr>
<td>Word Embedding</td>
<td>200</td>
</tr>
<tr>
<td>BiLSTM Word Embedding</td>
<td>100</td>
</tr>
<tr>
<td>Role Type Embedding</td>
<td>10</td>
</tr>
<tr>
<td>Trigger/Argument Type Embedding</td>
<td>20</td>
</tr>
<tr>
<td>Early Stopping Patience</td>
<td>5</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.5</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>0.001</td>
</tr>
<tr>
<td>Hidden Layer Size</td>
<td>60</td>
</tr>
<tr>
<td>Event Embedding</td>
<td>100</td>
</tr>
<tr>
<td><strong>Baseline Model</strong></td>
<td></td>
</tr>
<tr>
<td>I/O Embedding</td>
<td>2</td>
</tr>
<tr>
<td>Event Prediction Threshold</td>
<td>0.5</td>
</tr>
<tr>
<td>Weight Decay Rate</td>
<td>0</td>
</tr>
<tr>
<td><strong>Search-based Model</strong></td>
<td></td>
</tr>
<tr>
<td>Action Score Threshold</td>
<td>0.5</td>
</tr>
<tr>
<td>Beam Size</td>
<td>8</td>
</tr>
<tr>
<td>Action Embedding</td>
<td>4</td>
</tr>
<tr>
<td>Weight Decay Rate</td>
<td>0.001</td>
</tr>
</tbody>
</table>

Table 4.4: Hyper-parameters used in our experiments.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dev Set (%)</th>
<th>Test Set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>TEES</td>
<td>56.81</td>
<td>48.21</td>
</tr>
<tr>
<td>Baseline</td>
<td>61.88</td>
<td>45.57</td>
</tr>
<tr>
<td>SBNN</td>
<td>60.5</td>
<td>47.19</td>
</tr>
</tbody>
</table>

Table 4.5: Comparison of event detection performance.

Since we use all the beams in prediction, it is not clear how the beam size affects the performance. Hence, we conducted experiments on our model by varying the $k$-best parameter in beam search on the development set. Table 4.8 shows the performance of SBNN. We tried $2^i$ values for $i = 1, 2, 3, \ldots, 11$ and found that the best value was 8 with F1-score of 54.36%, which is 2.2 percentage points (pp) higher than TEES.
### Table 4.6: Comparison on computational efficiency based on the number of classifications.

<table>
<thead>
<tr>
<th>Model</th>
<th>Number of Classifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEES</td>
<td>6,141</td>
</tr>
<tr>
<td>Baseline</td>
<td>25,766</td>
</tr>
<tr>
<td>SBNN</td>
<td>4,093</td>
</tr>
</tbody>
</table>

### Table 4.7: Comparison among models in nested and overlapping event detection F1 score performance.

<table>
<thead>
<tr>
<th>Model</th>
<th>Nested</th>
<th>Overlap</th>
<th>Flat</th>
<th>Overall F1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEES</td>
<td>42.70</td>
<td>34.49</td>
<td>56.81</td>
<td>52.16</td>
</tr>
<tr>
<td>Baseline</td>
<td>47.14</td>
<td>37.85</td>
<td>61.90</td>
<td>52.49</td>
</tr>
<tr>
<td>SBNN</td>
<td>45.24</td>
<td>36.92</td>
<td>60.5</td>
<td>53.02</td>
</tr>
</tbody>
</table>

### Table 4.8: Event detection performance on CG task 2013 development set for SBNN with the top three performing $k$-values and when $k = 1$.

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEES</td>
<td>56.81</td>
<td>48.21</td>
<td>52.16</td>
</tr>
<tr>
<td>SBNN $k = 1$</td>
<td>48.95</td>
<td>43.79</td>
<td>46.23</td>
</tr>
<tr>
<td>SBNN $k = 8$</td>
<td>63.60</td>
<td>47.46</td>
<td>54.36</td>
</tr>
<tr>
<td>SBNN $k = 64$</td>
<td>60.30</td>
<td>49.16</td>
<td>54.17</td>
</tr>
<tr>
<td>SBNN $k = 256$</td>
<td>60.91</td>
<td>48.53</td>
<td>54.02</td>
</tr>
</tbody>
</table>

Since our event predictions are based on the predictions of TEES, our model cannot achieve the recall of 100%. The upper bound is computed by setting the threshold parameter of our model to zero, which then constructs all gold events possible from the predicted relations of TEES. We have determined the upper bound recall to be 53.47% (6.28pp higher than our current model’s score).

To close this gap, besides improving our model’s complexity, inter-sentence or self-referential events, which together accounts 3.1% of the total events, need to be
4.3 Conclusions and Future Work

We proposed a novel search-based neural network (SBNN) model for nested and overlapping event detection by treating the task as structured prediction for DAGs. We defined actions to construct events and we detected events in a bottom-up manner. We also developed a baseline model using Tree-LSTM. We applied our model on the CG Shared Task dataset and compared with the state-of-the-art event detection model of TEES.

Our new approach achieved the state-of-the-art performance without using any syntactic and hand-designed features. Furthermore, it is almost twice as computationally efficient than the current state-of-the-art TEES model and 6 times more than the baseline model based on the numbers of classifications. Moreover, it captures nested and overlapping event structures as well as the flat events better than TEES. Furthermore, the baseline model also outperformed the TEES model in nested and overlapping event detection, however, it requires more classifications than either TEES model or the search-based model.

Since SBNN is not limited to events, future work includes applying it to other DAG structures such as nested/discontiguous entities (Muis and Lu, 2016; Ju et al., 2018). Moreover, since the model is domain-independent, it can applied to open domain nested event extraction.

In this chapter, we have presented the experimental settings and results of neural models that can detect biomedical event structures in the sentence-level and generate their representations. As an application, we investigate in the next chapters to what extent these event representations can be used as sentence representation and how effective they are when evaluated specifically on biomedical textual similarity task. We
start in the next chapter with the survey of the state-of-the-art approaches in sentence representation.
Chapter 5

Sentence Representation

(The content in this chapter is an extended version of the author’s publication (Espinosa, 2017) published in Association for Computational Linguistics (ACL) Student Research Workshop (SRW) in 2017.)

In this chapter, we will provide an overview of text semantics modelling in general and the current techniques focussing on sentence modelling. We examine different approaches in the general and biomedical domains. We survey and examine the spectrum of these methods, which lies along two dimensions: input representation granularity and compositional model complexity.

5.1 Overview

Accurately representing the meaning of a piece of text, otherwise known as semantics modelling, is an important and central component in many natural language inference tasks, and it still remains an open problem. In this work, we focus on the sentence level and so we refer to this task as sentence modelling. To illustrate why sentence modelling is difficult, consider the pair of sentences $A$ and $B$ below in the context of a sentence similarity task.
CHAPTER 5. SENTENCE REPRESENTATION

A: The shares of the company dropped.
B: The organisation’s stocks slumped.

If we use a very naïve model such as bag-of-words to represent a sentence and use discrete counting of common words between the two sentences to determine their similarity, the score would be very low as they only share one word although they are highly similar.

While tackling the sentence modelling task, an unavoidable question presents itself: what constitutes sentence meaning? Unsurprisingly, due to the subject’s inherently interdisciplinary nature (Davidson, 1967; Johnson-Laird and Byrne, 2002; Uexkull, 1982; Fillmore, 1975), it is nearly impossible to arrive at a single definition. Without sidestepping the main question, we aim to shed light on this problem by pursuing an empirical investigation grounded on Frege’s principle, also known as the Principle of Compositionality, following previous work (Szabó, 2013; Bowman, 2016).

In this study, we adopt the definition from Szabó (2013) where $L$ is a natural language: “For every complex expression $e$ in $L$, the meaning of $e$ in $L$ is determined by the structure of $e$ in $L$ and the meanings of the constituents of $e$ in $L$”. Following Bowman (2016), we define the following as constituents of $e$: words with no decomposable parts (e.g., “cat”, “London”), morphemes (meaningful word parts, e.g., “un-”, “afford”, “-able”), and multi-word expressions (e.g., “Bob’s your uncle”, “drop them off”, “Premier League”). Furthermore, the meanings of the constituents are given by the lexical semantics of $L$ while the structure and constituency by the syntax of $L$. Finally, the compositional semantics of $e$ is the combination of the syntax and lexical semantics (Szabó, 2013).

How then do we represent the meaning of sentences? In order to perform computation on language, we must be able to represent them in ways that computers can understand. According to the Principle of Compositionality, the meaning of a whole is a function of the meanings of its parts. Hence, to represent a given complex expression
(e.g., sentence), we need to represent its constituents (e.g., word, phrases, morphemes).

There are generally two main approaches to representing constituents: local and distributed representations. With local representations, each constituent has a unique representation usually taken from its position in a vocabulary $V$. The best example of this sparse representation is the one-hot\(^1\) representation.

Formally, if $|V|$ is the size of the vocabulary $V$, the representation of $\text{word}_i$ is a one-dimensional vector of real numbers $\mathbb{R}^{|V|\times 1}$ with $0$s at all indices $j$ where $j \neq i$,

$$w_i = [0_1, 0_2, \ldots, 1_i, \ldots, 0_{|V|}]$$  \hspace{1cm} (5.1)  

For example, assuming “aardvark” and “zebra” are the first and last words in our vocabulary, we have: $w_{\text{aardvark}} = [1, 0, 0, 0, \ldots, 0]$ and $w_{\text{zebra}} = [0, 0, 0, 0, \ldots, 1]$.

However, this representation does not supply us with any notion of similarity, often computed using measures such as cosine similarity\(^2\) which in this case, results in either $0$ (dissimilar) or $1$ (similar). Assuming the terms “human” and “person” are similar, we would want their similarity value to be closer to $1$; however, computing their cosine similarity yields: $(w_{\text{person}})^T w_{\text{human}} = 0$.

Local representations suffer from dimensionality issues referred to as curse of dimensionality (Bellman, 1966) leading to sparse representations. Furthermore, they do not consider the syntactic relationship of a constituent with other constituents as word order is not considered, and thus cannot capture precisely the overall semantics of a given expression. Consequently, this kind of representation does not fit into the Principle of Compositionality.

These shortcomings are addressed by distributed representations (Hinton, 1984)

\(^1\)The term “one-hot” comes from digital circuit design which means a group of bits among which the legal combinations of values are only those with a single high (1) bit and all the rest low (0).
\(^2\)Computed as the Euclidean inner product of the vectors.
which consider the compositionality of an expression by taking into account the relationship of the constituents with one another. Additionally, this addresses the sparse representation problem by encoding a constituent based on its co-occurrence with other constituents appearing within its context. This leads to a denser $n$-dimensional vector where $n \ll |V|$.

5.1.1 Distributed Representations and Distributional Semantics

Distributional semantics have been used to represent constituents in a distributed manner by means of a particular function applied to a set of co-occurrence statistics. In other words, distributional semantic models (DSMs) assume that words that occur in similar contexts tend to have similar meanings. DSMs have been inspired by the concept of collocation by Firth (1957), “You shall know the word by the company it keeps”.

Estimating distributed representation has traditionally been done by initialising vectors with co-occurrence counts and then reweighting these vectors based on various criteria (Turney et al., 2010). However, recent developments cast this problem as a supervised task, where the weights in the word vector are directly set to values that optimally predict the contexts in which the corresponding words tend to appear (Bengio et al., 2003; Mikolov et al., 2013a).

Baroni et al. (2014) conducted a systematic comparative evaluation of context-counting and context-predicting models for generating distributed representations and concluded that the latter outperforms the former. Levy et al. (2015) later have shown that simple pointwise mutual information (PMI) methods also perform similarly if they are properly tuned.

The most popular architectures to efficiently estimate these distributed representations are \textit{word2vec} (Mikolov et al., 2013a) and \textit{GloVe} (Pennington et al., 2014).
5.1. OVERVIEW

More recently, sub-word and subsequently both word and sentence representations are learned using deeper architectures that are pre-trained on large amount of unlabeled text. Such methods include ELMo (Embeddings from Language Models) (Peters et al., 2018) and BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019b) and its variations (Wang et al., 2019; Yang et al., 2019) which have been successful in many natural language understanding task (Wang et al., 2018a). Subsequent developments estimate distributed representations at different levels of granularity (see Section 5.3.1).

5.1.2 Sentence Encoding

While much research has been directed into constructing representations for constituents, there has been far less consensus regarding the representation of larger semantic structures such as phrases and sentences (Blacoe and Lapata, 2012b). A simple approach is based on looking up the vector representation of the constituents (i.e., embeddings) and taking their sum or average which yields a single vector of the same dimension.

This strategy is effective in simple tasks but loses word order information and syntactic relations in the process (Mitchell and Lapata, 2008; Turney et al., 2010). Most modern neural network models have a sentence encoder that learns the representation of sentences more efficiently while preserving word order and compositionality information.

These models have three basic forms: (1) spatial-based, e.g., Convolutional Neural Networks (CNN) (LeCun et al., 1998); (2) sequence-based, e.g., Recurrent Neural Networks (RNN) (Elman, 1991; Sutskever et al., 2014), Long Short Term Memory (LSTM) models (Hochreiter and Schmidhuber, 1997); (3) and Tree-based, e.g., Recursive Neural Networks (RecNN) (Goller and Kuchler, 1996; Socher et al., 2011b).

More advanced models use stacked components of these basic models, pooling
mechanisms to induce abstract representations (He and Lin, 2016; Yin et al., 2015),
and attention mechanisms (Bahdanau et al., 2014) which have been extended as deep
bidirectional networks in the case of BERT-related models (Peters et al., 2018; Devlin
et al., 2019a; Yang et al., 2019; Wang et al., 2019). We will discuss the neural network
architectures in more detail in Section 5.3.2.

5.2 Natural Language Inference Tasks

An efficient way to evaluate sentence meaning representations is through their per-
formance on actual natural language inference (NLI) tasks in different domains. In
this work, we examine three related tasks (see Table 5.1) which are central to natural
language understanding: paraphrase detection, semantic similarity measurement and
interpretable semantic similarity measurement.

<table>
<thead>
<tr>
<th>Task</th>
<th>Benchmark Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General Domain</strong></td>
<td></td>
</tr>
<tr>
<td>Paraphrase Identification (PI)</td>
<td>MSR Par 2004, ACL PIT 2015</td>
</tr>
<tr>
<td>Semantic Textual Similarity (STS)</td>
<td>SICK 2014, ACL PIT 2015, ACL STS 2012 - 2016</td>
</tr>
<tr>
<td>Interpretable Semantic Textual Similarity (iSTS)</td>
<td>ACL iSTS 2016</td>
</tr>
<tr>
<td><strong>Biomedical and Clinical Domain</strong></td>
<td></td>
</tr>
<tr>
<td>Semantic Textual Similarity (STS)</td>
<td>BIOSSES 2017, MedSTS 2018</td>
</tr>
</tbody>
</table>

Table 5.1: NLI tasks and corresponding benchmark datasets

These tasks have been the focus of a number of shared tasks that have been or-
organised over the recent years leading to the availability of benchmark datasets (Dolan
et al., 2004; Marelli et al., 2014; Xu et al., 2015; Agirre et al., 2016b,c).

5.2.1 Semantic Similarity Measurement vs Paraphrase Detection

While there are overlaps between the tasks of semantic similarity measurement and
paraphrase detection, we distinguish them from each other. In both cases, a system is
given as input a pair of sentences A and B, and is required to return a judgement of the
input sentences’ similarity. If the range of the judgement is binary (e.g., yes or no), we
cast the task as paraphrase detection. Otherwise, if a judgement is expressed in terms
of continuous or ordinal values, we treat the task as a semantic similarity measurement
problem.

An “interpretable” variant of this task has also been introduced recently, formalised
as the alignment of textual chunks in the two input sentences. In this case, a similarity
score and a label indicating the type, e.g., equivalent, opposite, more-specific, etc) are
required for each alignment Agirre et al. (2016c).

To illustrate the type of analysis required by each of these different tasks, we
present the following sentences A and B (derived and expanded from Androutsopoulos
and Malakasiotis (2010)).

A: Trump’s win caused the shares of the parent company in the UK to drop unex-
  pectedly.

B: The stocks of the London-based organisation slumped unexpectedly after the US
election.

Sentence A is likely to be recognised as a paraphrase of sentence B (and vice-
versa) by paraphrase detection systems, while tools for semantic similarity measure-
ment could give a score of 3 as an output (using the ordinal similarity scale proposed
by Agirre et al. (2016b)), implying that the sentences can be considered as roughly
equivalent but different in terms of some important pieces of information. Both sen-
tences pertain to two events happening one after the other but sentence A presents
them as having a cause-and-effect relationship (i.e., that Trump’s win led to the unex-
pected drop) while sentence B simply presents an observation (i.e., that the two events
happened one after the other). If the two sentences are analysed based on their “inter-
pretable” semantic similarity (according to the annotation guidelines of Agirre et al.
(2016c)), their chunks can be aligned as shown in Figure 5.1.
Figure 5.1: Sample manual alignment of two sentences according to the schema proposed by Agirre et al. (2016c): “Trump’s win caused the shares of the parent company in the UK to drop unexpectedly.” and “The stocks of the London-based organisation slumped unexpectedly after the US election.”. Each aligned pair of chunks includes information on alignment type and score.

Table 5.1 presents known benchmark datasets for each of these tasks, with further details described in the next section.

### 5.2.2 Benchmark Datasets

In the general domain, there are five benchmark datasets that support the evaluation of different methods for paraphrase detection and semantic similarity measurement and two datasets in the biomedical and clinical domain for sentence similarity measurement task. Table 5.2 presents the characteristics of each dataset. It is worth noting that there has been a recent surge of interest in this task judging from the fact that almost all of these datasets have been developed only within the last five years.

Undoubtedly, such resources are very expensive to create. Often, completely manual annotation efforts are required, the cumbersome and overwhelming nature of which could potentially lead to annotation inconsistencies. Thus, to evaluate the reliability of some of these datasets, agreement between human annotators (i.e., inter-annotator agreement or IAA) was measured. Developers of other datasets, however, did not explicitly assess IAA and instead, reported only control measures to ensure data quality.
### 5.2. Natural Language Inference Tasks

#### Dataset Details

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of Sentence Pairs</th>
<th>Text Type</th>
<th>IAA / Human Baseline</th>
<th>Evaluation Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General Domain</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSR Par 2004 (Dolan et al., 2004)</td>
<td>5,800</td>
<td>news</td>
<td>AER = 0.931, F = 0.8848</td>
<td>F1-score</td>
</tr>
<tr>
<td>SICK 2014 (Marelli et al., 2014)</td>
<td>9,927</td>
<td>image captions, video descriptions</td>
<td>None (SD = 0.76)</td>
<td>Pearson correlation</td>
</tr>
<tr>
<td>PIT 2015 (Xu et al., 2015)</td>
<td>18,762</td>
<td>tweets</td>
<td>P = 0.735 (SS), F = 0.823 (PI)</td>
<td>Pearson correlation</td>
</tr>
<tr>
<td>STS 2016 (Agirre et al., 2016b)</td>
<td>15,436</td>
<td>multi-source (i.e., news, student answers, post-edited translations, tweets, MSR Par, etc)</td>
<td>None (annotators with correlation scores &lt; 0.80 removed and kappa score &lt; 0.20 excluded)</td>
<td>Pearson correlation</td>
</tr>
<tr>
<td>iSTS 2016 (Agirre et al., 2016c)</td>
<td>2,927</td>
<td>headlines, image captions, student answers</td>
<td>None</td>
<td>F1-score</td>
</tr>
<tr>
<td><strong>Biomedical/Clinical Domain</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIOSSES 2017 (Soğancıoğlu et al., 2017)</td>
<td>100</td>
<td>PubMed articles</td>
<td>P = 0.902 (Kappa Score)</td>
<td>Pearson correlation</td>
</tr>
<tr>
<td>MedSTS 2018 (Wang et al., 2018b)</td>
<td>1,068</td>
<td>clinical notes</td>
<td>P = 0.67 (Kappa Score)</td>
<td>Pearson correlation</td>
</tr>
</tbody>
</table>

Table 5.2: Details on the benchmark datasets with five (5) datasets from general domain and one (2) datasets from the biomedical/clinical domain. Acronym definitions: Microsoft Paraphrase (MSR Par 2004), Sentences Involving Compositional Knowledge (SICK) 2014, Paraphrase Identification in Twitter (PIT) 2015, Semantic Textual Similarity (STS) 2016, and Interpretable Semantic Textual Similarity (iSTS) 2016, Biomedical Sentence Similarity Estimation (BIOSSES) 2017, Medical Semantic Textual Similarity (MedSTS) 2018.

#### State-of-the-Art Methods for Measuring Semantic Similarity

On each of the benchmark datasets, we present in Figure 5.2 the five best performing—thus, state-of-the-art—semantic similarity measurement systems, based on the performance that has been reported in the literature. We then identified four broad types
CHAPTER 5. SENTENCE REPRESENTATION

according to which these systems can be grouped, namely: machine learning (ML),
deep learning (DL), and rule-based.

![Diagram showing performance comparison of top 5 systems for each dataset]

Figure 5.2: Top 5 performing systems on each dataset (in descending order of F1-score or Pearson correlation). **MSR Par 2004** (Wang et al., 2019; Devlin et al., 2019a; Ji and Eisenstein, 2013; Cheng and Kartsaklis, 2015; Yin et al., 2015); **SICK 2014** (Mueller and Thyagarajan, 2016a; He and Lin, 2016; Hashimoto et al., 2016; Zhou et al., 2016; He et al., 2015); **PIT 2015-PI** (Vosoughi et al., 2016; Eyecioglu and Keller, 2015a,b; Zarrella et al., 2015a; Zhao et al., 2014); **PIT 2015-SS** (Zarrella et al., 2015b,c; Bicici, 2015a,b; Bertero and Fung, 2015); **STS 2016** (Yang et al., 2019; Devlin et al., 2019a; Rychalska et al., 2016; Brychcin and Svoboda, 2016; Afzal et al., 2016); **iSTS 2016** (Banjade et al., 2016a,b,c; Magnolini et al., 2016a,b); **BIOSSES 2017** (Peng et al., 2019a,b; Blagec et al., 2019; Peng et al., 2019c; Chen et al., 2019); **MedSTS 2018** (Peng et al., 2019a,c,b; Lee et al., 2019; Chen et al., 2019).

Methods driven by machine learning are underpinned by standard classifiers that make predictions based on hand-crafted features while deep learning methods combine feature learning and classification in one task. Rule-based approaches use a set of rules, similarity functions and lookup resources—which are all usually hand-crafted—in computing similarity scores.

To enable us to draw a rough performance-wise comparison of these four method types, we categorised the best performing systems in Figure 5.2 accordingly (cf. figure key). We can observe that some datasets are dominated by particular method types.
in terms of their proportion within the top five systems, namely, DL methods in SICK 2014, MSR Par 2014, BIOSSES 2017 and MedSTS 2018; ML methods in PIT datasets and STS 2016; and rule-based methods in iSTS 2016. Based on the proportion in the overall forty systems across datasets, the DL-based methods account for 52.5% (21 of 40). Next are the ML-based systems which comprise sixteen systems (or 40%) and the least are the rule-based systems of which there are only three.

In terms of the state-of-the-art model, six out of the eight datasets use DL-based systems, which suggests that this method works well in general over the other methods. Interestingly, the DL method has surpassed the human baseline performance on the MSR Par 2014 corpus. Furthermore, the DL method outperformed the ML methods by a large margin in the STS 2016 dataset. The state-of-the-art models for the two datasets, MSR Par 2004 and STS 2016, can be attributed to the use of representations trained using BERT method or its variants (Wang et al., 2019; Yang et al., 2019).

In general, deep learning methods have obtained better performance relative to ML-based ones (as evidenced by the number of top-ranking systems which are DL-based) in spite of not requiring domain-specific knowledge. They use few hand-crafted features—or none at all—while the ML-based ones require their design which is typically informed by extensive and comprehensive domain knowledge. As a result, non-DL methods have become more difficult to adapt to new domains while DL methods are embraced by increasingly more members of the NLP research community.

We recognise that the different systems are not directly comparable in terms of their performance as they were evaluated for different tasks, on different datasets. However, we can make some qualitative observations across tasks. For example, the best systems addressing the interpretable Semantic Textual Similarity (iSTS) task performed poorly relative to those for other tasks. Agirre et al. (2016c) reported that identifying alignment types between chunks was the most challenging aspect of iSTS. Indeed, as they have observed, participants with higher scores in this aspect of the task performed
better overall.

The next most difficult task is Paraphrase Identification in Twitter (PIT) where the performance of the top systems are much lower than the human baseline. It can also be observed that based on the scores, the PIT Semantic Similarity (SS) subtask is more difficult than the PIT Paraphrase Identification (PI) subtask, which can be attributed to the finer-grained requirements of the former (Xu et al., 2015).

The Semantic Textual Similarity (STS) (Agirre et al., 2016b) task also shows some level of difficulty which can be attributed to the diversity of text types (see Table 5.2), making generalisation a challenge for the systems. However, recent BERT models (Yang et al., 2019; Devlin et al., 2019a) have pushed the performance higher than previous methods. In fact, recent models (Devlin et al., 2019a; Wang et al., 2019) have outperformed human-level performance on the MSR Paraphrase Corpus (Dolan et al., 2004), the first publicly available corpus for paraphrase identification. Compared to the STS dataset, the MSR Par corpus concerns only one text type, and thus systems were not faced with the challenge of generalising their algorithms across domains. However, this seems to be not an issue with the BERT models anymore as reflected in its state-of-the-art performance in both STS dataset (Yang et al., 2019) and the MSR Par dataset (Wang et al., 2019).

Finally, high performance can be observed on the SICK dataset (Marelli et al., 2014) which is dominated by DL-based models. Partly, this could be attributed to it being a subset of the MSR Paraphrase Corpus though it was post-processed specifically to provide a benchmark for testing Compositional Distributional Semantic Models (CDSMs). That is, some factoids in this dataset were slightly modified (e.g., “Theresa May” replaced with “The prime minister”) in order to capture elements pertinent to compositional distributional semantics such as contextual synonymy and other lexical variation, active/passive and other syntactic alternations, as well as the impact of negation, determiners and other grammatical elements (Marelli et al., 2014).
In summary, the noisy nature of data (e.g., PIT) and the requirement for alignment of chunks (e.g., iSTS) are still open challenges in these tasks.

5.3 Classes of Methods: A Closer Look

We have seen that deep learning methods have become the basis of dominant approaches to semantic similarity measurement for reasons mentioned in the previous section.

As a contribution of this work, in which we focus on deep learning methods, we survey the spectrum of these methods and argue that they lie in two dimensions (see Figure 5.3): input representation granularity and compositional model complexity, which are both central to sentence modelling (see Sections 5.3.1 and 5.3.2).

Figure 5.3 shows a snapshot of efforts based on a wide range of deep learning methods for sentence modelling, where top ranking DL-based systems are highlighted (Wang et al., 2019; Mueller and Thyagarajan, 2016a; Vosoughi et al., 2016; Yang et al., 2019; Peng et al., 2019a). The rest are the next best ranking systems (Zhou et al., 2016; Yin et al., 2015; He and Lin, 2016; He et al., 2015; Cheng and Kartsaklis, 2015; Hashimoto et al., 2016; Wieting et al., 2016; Kiros et al., 2015; Tai et al., 2015b; Socher et al., 2011a; Blacoe and Lapata, 2012a) along with generic sentence modelling algorithms (Zhang et al., 2015; Sutskever et al., 2014; Kalchbrenner et al., 2014; Hashimoto et al., 2014; Mikolov et al., 2013a; Levy and Goldberg, 2014; Bahdanau et al., 2014; Mikolov et al., 2013b; Vinyals et al., 2015; Kalchbrenner and Blunsom, 2013).

On the one hand, we observe that most systems, in particular the top-performing ones, used word-based representations while coarser representations or higher level semantic structures are still left unexplored such as the use of predicate-argument or event structures.
Figure 5.3: Deep learning methods for sentence modelling (top-performing systems highlighted).

On the other hand, it is worth noting that the top-performing systems use compositionality-driven architectures, confirming our assumptions regarding compositionality as discussed in Section 5.1. We can see in Figure 5.3 that most of the highlighted top-performing systems are on the upper quadrants of the diagram which are closer to the compositionality-driven end of the model complexity axis.

In the next sections, we discuss in detail each of the dimensions.
5.3. CLASSES OF METHODS: A CLOSER LOOK

5.3.1 Representation Granularity

The horizontal axis (see Figure 5.3) shows the granularity of input representation. The most popular input representation is word-based (see Figure 5.3) which was pioneered by Mikolov et al. (2013a) and was later extended to account for phrases (Mikolov et al., 2013b). Zhang et al. (2015); dos Santos and Zadrozny (2014), however, argue that character-level representations are more robust to languages characterised by the proliferation of morphological variations, e.g., tweets (Vosoughi et al., 2016), and are more language-independent since characters always constitute a necessary construct in every language.

Recently, word or sentence representations using BERT and related models (Devlin et al., 2019a; Wang et al., 2019; Yang et al., 2019) have proved to be successful in many natural language understanding tasks (Wang et al., 2018a). However, while the representations are fine-tuned for word or sentence representations for the target task, the underlying BERT models are pre-trained using sub-word input representations (Devlin et al., 2019a), hence, they are categorised in Figure 5.3 as between the word and character representation granularity.

The sub-words used in BERT models are created (Devlin et al., 2019a) in the following manner. The tokeniser works by going through a big corpus of words and greedily incorporates words into its vocabulary based on frequency, up to a maximum vocabulary size. If words show up quite frequently, then they merit their own place in the vocabulary, otherwise, the words are split into its constituent sub-words that appeared frequently enough. Hence, the input representation of a sentence can be composed of characters, words or subwords, which are collectively called as sub-words. For example, the word “stealing” can be split into “steal” and “ing” but it may not be tokenised and remains as “stealing” because it occurs quite frequently. However, the word “mississippi” could be tokenised into subwords which occur frequently enough,
for example into the following: “miss”, “iss”, “ip”, “pi”. This is different from the more traditional tokeniser that splits stem words from suffixes or pulls apart lemmas and morphological suffixes and this produces good results in practice (Devlin et al., 2019a).

On the other side of the spectrum, Levy and Goldberg (2014) proposed to generalise the skip-gram model to use arbitrary contexts to capture different notions of similarities (e.g., functional similarities using syntactic contexts vs topical similarities using linear bag-of-words contexts in (Mikolov et al., 2013a)). Furthermore, Hashimoto et al. (2014) extended this skip-gram model to predicate-argument structures to capture long-range dependencies while Kiros et al. (2015) abstracted this to the sentence level, termed as skip-thought vectors.

In summary, this axis characterises a trade-off between data sparsity and syntactic dependencies captured in the representation. On the one hand, character-based methods are not faced with the data sparsity problem; however, it is not straightforward to determine whether composing sentences based on individual character representations would represent the originally intended semantics. On the other hand, while sentence-based representations could intuitively represent the actual semantics, this approach suffers from data sparsity and does not directly address the “interpretable” variant of the semantic similarity measurement task which requires the semantic representation of sentence constituents.

### 5.3.2 NN Architectures’ Complexity

The vertical axis (see Figure 5.3) shows the spectrum of neural network architectures’ complexity ranging from bag-of-items-driven\(^3\) architectures to compositionality-driven ones to account for the morphological, lexical, syntactic, and compositional aspects of

\(^3\)We use *items* instead of *words* to generalise amongst various representations.
a sentence.

Some of the popular methods are based on Vector Space Models (VSMs), which represent a sentence by performing algebraic operations (e.g., addition or averaging) over the vector representations of individual constituents (Turney et al., 2010; Blacoe and Lapata, 2012b; Mikolov et al., 2013a). However, these models have received criticism as they use linear bag-of-words context and thus do not take into account syntax.

Spatial neural networks (e.g.,Convolutional Neural Networks or ConvNets), were pioneered by LeCun et al. (1998) for image classification, have been shown to be capable of modeling sentences by mapping them to vectors (Kalchbrenner and Blunsom, 2013). Additionally, they can capture morphological variations (dos Santos and Zadrozny, 2014; Chiu and Nichols, 2016). However, this architecture still does not capture syntactic information, thus Sutskever et al. (2014) proposed the use of sequence-based neural networks (e.g.,

Recurrent Neural Networks, Long Short Term Memory models) to capture long-range temporal dependencies. Furthermore, Dyer et al. (2015b) showed that bidirectional sequence modeling captures more syntactic information. Socher et al. (2011a) and Tai et al. (2015a) introduced a generalization of the sequential LSTM based on tree-structured network topologies (e.g., Recursive Neural Networks) as an effective model to capture syntactic dependencies in a sentence. However, this type of network requires input from an external resource (i.e., dependency/constituency parser).

Top-performing models in the upper left quadrant of Figure 5.3 involved stacked architectures of the three basic forms above (He and Lin, 2016; Yin et al., 2015; Kalchbrenner et al., 2014; Cheng and Kartsaklis, 2015; Zhang et al., 2015) which capture the syntactic and semantic structure of a language.

However, since most of these architectures model sentences as vectors with a fixed size, they risk losing information especially when input sentence vectors are of varying lengths. Bahdanau et al. (2014) addressed this problem, originally in the context of
machine translation, by introducing the concept of attention where the network learns to align parts of the source sentence that match the constituents of the target sentence, without having to explicitly form these parts as hard segments. This enables phrase-alignments between sentences as described by Yin and Schütze (2016) in the context of a textual entailment recognition task.

Recently, deep bidirectional multi-head self-attention networks are used in the BERT model (Devlin et al., 2019a) and its variants (Yang et al., 2019; Wang et al., 2019) in the form called Transformer network (Vaswani et al., 2017). The Transformer network relies solely on attention mechanism and dispenses with recurrence and convolutions entirely.

5.3.3 NN Architectures for Semantic Similarity

Figure 5.4 shows a typical DL architecture for semantic similarity measurement, a central component of which is sentence modelling. The sentence representation, as outputs of the sentence models, are usually fixed-length vectors, which are compared using similarity metrics and input to a classifier to determine the final output. Various similarity metrics have been used, obtaining various performance results (Cheng and Kartsaklis, 2015; He and Lin, 2016; Mueller and Thyagarajan, 2016a; Vosoughi et al., 2016).

As shown in this general architecture, some DL methods for sentence modelling, marked by broken lines, use hand-crafted features (Cheng and Kartsaklis, 2015; Yin et al., 2015), data augmentation, e.g., synonym augmentation (Mueller and Thyagarajan, 2016a; Vosoughi et al., 2016), external resources, e.g., parsers, chunkers, POS taggers (Tai et al., 2015a), model pretraining (Mueller and Thyagarajan, 2016a) and pretrained vectors (which form the majority). Cheng and Kartsaklis (2015) proposed an architecture that imposes dynamic word sense induction for each word (i.e., using
5.4 Conclusion

In this chapter, we have presented the state-of-the-art in sentence representation. In particular, we have described a two-dimensional framework which highlighted some...
areas that are under-explored such as event structures.

In the next chapter, we present some preliminary experiments to uncover the limitations of a state-of-the-art approach in one of the datasets.
Chapter 6

Sentence Representation Preliminary Experiments

Using the framework presented in Chapter 5, we describe in this chapter the preliminary experiments we conducted in order to gain a deeper understanding on the limitations of the state-of-the-art model in the SICK 2014 dataset. Our preliminary quantitative and qualitative experiments reveal the limitations of the current state-of-the-art model (Mueller and Thyagarajan, 2016a) in the context of sentence similarity tasks.

6.1 Experiments Settings

Firstly, we define sentence similarity as a supervised learning task where each training example consists of a pair of sentences \((x_1^a, \ldots, x_T^a), (x_1^b, \ldots, x_T^b)\) of fixed-sized vectors (where \(x_i^a, x_j^b \in \mathbb{R}^{d_{inp}}\) denoting constituent vectors from each sentence, respectively, which may be of different lengths \(T_a \neq T_b\)) along with a single real-valued label \(y\) for the pair.

We evaluated the performance of the state-of-the-art model on this task.
CHAPTER 6. PRELIMINARY EXPERIMENTS

6.1.1 Model Overview

Since we focus on end-to-end sentence modelling, we implement a simplified (see Table 6.1) version of MaLSTM (Mueller and Thyagarajan, 2016b), i.e., the state-of-the-art model on this task (see Figure 5.2).

The model uses a siamese architecture of Long-Short Term Memory (LSTM) to read word vectors representing each input sentence. Each LSTM cell has four components: input gate $i_t$, forget gate $f_t$, memory state $c_t$, and output gate $o_t$; which decides the information to retain or forget in a sequence of inputs.

Equations 1-6 are the updates performed at each LSTM cell for a sequence of input $(x_1, ..., x_T)$ at each timestep $t \in \{1, ..., T\}$, parameterised by weight matrices $W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o$ and bias vectors $b_i, b_f, b_c, b_o$.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (6.1)$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (6.2)$$
$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (6.3)$$
$$c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \quad (6.4)$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (6.5)$$
$$h_t = o_t \odot \tanh(c_t) \quad (6.6)$$

This model computes the sentence similarity based on the Manhattan distance between the final hidden state representations for each sentence: $g(h_a^T, h_b^T) = \exp(-||h_a^T - h_b^T||_1) \in [0, 1]$, which was found to perform better empirically than other simple similarity functions such as cosine similarity (Mueller and Thyagarajan, 2016b). The Manhattan distance (Minkowski, 1910) between two state representations is calculated as the sum of the absolute differences between each representation.
6.1. EXPERIMENTS SETTINGS

<table>
<thead>
<tr>
<th>Salient Feature</th>
<th>MaLSTM</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>pre-training</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>synonym augmentation</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>prediction calibration</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>optimisation method</td>
<td>Adadelta</td>
<td>Adam</td>
</tr>
</tbody>
</table>

Table 6.1: Model comparison between MaLSTM and our implementation

6.1.2 Training Details

We use the 300-dimensional pre-trained word2vec\(^1\) (Mikolov et al., 2013b) word embeddings and compare the performance with that of GloVe\(^2\) (Pennington et al., 2014) embeddings. Out-of-embedding-vocabulary (OOEV) words are replaced with an \(<\text{unk}>\) token. We retain the word cases and keep the digits. For character representation, we fine-tune the 50-dimensional embeddings.

Our model uses 50-dimensional hidden representations \(h_t\) and memory cells \(c_t\). Optimisation of the parameters is done using the SGD-based Adam method (Kingma and Ba, 2014) and we perform gradient clipping to prevent exploding gradients. After conducting initial experiments, we found the optimal training parameters to be the following: batch size = 30, learning rate = 0.01, learning rate decay = 0.98, dropout = 0.5, number of LSTM layers = 1, maximum epochs = 10, patience = 5 epochs. We used the Tensorflow\(^3\) library to implement the model.

6.1.3 Dataset and Evaluation

We measure the model’s performance on three benchmark datasets, i.e., SICK 2014 (Marelli et al., 2014), STS 2016 (Agirre et al., 2016a) and PIT 2015 (Xu et al., 2015), using Pearson correlation. We assert that a robust model should perform consistently well in these three datasets.

\(^1\)code.google.com/p/word2vec, accessed 25 June 2019
\(^3\)https://www.tensorflow.org/, accessed 25 June 2019
CHAPTER 6. PRELIMINARY EXPERIMENTS

### Table 6.2: Pearson Correlation. Performance comparison across input representations and composition models. Baseline method uses cosine similarity measure to predict similarity between sentences.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Baseline</th>
<th>LSTM</th>
<th>Vector Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>GloVe word2vec char</td>
<td>GloVe word2vec char</td>
</tr>
<tr>
<td>SICK 2014</td>
<td>0.5713</td>
<td>0.7514 0.7149 0.3487</td>
<td>0.4877 0.4998 0.0178</td>
</tr>
<tr>
<td>PIT 2015</td>
<td>0.4001</td>
<td>0.1562 0.1334 0.0086</td>
<td>0.1273 0.0859 0.0000</td>
</tr>
<tr>
<td>STS 2016</td>
<td>0.5055</td>
<td>0.4174 0.2378 0.1067</td>
<td>0.4971 0.4741 -0.0100</td>
</tr>
</tbody>
</table>

Table 6.3: Percentage of Out-of-Embedding-Vocabulary (OOEV) words

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Vocab Size</th>
<th>% OOEV word2vec</th>
<th>% OOEV GloVe</th>
</tr>
</thead>
<tbody>
<tr>
<td>SICK 2014</td>
<td>2,556</td>
<td>7.0</td>
<td>5.9</td>
</tr>
<tr>
<td>PIT 2015</td>
<td>15,156</td>
<td>16.5</td>
<td>9.6</td>
</tr>
<tr>
<td>STS 2016</td>
<td>25,264</td>
<td>42.6</td>
<td>32.1</td>
</tr>
</tbody>
</table>

Furthermore, using the framework described in Section 5.3, we chose to compare the model performance at two levels of input representation (i.e., character-level vs word-level) and composition models (i.e., LSTM vs vector sum) in order to eliminate the need for external tools such as parsers.

### 6.2 Results and Discussion

Table 2 shows the performance across input representations and composition models. Our simplified model performs relatively well (Pearson correlation = 0.7149) when compared to what was reported in the original MaLSTM paper (Pearson correlation = 0.8822) on the SICK dataset (using word2vec). Considering that our model did not take advantage of the additional salient features described in Table 6.1, this is a reasonable performance drop in exchange for an end-to-end model.

Noticeably, the baseline method outperforms the other methods especially on

---

4We represent each sentence with term-frequency vectors.
6.2. RESULTS AND DISCUSSION
datasets with higher percentage of OOEV words (see Table 3). Although we see that GloVe embeddings have better vocabulary coverage than word2vec in all datasets, even then, we observe that around a third of the vocabulary in STS is not covered by GloVe.

With respect to input representation, the word-based one yields better performance in all datasets over character-level representation for the obvious reason that it carries more semantic information. Furthermore, the character-level representation using LSTM performs better than using Vector Sum (VS) because it is able to retain sequential information. Regarding word embeddings, GloVe resulted in higher performance compared to word2vec when using the same composition model (e.g., 0.4174 vs 0.2378) with LSTM in STS. This can be attributed to its higher vocabulary coverage except when using VS in SICK.

With respect to the composition model, LSTM performs better over VS in all datasets except in the STS dataset where the pattern reverses. We can clearly see in Figure 6.1 that the VS composition model performs better in all categories except in Post-editing where LSTM using GloVe yields the best correlation. Particularly, LSTM using word2vec performs poorly in some datasets namely, Question-Question, Plagiarism, and Answer-Answer.

The worst performance was obtained on the PIT dataset. Although it did not have as many OOEV words as STS, its diverse and noisy user-generated text (Strauss et al., 2016a)—typical of social media text—make it a very challenging dataset.

In the case of increasing training set size, we investigated different sizes according to percentage from actual dataset: \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}. The results show that Pearson correlation is increasing for word-SICK combination while the other combinations did not show any noticeable increase over increasing dataset size. This means that increasing training set size for the word-SICK combination will possibly improve the performance of the model.

As regards the number of hidden units, we experimented with the following values
Figure 6.1: Pearson correlation in STS 2016 evaluation sets (Key: L=LSTM, V=Vector Sum, C=char, W=word2vec, G=GloVe)
6.3. ERROR ANALYSIS

for the word-SICK combination: \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 250, 300\}. We observed that the first four values result in high Pearson correlations with 40 hidden units giving the highest Pearson correlation of 0.7 and after which as the number of hidden units increase, the correlation decreases consistently. This means that the model can only capture limited number of features in its hidden state representations.

We also observed the same behaviour if we have more number of LSTM layers stacked on top of one another. We experimented with the following values \{1, 3, 5, 7, 9, 11\} and we found that 1 layer gives the highest correlation and it decreases when there are more layers.

6.3 Error Analysis

To better understand the reason behind the performance drop of the model, we extracted the 100 most difficult sentence pairs in each dataset by ranking all of the pairs in the test set according to the absolute difference between the gold standard and predicted similarity scores.

We observed that 60% of them share many similar words (except for a word or two) or contain Out-of-Embedding-Vocabulary (OOEV) words (see Examples 1 and 2, respectively) that led to a complete change in meaning. Meanwhile the remaining 40% are sentence pairs which are topically similar but completely mean different. This was observed in both compositional models.

In the examples below, we show the scores (i.e., Pearson correlation) from the gold standard and some of the methods presented in Figure 5.1. The two sentences come from an actual pair in the dataset. Example 2 shows two additional sentences with OOEV words (indicated as \texttt{<unk>}), which served as actual input to the model.

Example 1: SICK
A person is scrubbing a zucchini
The woman is cutting cooked octopus

Example 2: STS Question-Question

What kind of socket is this?
What kind of bug is this?
What kind <unk> socket is this <unk>
What kind <unk> bug is this <unk>

On the one hand, Example 1 shows a pair of sentences which although they can be interpreted to belong in the same domain *food preparation*, they are semantically different in their *verb, subject, and direct object*, which presumably for that reason they were labelled in the gold standard as highly dissimilar. In contrast, both of the word-based models (i.e., V+G, L+G) predicted them to be highly similar (in varying degrees). This limitation can be attributed to the relatedness of its words (e.g., *person* vs *woman*, *cutting* vs *scrubbing*). Under the *distributional hypothesis assumption* (Harris, 1940; Firth, 1957), two words will have high similarity if they occur in similar contexts even if they neither have the same nor similar meanings. Furthermore, the higher similarity obtained by the LSTM model (i.e., L+G) over Vector Sum (i.e., V+G) can be attributed to its ability to capture syntactic structure in sequences such as in a sentence.

Example 2 shows a sentence pair which was labelled as completely dissimilar but was predicted with high similarity in both models. This shows the inability of the models to put more weight on semantically rich words which change the overall meaning of a sentence when compared with another.

Additional examples are shown below.
6.4 Discussion

This work is intended to serve as an initial study for end-to-end sentence modelling and to identify the limitations and challenges associated with it. The models and representations compared, while typical of current sentence modelling methods, are not an exhaustive set and some variations exist. A natural extension to this study is to explore other input granularity representations and composition models presented in the framework. For example, in this study we only used word representations; however, multi-word expressions are common occurrences in the English language. This could be addressed by modelling sentence constituents using recursive tree structures (Tai et al., 2015b) or by learning phrase representations (Wieting et al., 2015).
The limitations of the current word embeddings as revealed in this work have been studied in the context of word similarity tasks (Levy and Goldberg, 2014; Hill et al., 2016). Furthermore, Kiela et al. (2015) have shown that specialising semantic spaces to downstream tasks and applications requiring similarity or relatedness can improve performance. But to the best of our knowledge, our work is the first one that explicitly investigated the limitations of word embeddings in the context of sentence similarity tasks.

Some studies (Faruqui et al., 2014; Yu and Dredze, 2014; Ono et al., 2015; Ettinger et al., 2016) have proposed to learn word embeddings by going beyond the distributional hypothesis assumption either through a retrofitting or joint-learning process with some using semantic resources such as ontologies and entity relation databases. This direction can be explored since entities encode much of the semantic information in a language.

The inability of the state-of-the-art model to encode semantically rich words (e.g., socket, bug in Example 2) with higher weight when compared with another substantiates the idea of Blacoe and Lapata (2012a) that distributive semantics representation and composition must be mutually learned. Recently, Hashimoto et al. (2016) proposed a joint model for sentence representation at different linguistic levels (i.e. morphology, syntax, semantics) in a hierarchy of tasks. With that model, different tasks (i.e., POS tagging, dependency parsing, semantic role labelling) which are often done separately, can be trained in an end-to-end manner with the assumption that higher-level tasks benefit from lower-level tasks.

In the following paragraphs, we discuss the main findings in this chapter in connection with our focus on biomedical event detection.

Firstly, based on the framework that we have developed in the previous chapter, we have seen in Figure 5.3 that under the input granularity axis, using higher level semantic structures such as predicate-argument structures or event structures for sentence
modelling are under-explored. In this thesis, we choose to focus on event structures: their detection and representation.

The second finding in this chapter is that the general domain data contains more noisy data that can affect the performance on textual similarity tasks. In particular, we observed in our experiments in this chapter that the PIT dataset obtained the worst performance. We attributed the low performance to its diverse and noisy user-generated text (Strauss et al., 2016a) typical of social media text. In contrast, the biomedical literature is less noisy and uses formal and academic language creating less ambiguity.\(^5\) This also support why we focus our study on biomedical domain.

In the next section, we present an in-depth investigation about the second finding. Concretely, we want to investigate the effect of noisy dataset on a particular task, named-entity recognition to constrain the analysis.

## 6.5 Further Investigation on Noisy Data

*The content in this section is a shortened version of the author’s publication (Espinosa et al., 2016) in Computational Linguistics (COLING) 2016 Workshop on Noisy User-generated Text."

In the previous section, we have seen that our model obtained worst performance on the PIT dataset. One of the main reasons is the noisy nature of the dataset typical in social media text. In this section, we present an investigation of the effect of noisy dataset on named entity recognition task and we explore methods that can address this.

Named entity recognition (NER) in social media (e.g., Twitter) is a challenging task due to the noisy nature of text. As part of our participation in the W-NUT 2016 Named Entity Recognition Shared Task\(^6\), we proposed an unsupervised learning approach

\(^5\)http://people.ischool.berkeley.edu/~hears/text-mining.html, accessed 25 June 2019

using deep neural networks and leverage a knowledge base (i.e., DBpedia) to bootstrap sparse entity types with weakly labelled data.

To further boost the performance, we employed a more sophisticated tagging scheme and applied dropout as a regularisation technique in order to reduce overfitting. Even without hand-crafting linguistic features nor leveraging any of the W-NUT-provided gazetteers, we obtained robust performance with our approach, which ranked third amongst all shared task participants according to the official evaluation on a gold standard named entity-annotated corpus of 3,856 tweets.

6.5.1 Methodology

We cast the NER task as a sequence labelling problem: every tweet is a sequence of tokens, each of which is automatically assigned a label (or tag) that is indicative of its membership to a semantic type or category. In learning and applying token labels, two different tagging schemes were compared in order to confirm previously reported observations that employing more sophisticated tagging schemes leads to better predictions (Ratinov and Roth, 2009; Dai et al., 2015).

We implement conditional random fields (CRFs) as the baseline approach because they are known to demonstrate strong performance (McCallum and Li, 2003; Leaman et al., 2008; Finkel and Manning, 2009). Meanwhile, convolutional neural networks (CNNs) are sparse feed-forward neural networks which have been shown to effectively extract morphological features such as word prefixes and suffixes (Chiu and Nichols, 2016). One can thus explore the combination of the ability of CRFs to model relations between labels of tokens in a tweet, and the effectiveness of CNNs to extract morphological features of words. CNNs, however, suffer from one drawback, i.e., failure to capture the contextual information surrounding a word. To alleviate this issue, we instead employed recurrent neural networks (RNNs) (Rumelhart et al., 1988). RNN
models make use of the sequential information found in structured input (e.g., in a sentence). They are called recurrent because they perform the same computation for every element in a sequence with inputs as the only difference.

The input to the network is a concatenation of character representations of words extracted from a convolutional neural network and word embeddings (see Figure 6.2). The inputs are fed into a bi-directional LSTM (BLSTM) and the outputs are concatenated and fed into the CRF layer to jointly decode the best label sequence.

The schematic diagram depicting this neural network architecture is shown in Figure 6.3 following the work of Ma and Hovy (2016) who first proposed this method for NER in newswire. In our work, the same architecture is applied to the task of NER in tweets.

To mitigate the data sparsity of some entity types as observed in the data set, we sought to increase their number of annotated samples.

To this end, we leveraged DBpedia Spotlight (Daiber et al., 2013), a tool for linking mentions in text to DBpedia entries, which in our case was employed to weakly annotate our own collection of tweets, gathered between July and August 2016. The
tool can be configured in order to define the resulting entity matches. For example, there are configuration parameters, e.g., confidence and contextual score, which can be specified to control the quality of the output (Mendes et al., 2011).

Confidence is a parameter for disambiguation, ranging from 0 to 1, which takes into account factors such as topical pertinence and contextual ambiguity. The confidence threshold setting instructs the tool to exclude low-ranking annotations at the risk of losing potentially correct ones. For example, a confidence level of 0.7 will eliminate 70% of low-ranking candidates.

Meanwhile, contextual score is the cosine similarity between term vectors weighted by the Term Frequency Inverse Candidate Frequency (TF*ICF) measure introduced by
Mendes et al. (2011). ICF is based on the idea that the discriminative power of a word is inversely proportional to the number of DBpedia resources it is associated with. The above parameters provide the tool with a way to rank the list of candidates.

In applying DBpedia Spotlight on our own collection of tweets, we initially set the confidence and contextual score parameters to 0.7. Fuzzy string matching heuristics were also employed, to enable the matching of lexical variants in tweets (e.g., those which differ in length by at most two characters) against DBpedia entries. The tool assigns matched entities with semantic labels that come from DBpedia’s ontology, which defines a hierarchy of types.

In order to categorise these entities according to the shared task’s categories of interest, we defined the mappings shown in Table 6.4, informed by the annotation guidelines provided by the shared task organisers. For many entities, DBpedia Spotlight provides multiple hierarchical labels. For example, the entity “Justin Bieber” is

<table>
<thead>
<tr>
<th>Entity Type</th>
<th>DBpedia Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>company</td>
<td>Website, Company, Software, BroadcastNetwork, RadioStation</td>
</tr>
<tr>
<td>geo-loc</td>
<td>PopulatedPlace, WorldHeritageSite</td>
</tr>
<tr>
<td>other</td>
<td>Non-ProfitOrganisation, Event, SportsLeague, PoliticalParty, MilitaryUnit</td>
</tr>
<tr>
<td>movie</td>
<td>Film</td>
</tr>
<tr>
<td>musicartist</td>
<td>MusicArtist</td>
</tr>
<tr>
<td>person</td>
<td>Person, Athlete, Celebrity, Cleric, Coach, BeautyQueen, BusinessPerson, Criminal, Economist, Engineer, FictionalCharacter, Journalist, Judge, Lawyer, Model, Monarch, MovieDirector, Noble, OrganisationMember, Politician, PoliticianSpouse, Royalty, Scientist, SportsManager, TelevisionDirector, TelevisionPersonality, Writer, OfficeHolder</td>
</tr>
<tr>
<td>product</td>
<td>VideoGame, Device, MeanOfTransportation, Food, LineOfFashion, ProgrammingLanguage</td>
</tr>
<tr>
<td>tvshow</td>
<td>TelevisionShow, TelevisionSeason, TelevisionEpisode</td>
</tr>
<tr>
<td>sportsteam</td>
<td>SportsTeam</td>
</tr>
</tbody>
</table>

Table 6.4: Proposed mapping between W-NUT entity types and DBpedia categories
labelled by the tool as both Person and MusicArtist, which map to W-NUT’s person and musicartist, respectively. In such cases, we take the most specific entity type (e.g., musicartist) as the final label.

Furthermore, if an entity match $x$ is subsumed by another match $y$, we disregard $x$ and retain the longer matching entity $y$ with its corresponding type. For example, if a tweet contains three entity matches such as “New York USA”, “New York” and “USA”, we disregard the two shorter entity matches (“New York”, “USA”) and choose the entity with the longer span of tokens (“New York USA”).

The above techniques, however, did not result in better performance. With the threshold of 0.7, which turned out to be too low, even named entities that DBpedia Spotlight was not very confident about were included in the training set, thus introducing noise. This corresponds to token sequences which were recognised as entities of a certain type when in reality they are actually non-entities or fall under a different entity type. This noise is compounded as the bi-directional LSTM models we employed made use of the context surrounding the wrongly recognised entities to learn the segmentation and classification tasks.

To resolve these issues, we increased the required level of confidence in weakly annotated named entity types, from 0.7 to 0.9. Furthermore, we added weakly annotated data only to the entity types for which our models have obtained poor performance based on our initial experiment, namely: product, movie, musicartist, tvshow. In this way, we avoid adding unexpected noise with respect to the other named entity types.

### 6.5.2 Results and Discussion

We then evaluated the contribution of augmenting the sparse entity types in the training data with weakly labelled named entities. As presented in Table 6.5, performance noticeably increased for all entity types except in the case of tvshow, in which no
6.5. **FURTHER INVESTIGATION ON NOISY DATA**

<table>
<thead>
<tr>
<th>Entity Type</th>
<th>Training Set only</th>
<th>Training Set + Weakly Labelled entities for sparse types</th>
</tr>
</thead>
<tbody>
<tr>
<td>tvshow</td>
<td>44.44</td>
<td>44.44</td>
</tr>
<tr>
<td>product</td>
<td>40</td>
<td>41.38</td>
</tr>
<tr>
<td>musicartist</td>
<td>28.57</td>
<td>46.15</td>
</tr>
<tr>
<td>movie</td>
<td>0</td>
<td>16.67</td>
</tr>
</tbody>
</table>

Table 6.5: Effect on the F-scores after adding weakly labelled data to sparse entity types

<table>
<thead>
<tr>
<th>System Name</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>CambridgeLTL</td>
<td>60.77</td>
<td>46.07</td>
<td>52.41</td>
</tr>
<tr>
<td>Talos</td>
<td>58.51</td>
<td>38.12</td>
<td>46.16</td>
</tr>
<tr>
<td>akora</td>
<td>51.70</td>
<td>39.48</td>
<td>44.77</td>
</tr>
<tr>
<td>NTNU</td>
<td>53.19</td>
<td>32.13</td>
<td>40.06</td>
</tr>
<tr>
<td>ASU</td>
<td>40.58</td>
<td>37.58</td>
<td>39.02</td>
</tr>
</tbody>
</table>

Table 6.6: Performance of our NER system, akora, compared to other top-ranking W-NUT 2016 systems

The effect is especially high for *movie* and *musicartist*—entity types for which the F-scores were previously zero. This demonstrates that the incorporation of weakly labelled data in training the neural network can lead to significant improvement with respect to sparse entity types.

Underpinned by the final NER model, our system (“akora”) was applied on the shared task’s test data, yielding an overall precision = 51.70, recall = 39.48, and F-score = 44.77. It ranked third amongst participating systems (Strauss et al., 2016b) as shown in Table 6.6. Details of the evaluation results according to entity type are presented in Table 6.7.

### 6.5.3 Conclusion

In this section, we presented an investigation of the effect of noisy dataset in the context of named entity recognition task. Our experiments have shown that taking advantage of weakly annotated data to alleviate data sparsity of some named entity types resulted in
increased performance in the respective entity types. Future work includes performing further error analysis by means of hidden layer visualisation to enable us to investigate representations created at each layer. Moreover, Bayesian optimisation of parameters can be done as proposed by Rasmussen (2006). Finally, considering the language drift inherent in social media content, looking into word sense disambiguation may help improve performance.

In the next chapter, we discuss how we apply event-based representations we developed in Chapter 3 and Chapter 4 to a biomedical semantic similarity task.
Chapter 7

Event Representation Applied to Biomedical Semantic Similarity Task

7.1 Overview

In this chapter, we describe how we use event representations developed in Chapter 3 as sentence representation in semantic textual similarity tasks. An event representation, which is a $d$-dimensional vector, carries the event structure information. As we have discussed in Chapter 3, an event structure consists of trigger-argument relations. We investigate in this chapter to what extent we can use the event structure representation as sentence representation in the context of semantic similarity estimation task.

In the following section we provide a brief review about sentence representation in the biomedical domain which was discussed in more detail in Chapter 3. Then we discuss our methodology and the experimental settings in the succeeding sections. Finally, we present the experimental results and analysis.
7.2 Introduction

The increasing amount of scientific literature published in the biomedical domain leaves researchers with considerable challenges of keeping up-to-date with the latest advances and developments. Advanced text mining techniques are needed to retrieve pertinent information from the literature in a precise and accurate manner (Ananiadou et al., 2014). One of the essential components that enable efficient information retrieval is the ability to determine semantic similarity between textual components (e.g., words, sentences, paragraphs) (Soğancıoğlu et al., 2017). Thus, there is a need to improve the semantic similarity estimation between textual components. In this chapter, we focus on the sentence-level semantic similarity estimation.

Specifically, we seek to investigate to what extent we can use the event representations developed in Chapter 3 for textual similarity task. We believe that accurate event representations will have further downstream applications such as event-based search systems on the massive biomedical literature (Ananiadou et al., 2010; Miyao et al., 2006; Van Landeghem et al., 2013a).

In contrast to the general domain, as we saw in Chapter 3, there are only few approaches to date that focus on semantic similarity estimation on the biomedical domain. Semantic similarity methods developed for the general domain produce poor results for biomedical text because they do not effectively cover biomedical knowledge (Soğancıoğlu et al., 2017).

Sentence-level semantic similarity computation approaches in the biomedical domain include string similarity measures (Ukkonen, 1992; Lawlor, 1980; Prasad and Sharma, 2018), distributed vector representations of sentences learned in an unsupervised manner from large biomedical corpus (Blagec et al., 2019; Chiu et al., 2016b; Moen and Ananiadou, 2013b) and ontology-based approaches that utilise specific ontologies (Sáñchez and Batet, 2011; Ferreira and Couto, 2019; Aouicha and Taieb,
2016; Harispe et al., 2014; Mabotuwana et al., 2013). To the best of our knowledge, our approach on the use of event representations as sentence representations for the semantic similarity tasks in the biomedical domain is novel.

In the next section, we describe our methodology for applying event representations to semantic similarity estimation in the biomedical domain.

7.3 Methodology

In this section, we first formulate the task and then describe our method.

Given a pair of sentences \((s_1, s_2)\), we extract a set of event representations from each sentence \((e_1, \ldots, e_m, e_1, \ldots, e_n)\) where \(e_i\) are \(d\)-dimensional vectors and \(m\) and \(n\) are the number of events for \(s_1\) and \(s_2\), respectively. We then generate a single sentence representation \(r\) from all the events in a sentence and use this representation as input to a similarity function \(f(r_1, r_2)\) to determine the similarity score between the pair of sentences.

The first step involves the extraction of the event structures from each sentence. Next, we generate a sentence representation from all the event representations in a sentence. Lastly, we evaluate the sentence representation on the semantic similarity task.

7.3.1 Event Detection

We use the event detection model developed in Chapter 3 to detect event structures from the BIOSSES dataset (Soğancioğlu et al., 2017) (to be discussed in more detail in §7.4). However, our model requires relations as input to detect the event structures and generate their representations. To achieve this, we use the state-of-the-art event extraction TEES system (Björne and Salakoski, 2018a) to detect relations. To detect
the event structures and generate event representations, we use our event detection model.

Our model has a threshold parameter, which determines the confidence level of the model on event detection. A threshold of zero (0) constructs all possible events from the given relations while higher thresholds predict events with higher precision and accuracy. Typical threshold values range from \([0,0.5]\). In Chapter 4, we have shown that the best performance of our model was at threshold equal to 0.5 but this can be set to a lower value at test time to allow more events to be predicted but lower precision and accuracy.

We use our event detection model that is trained on a large biomedical event dataset with forty (40) event types (Pyysalo et al., 2015) and though the set of event types is not exhaustive, it covers most of the event structures. The number of event types can also be expanded by combining several corpora. For instance, Miwa et al. (2013b) showed that combining seven (7) corpora resulted in sixty-five (65) event types. However, we think that this is not significantly larger than the current set of types considering the number of corpora combined and thus we leave this as future work.

Event structures are detected on the sentence level and the model can detect multiple events in each sentence. An event representation, which is a \(d\)-dimensional vector representation is generated for each predicted event structure.

### 7.3.2 Sentence Representation

We generate sentence representations from the event representations. Since there could be multiple/nested events \(n\) detected in a sentence, we employ an aggregation function to generate a single sentence representation. Currently, there are no other approaches generating sentence representations from event representations, hence, in our study we use a simple sum function shown in Equation 7.1, to generate a single representation
7.3. METHODOLOGY

$r$ for each sentence. The sum function has been shown to be an effective aggregation function in generating sentence representations from word representations (Wieting et al., 2015; Blacoe and Lapata, 2012a).

\[ r = \sum_{i=1}^{n} e_i \]  (7.1)

More sophisticated aggregation function such as an attention mechanism (Bahdanau et al., 2014) or other neural network architectures (Cheng et al., 2018; Zhang et al., 2018) is not possible at this point since there are no datasets with both gold annotations for events and semantic similarity.

7.3.3 Semantic Similarity Evaluation

We use cosine similarity function (Equation 7.2) to measure the similarity between the pair of sentence representations \((r_1, r_2)\). To compute the cosine similarity, we take the dot product of the two sentence representations \((r_1, r_2)\) and normalise it by the product of the vector lengths. Values range between -1 and 1, where -1 is perfectly dissimilar and 1 is perfectly similar.

\[ \cos(r_1, r_2) = \frac{r_1 \cdot r_2}{||r_1|| \cdot ||r_2||} = \frac{\sum_{i=1}^{n} r_{1i} \cdot r_{2i}}{\sqrt{\sum_{i=1}^{n} r_{1i}^2} \sqrt{\sum_{i=1}^{n} r_{2i}^2}} \]  (7.2)

Once we have computed the cosine similarity for each sentence, we now compare if there is a linear relationship between the predicted similarity scores \(x\) (i.e., cosine similarity scores) and the gold similarity scores \(y\). To quantify this, we compute the Pearson correlation coefficient \(r\) (Equation 7.3) following Soğancıoğlu et al. (2017). A value of 1 represents a perfect positive relationship, -1 a perfect negative relationship and 0 indicates the absence of relationship between the two variables.

\[ r = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2(y_i - \bar{y})^2}} \]  (7.3)

where \(n\) is the number of sentence pairs, \(x_i\) is the predicted similarity score, \(y_i\) the
gold similarity score, $\bar{x}$ is the mean predicted similarity score and $\bar{y}$ is the mean gold similarity score.

### 7.4 Experimental Settings

In this section, we describe the experimental details on how we use the event representations to the semantic similarity task in the biomedical domain. The event detection model we developed in Chapter 3 is trained on the Cancer Genetics (CG) biomedical dataset (Pyysalo et al., 2015). Consequently, it is reasonable to evaluate the event representations on a semantic similarity task in the biomedical domain.

In the next paragraphs, we provide more details about the datasets used in our experiments.

To the best of our knowledge, there is no available dataset that includes both event structure and semantic similarity in the biomedical domain. There are separate datasets available for event structure detection (Pyysalo et al., 2015) and semantic similarity (Soğancioğlu et al., 2017). We recognise that having no dataset available with both event structure and semantic similarity annotations imposes limitations in our experiments as we cannot evaluate the performance of gold event representations on semantic similarity tasks.

Informed by the constraints mentioned above, we proceed to investigate if the event extraction model trained on a specific biomedical dataset can generate representations that will help in the semantic similarity estimation on the same domain but from a different dataset distribution. We explain this in more detail below.

For the event detection, our event detection model relies on predicted relations. In Chapter 4, we use the state-of-the-art TEES system (Björne and Salakoski, 2018a) to predict the entities and relations on the Cancer Genetics (CG) biomedical dataset (Pyysalo et al., 2015). The TEES system uses a pipeline approach for extracting events.
7.4. EXPERIMENTAL SETTINGS

We have described the pipeline approach in §4.1. The TEES system uses a named entity recognition module that is primarily trained to predict biomedical entities.

We detect events based on the CG dataset (Kim et al., 2009a) which contains forty (40) event types about the biological processes relating to the development and progression of cancer. The CG corpus texts are extracted from the titles and abstracts of the publications from the PubMed literature database. We are aware that the types of events defined in the CG dataset can potentially limit the kinds of events that can be predicted in the biomedical domain.

For the semantic similarity dataset in the biomedical domain, there is only one dataset available, BIOSSES (Soğancioğlu et al., 2017). The BIOSSES dataset comprises 100 sentence pairs, in which each sentence was selected from the TAC (Text Analysis Conference) Biomedical Summarisation Track Training Dataset containing articles from the biomedical domain. The shared task website does not provide information on how articles were chosen from biomedical domain. Hence we assumed that the sentences are drawn randomly from a broader set of possible topics in the biomedical literature and thus are not limited to cancer-related articles. Aside from the inherent event extraction prediction errors, we also recognise that the performance of event detection will also be affected by the difference in the data distribution on which the event detection model was trained on (i.e., cancer-related articles) and on which it will have to predict events (i.e., a wider set of articles with some of them not related to cancer). As we saw in Chapter 3, there is also another related dataset called MedSTS (Wang et al., 2018b) but it is not strictly in the biomedical domain which is our main focus. Hence, we leave the experimentation on the said dataset as future work.

The sentence pairs in the BIOSSES dataset were evaluated by five different annotators and given a score ranging from 0 (no relation) to 4 (equivalent) following

---


the guidelines of SemEval 2012 Task 6 on STS (Agirre et al., 2012). Following Soğancioğlu et al. (2017) we take the average of the five annotator scores to compute the final similarity score for each sentence pair. The dataset has an upper bound Pearson correlation of 0.902 and currently, the reported state-of-the-art performance is 0.871 (Blagec et al., 2019).

To extract the event representations, we use the published single model of the TEES system for the CG dataset 3 to extract the relations on the BIOSSES dataset. Using the TEES system, we predict the relations on the individual sentences (in total 200 sentences). Using the predicted relations, our event detection module predicts the event structures and generate their representations. We use the best search-based model we presented in Chapter 4 for the event detection module. Since the dataset is quite small, we extract events at different thresholds for comparison.

7.4.1 Baseline

As a baseline method, we compare our approach with a sentence-based representation generated using BERT (Devlin et al., 2019a), which stands for Bidirectional Encoder Representations from Transformers. BERT obtained state-of-the-art results on eleven natural language processing tasks in the GLUE (General Language Understanding Evaluation) benchmark (Wang et al., 2018a). It outperformed other pre-trained representations such as ELMO (Embeddings from Language Models) (Peters et al., 2018) and OpenAI GPT (Generative Pre-trained Transformer) (Radford et al., 2018). Both ELMO and OpenAI GPT are unidirectional pre-trained representations where every token can only attend to previous tokens in a left-to-right architecture but this could be harmful for token-level tasks such as question answering, where it is crucial to incorporate context from both directions (Devlin et al., 2019a). BERT alleviates the previous

---

3https://b2share.eudat.eu/api/files/b0f387f7-8fbe-4e5e-9d69-2b03fe2dbe0c/CG13-single.zip, accessed 25 June 2019
unidirectionality constraints as it is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context (Devlin et al., 2019a).

In this work, we generate the sentence representations on two BERT models trained on biomedical corpora, namely: NCBI BlueBERT \(^4\) (Peng et al., 2019d) and BioBERT \(^5\) (Lee et al., 2019) both of which are trained on biomedical corpora. Concretely, NCBI BlueBERT implementation was trained using PubMed abstracts and clinical notes (MIMIC-III) while BioBERT was trained only using PubMed abstracts. For the NCBI BlueBERT model, we used the \texttt{NCBI BlueBERT-Base, Uncased, PubMed+MIMIC-III} pre-trained representations and for the BioBERT we use \texttt{BioBERT v1.1 (+ PubMed 1M)} since these specific models achieved the reported best performance on the BIOSSES dataset. NCBI BlueBERT was reported to achieve state-of-the-art Pearson Correlation score for the BIOSSES dataset at 0.9160 compared to 0.8270 using BioBERT (Peng et al., 2019d). While NCBI BlueBERT has been applied to more datasets, both models provide have been applied to a variety of biomedical text mining tasks such sentence similarity, named entity recognition, relation extraction, document multilabel classification and inference task.

In this work, we intend to compare the performance of both models without any further fine-tuning. To have a fair comparison with our event-based sentence representations we do not fine-tune the BERT models on the BIOSSES dataset and treat the representations as an off-the-shelf pre-trained representations similar to Word2Vec (Mikolov et al., 2013a) or GloVe (Pennington et al., 2014). To achieve this, we extract the sentence representations of the BIOSSES dataset using the respective published models using \texttt{BERT-as-service} \(^6\) (Xiao, 2018) implementation.

Our intention in this work is to compare the representations generated by each

\(^4\)https://github.com/ncbi-nlp/ncbi_bluebert accessed 19 December 2019
\(^5\)https://github.com/naver/biobert-pretrained accessed 19 December 2019
\(^6\)https://github.com/hanxiao/bert-as-service accessed 19 December 2019
respective model with the representations generated by our event model on the performance in sentence similarity task on the BIOSSES dataset.

### 7.5 Results and Discussion

We present in this section the results of the event detection and the semantic similarity evaluation of the event representations.

#### 7.5.1 Baseline Results

Table 7.1 shows the reported versus the actual performance of the two BERT models we investigated in this study as applied on the BIOSSES dataset. We can see that the reported performance for both models are higher than the actual performance. The higher reported performance can be attributed to the fine-tuning of the respective models on the BIOSSES dataset. While both models show a drop in actual performance, the NCBI BlueBERT model has a bigger drop compared to the BioBERT model which suggests that as an off-the-shelf pre-trained representation, BioBERT would likely performed better on generating representations for a sentence similarity task in the biomedical domain.

We describe in the next section the results of our event-based sentence representations on the BIOSSES dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Reported (Peng et al., 2019d)</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCBI BlueBERT</td>
<td>0.9160</td>
<td>0.4723</td>
</tr>
<tr>
<td>BioBERT</td>
<td>0.8270</td>
<td>0.7208</td>
</tr>
</tbody>
</table>

Table 7.1: Comparison of Pearson correlation scores as reported vs the actual performance on the BIOSSES dataset.
7.5. RESULTS AND DISCUSSION

7.5.2 Event Extraction

Since the event detection is performed on the sentence level, we run the TEES system on each sentence individually. Of the 200 individual sentences, the TEES system predicted only 95 sentences with entities or relations. In the other 105 sentences, the TEES system did not predict any entity or relation. This can be attributed to the TEES system’s inherent model performance limitation.

The 95 individual sentences resulted in only 29 sentence pairs (equivalent to 58 sentences). The remaining 37 sentences are those sentences with their sentence pair having no entity or relation predicted. Since entities and relations are needed to construct events, our semantic similarity evaluation is limited to the 29 sentence pairs which is roughly one-third of the size of the original dataset.

Using our event detection module, we detect events for each sentence in the 29 sentence pairs which contain predicted entities or relations. We tested different threshold values to vary the number of events predicted per sentence. Different thresholds affect the number of events predicted in each sentence. This consequently affects how many sentence pairs have event predictions.

We show in Table 7.2 the statistics of event structures and representations extracted at different threshold levels from the 29 sentence pairs (or from all the 58 sentences). In Table 7.2, the different $P$ column labels indicate the paired sentences which total to 29 sentence pairs while the two $U$ column labels indicate the unpaired sentences which total to 37 individual sentences. At a given threshold value, the event detection module may predict events for both sentences in a pair ($P_2$), or only predict events in one of the sentences in the pair ($P_1$) or no events are predicted in the pair ($P_0$).

As expected, we observe that the lower the threshold is the more sentence pairs are predicted with events ($P_2$) and the less sentence pairs with no events $P_0$. For example, with threshold 0.0, all the 29 sentence pairs contain events while with threshold 0.5
only 5 of the sentence pairs have predicted events. Higher prediction thresholds indicate higher confidence of the model on the event predictions. We can see that the number of events predicted decreases with higher thresholds as the model is more precise and accurate in its predictions. The same pattern can also be observed on the unpaired sentences: generally, the number of sentences with predicted events \((U_E)\) decreases as the threshold increases. Conversely, the number of sentences with no predicted events \((U_0)\) increases as the threshold increases.

We describe in the next section how we evaluate the semantic similarity of the predictions at different thresholds.

### 7.5.3 Semantic Similarity Evaluation

We present the results of the semantic similarity evaluation on the predictions at different thresholds discussed in the previous section.

Table 7.3 shows the score distribution of the sentence pairs both in the gold and those predicted at different thresholds. The annotator scores for each sentence pair are discrete values: 0,1,2,3,4. To get a single gold similarity score for a sentence pair,
we averaged the five annotator scores following Soğancıoğlu et al. (2017). When the annotator scores are averaged, we get a continuous value which we place in one of the 5 different bins defined as the following: $0 : [0, 0.5)$, $1 : [0.5, 1.5)$, $2 : [1.5, 2.5)$, $3 : [2.5, 3.5)$, $4 : [3.5, 4.5]$. The intervals in these bins are reasonable because the lower and upper interval values are $\pm 0.5$ from the original categorical values. Table 7.3 shows that most sentence pairs are in the bins 2 or 3 in the gold dataset and the number of sentence pairs at different thresholds follow the same distribution.

<table>
<thead>
<tr>
<th>Data/Threshold</th>
<th>$P_n$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold</td>
<td>100</td>
<td>14</td>
<td>12</td>
<td>28</td>
<td>35</td>
<td>11</td>
</tr>
<tr>
<td>0.5</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.4</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>0.3</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>0.2</td>
<td>13</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>0.1</td>
<td>17</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>0.0</td>
<td>29</td>
<td>4</td>
<td>2</td>
<td>9</td>
<td>9</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 7.3: The distribution of the sentence pairs along different gold similarity scores. Legend: $P_n$ number of sentence pairs, the scores are grouped into the following bins: $0 : [0, 0.5)$, $1 : [0.5, 1.5)$, $2 : [1.5, 2.5)$, $3 : [2.5, 3.5)$, $4 : [3.5, 4.5]$

The number of samples for each bin is quite small even when the threshold is 0.0. To increase the number of samples in each class, we investigate the effect on the performance if we convert the task into a binary classification task instead of a multi-class classification problem. To achieve this, we reduce the number of bins for the scores from five to two bins, defined as follows: $0 : [0, 2.5)$, $1 : [2.5, 4]$. In this way, there are more samples in each class. Table 7.4 shows the reduced bins with more number of samples in each bin at different thresholds.

We now describe how we compute the semantic similarity for the sentence pairs both in the five-bins and two-bin groupings. First, we generate the sentence representation for all sentences as described in §7.3.2. Then, we compute the semantic similarity
Table 7.4: The distribution of the sentence pairs along different gold similarity scores after converting the task into a binary classification. Legend: $P_n$ number of sentence pairs, the scores are grouped into the following bins: 0 : $[0, 2.5)$, 1 : $[2.5, 4]$

<table>
<thead>
<tr>
<th>Threshold</th>
<th>$P_n$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold</td>
<td>100</td>
<td>54</td>
<td>46</td>
</tr>
<tr>
<td>0.5</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>0.4</td>
<td>11</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>0.3</td>
<td>11</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>0.2</td>
<td>13</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>0.1</td>
<td>17</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>0.0</td>
<td>29</td>
<td>15</td>
<td>14</td>
</tr>
</tbody>
</table>

via cosine similarity as described in §7.3.3. Finally, we calculate the Pearson Correlation coefficient, also described in §7.3.3. Table 7.5 shows the results at different thresholds. $P_n$ is the number of sentence pairs, $Model_2$ the Pearson correlation when using two bins, $Model_5$ the Pearson correlation when using five bins.

Table 7.5: Pearson correlation scores at different thresholds and different number of bins using different models: Event, BioBERT and BlueBERT. Legend: $P_n$ number of sentence pairs, $Model_2$ Pearson correlation with two bins, $Model_5$ Pearson correlation with five bins.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>$P_n$</th>
<th>Event2</th>
<th>Event5</th>
<th>BioBERT2</th>
<th>BioBERT5</th>
<th>BlueBERT2</th>
<th>BlueBERT5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>5</td>
<td>0.8167</td>
<td>0.5731</td>
<td>0.9531</td>
<td>0.9580</td>
<td>0.7356</td>
<td>0.8053</td>
</tr>
<tr>
<td>0.4</td>
<td>11</td>
<td>0.4770</td>
<td>0.3660</td>
<td>0.7806</td>
<td>0.6769</td>
<td>0.5258</td>
<td>0.6844</td>
</tr>
<tr>
<td>0.3</td>
<td>11</td>
<td>0.5567</td>
<td>0.3819</td>
<td>0.7806</td>
<td>0.6769</td>
<td>0.5258</td>
<td>0.6844</td>
</tr>
<tr>
<td>0.2</td>
<td>13</td>
<td>0.2727</td>
<td>0.1242</td>
<td>0.4933</td>
<td>0.5346</td>
<td>0.3973</td>
<td>0.3141</td>
</tr>
<tr>
<td>0.1</td>
<td>17</td>
<td>0.0979</td>
<td>-0.0648</td>
<td>0.6116</td>
<td>0.6559</td>
<td>0.5093</td>
<td>0.3899</td>
</tr>
<tr>
<td>0.0</td>
<td>29</td>
<td>0.0960</td>
<td>-0.0485</td>
<td>0.7656</td>
<td>0.6508</td>
<td>0.3896</td>
<td>0.4328</td>
</tr>
</tbody>
</table>

Table 7.5 shows that grouping the samples into two bins ($Event_2$) yields higher Pearson correlation than with five bins ($Event_5$). This can be attributed to the increase of samples in each bin when number of bins is two. Moreover, we can observe that the Pearson correlation value in both bins decreases as the threshold value decreases. This is expected because at the higher thresholds the events predicted are more likely to be the real events while at the lower thresholds errors may be introduced in the event
prediction. The relatively lower scores in five bins (Event5) compared to the two bins (Event2) is justifiable because of the more distributed scores along the different bins.

Between the BERT models, the BioBERT model yields a better performance than BlueBERT which is consistent with its baseline performance. This implies the reliability of BioBERT compared to BlueBERT as an off-the-shelf pre-trained representations for semantic similarity task. As regards the performance of the BERT models in different bins, there is not a clear pattern where either bin is better than the other. However, the general trend shows that the scores are decreasing as the threshold value decreases except for a few outliers. This indicates the reliability of event-based predictions determined by the threshold values in generating representations for semantic representation. What is clear in the result is that the highest scores in BioBERT and BlueBERT are in threshold 0.5 which corresponds to the higher confidence in event predictions. This suggests that the event representations contain the core semantic representation or at least equivalent to what is learned in the BERT models.

In summary, the results in Table 7.5 demonstrate that the events predicted at higher thresholds provide better information for semantic similarity. While the results are not conclusive because of the limited data, the results in Table 7.5 indicate that event structures and their representations can be used as effective sentence representation for semantic similarity estimation.

To gain further understanding on the effect of event representations on semantic similarity, we conducted an analysis in the next section.

7.5.4 Analysis

We now investigate how event predictions affect the semantic similarity performance.

Concretely, we examine the events predicted on a set of sentence pairs and the cosine similarity of the sentence representations. We choose the sentence pairs and the
predictions at threshold 0.5 because of the higher confidence of the predictions. Table 7.6 shows the five sentence pairs predicted with events at threshold equals 0.5 together with their gold (at different bins) and predicted similarity scores.

<table>
<thead>
<tr>
<th>Sentence Pair ID</th>
<th>Gold Score (bin 5)</th>
<th>Gold Score (bin 2)</th>
<th>Event</th>
<th>BioBERT</th>
<th>BlueBERT</th>
</tr>
</thead>
<tbody>
<tr>
<td>89</td>
<td>0</td>
<td>0</td>
<td>0.8134</td>
<td>0.9180</td>
<td>0.8853</td>
</tr>
<tr>
<td>83</td>
<td>0</td>
<td>0</td>
<td>0.8993</td>
<td>0.9106</td>
<td>0.8645</td>
</tr>
<tr>
<td>31</td>
<td>4</td>
<td>1</td>
<td>0.9994</td>
<td>0.9725</td>
<td>0.9186</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>0</td>
<td>0.7626</td>
<td>0.9342</td>
<td>0.9200</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>1</td>
<td>0.9319</td>
<td>0.9735</td>
<td>0.9552</td>
</tr>
</tbody>
</table>

Table 7.6: Comparison of the gold (at different bins) and predicted cosine similarity scores of the sentence pairs for the Event model at threshold=0.5. Shown also are the corresponding cosine similarity scores using the BioBERT and BlueBERT models.

Table 7.6 shows that sentence pair 31, which has a gold similarity score of 4 at bin=5 and 1 at bin=2, results in the highest cosine similarity (0.9994) among the pairs. This is followed by sentence pair 5 with cosine similarity score of 0.9319. The gold scores and cosine similarity scores follow the same pattern for the rest of the sentence pairs. The linear relationship between the gold similarity scores and the cosine similarity scores in Table 7.6 confirm that event representations can be used to compare semantic similarity between sentences. The relatively high cosine similarity scores in Table 7.6 which is not proportional to the gold scores (i.e., gold similarity score of 0 should have been equal to cosine similarity score of -1) is reasonable because the model has not been trained to generate representations which considers both the events and semantic similarity annotations in a joint modelling manner.

As far as the BERT models are concerned, BioBERT follows similar trend with the gold scores in both the two and five bins, which also follows the Event scores pattern. However this is not the case for BlueBERT except for the sentence pairs with lowest gold scores in both bins. This observation is a further confirmation that BioBERT is a more reliable sentence similarity model than BlueBERT.
7.5. RESULTS AND DISCUSSION

We now further examine how the actual events predicted in each sentence pair influenced the cosine similarity scores. We examine two sentence pairs: the one with the highest gold or predicted similarity score (sentence pair 31) and the one with the lowest gold similarity (also lower predicted similarity) score (sentence pair 89). We pick sentence pair 89 over sentence pair 18, which has the lowest cosine similarity score, since we want to know the kind of events predicted with reference to the gold similarity scores. As mentioned in the previous paragraph, the resulting cosine similarities are such because the generated event representations were not trained considering both the event and semantic similarity annotations.

Sentence pair 31 consists of the following sentences.

**Sentence 1:** *Furthermore, ablation of both Erk1 and Erk2 impaired tumor development, whereas inactivation of either one alone had no effect.*

**Sentence 2:** *Only concomitant ablation of ERK1 and ERK2 impairs tumor growth.*

The following are the entities, triggers and events predicted for Sentence 1. Each entity or trigger has the following information: trigger or entity id (e.g. T1), trigger or entity type (e.g. Protein), location in the sentence [start, end] (e.g. 30 34) and the actual word mention in the sentence of the detected trigger or entity type (e.g. Erk1). Every event contains the event id (eg. E1), the event type (e.g. Metastasis) with the trigger id (e.g. TR5). Events can have zero or more arguments. In this example, Event E1 does not have any argument.

**Entities 1:**

T1 Protein 30 34 Erk1
T2 Protein 39 43 Erk2

**Triggers and Events 1:**

TR3 Mutation 13 21 ablation
TR4 Negative_regulation 44 52 impaired
The following are the entities, triggers and events predicted for Sentence 2.

**Entities 2:**

T1 Protein 29 33 ERK1
T2 Protein 38 42 ERK2

**Triggers and Events 2:**

TR3 Mutation 17 25 ablation
TR4 Negative_regulation 17 25 ablation
TR5 Negative_regulation 43 50 impairs
TR6 Metastasis 51 56 tumor
TR7 Growth 57 63 growth
E1 Metastasis:TR6

We can see in the predictions above that our event detection model predicted the same set of entities in both sentences and the same events but different set of triggers. Note that we only used the events and their representations to compare sentence pairs. In this case, our model predicted the same event for both sentences: *Metastasis* from the word mention *tumor*. This shows that events and their representations can be used as a basis to compare the semantic similarity between pairs of sentences.

We now examine another sentence pair (Sentence pair 89) with lower gold similarity and cosine similarity score.
Sentence 1: With respect to LATS2, it has been reported that LATS2 induces G2/M arrest and subsequent apoptotic cell death.

Sentence 2: The expression of miR-146a has been found to be up-regulated in papillary thyroid carcinoma, anaplastic thyroid cancer and cervical cancer.

The following are the entities, triggers and events predicted for Sentence 1.

**Entities 1:**

T1 Protein 16 21 LATS2  
T2 Protein 49 54 LATS2

**Triggers and Events 1:**

TR3 Positive_regulation 55 62 induces  
TR4 Negative_regulation 68 74 arrest  
TR5 Death 105 110 death  
E1 Death:TR5

The following are the entities, triggers and events predicted for Sentence 2.

**Entities 2:**

T1 Protein 18 26 miR-146a

**Triggers and Events 2:**

T2 Gene_expression 4 14 expression  
T3 Positive_regulation 48 60 up-regulated  
T4 Metastasis 82 91 carcinoma  
T5 Metastasis 112 118 cancer  
T6 Metastasis 133 139 cancer  
E1 Gene_expression:T2  
E2 Positive_regulation:T3 Theme:E1
We can see that sentence pair 89 have a different set of entities, triggers and events predicted in each sentence and this explains why it has a comparatively lower similarity score.

The two examples above discussed show that the event structures detected in each sentence indeed influenced the cosine similarity scores via the event representations. This indicates that event representations can be used as basis for comparing the semantic similarity between sentences.

Furthermore, we also observe that the statements above are cancer-related. This is expected because the event detection model was trained on cancer-related dataset, hence, it can predict events on cancer-related statements. We believe that having an event detection model that is trained to predict events on a wider set of event types will result in more sentence pairs to be predicted with events.

In summary, both event-based representations and BioBERT representations show some higher degree of reliability as a sentence similarity metric than BlueBERT. Deciding which of these two representations is better would depend on the need. If there is a need for explainability, we think that the event-based representations would be better as the similarity scores are supported by the actual event structures predicted in each sentence. However, increasing the recall on event extraction remains a challenge and thus requires improvement in other components in the event extraction pipeline such as in named entity recognition and relation extraction. This limitation is where the BERT representations in general show its strength as they can generate representations for every sentence. The main drawback of these representations is they are not explainable in themselves in the way event-based representations are.
7.6 Conclusion and Future Work

In this chapter, we have investigated how event representations can be used for semantic similarity estimation between pairs of sentences. We based our experiments on the biomedical domain and extracted the events from the BIOSSES semantic similarity dataset and generated the sentence representations. We computed the semantic similarity using cosine similarity measure and measure the Pearson correlation between the gold similarity scores and the predicted cosine similarity scores. Although the results are not conclusive because of the small size of the dataset, the results and the detailed analysis indicate that event representations can be used as sentence representation in a semantic similarity task. Specifically, the analysis shows that those sentence pairs with the same events predicted resulted in higher cosine similarity scores while those sentences with different events predicted resulted in lower cosine similarity scores. Furthermore, the cosine similarity scores showed linear relationship with the gold similarity scores as measured by the Pearson correlation. Our results also showed that event-based representations for predictions at higher thresholds correspond also to higher Pearson correlation scores using the BERT models indicating that event-based representations contain the same level of semantics as learned by the BERT models.

There are many avenues to extend this work. One way is to construct a dataset with both event and semantic similarity annotations. For example, the BIOSSES dataset can be annotated with event structures. While this will be a work-intensive effort, this will enable joint training of event detection and semantic similarity in the biomedical domain.

Since it is likely that event annotation requires more domain expertise than semantic similarity annotation, the converse annotation process could be addressed instead. Semantic similarity datasets can be created from existing event datasets. For example, we can construct semantic similarity datasets from the event corpora used by Miwa
et al. (2013b) to construct a wide coverage biomedical event extraction system. Following this direction will also widen the range of event types that can be predicted.

With a bigger dataset with both event and semantic similarity annotations, the methods described in this chapter can be substantially improved. We can train a neural network that predicts the semantic similarity based on the events. Furthermore, we can explore more sophisticated aggregation functions (Bahdanau et al., 2014) to create a single representation out of the multiple events predicted in a sentence.

Aside from dataset annotation efforts, other learning techniques could also be investigated. On the one hand, distant learning approaches (Pershina et al., 2014; Mintz et al., 2009) can be used to add more data to the BIOSSES dataset. On the other hand, zero-shot (Romera-Paredes and Torr, 2015; Palatucci et al., 2009; Socher et al., 2013), one-shot (Duan et al., 2017) or few-shots (Snell et al., 2017; Triantafillou et al., 2017) learning methods can also be explored to learn from very limited data such as the BIOSSES dataset.
Chapter 8

Conclusion

In this chapter, we revisit the research objectives we have established in Section 1.2 and describe how we fulfilled each one of them. We also enumerate the possible avenues and recommendations on extending our work in the future.

8.1 Evaluation of Research Objectives

In order to address research question $Q_1$ (What is the state-of-the-art in event detection and its limitations?), we established objective $O_1$.

$O_1$: *To conduct a review of existing work on event detection covering general and biomedical domains and analyse the limitations of the state-of-the-art*

We began in Chapter 3 by defining what an event structure is in Section 3.1 and discussing its properties such as being structure-invariant. We also discussed how events are represented especially in the case of nested and overlapping events. We pointed out the tree data structures are not able to represent complex events and we proposed to represent them as DAG structures. We then defined the overall task of event extraction in Section 3.2 and discussed the different sub-tasks involved. We enumerated the major approaches to event extraction and underlined that our focus is on sentence-level event
CHAPTER 8. CONCLUSION

detection. In Section 3.3, we surveyed the various event detection approaches. We first examined the datasets from different domains and the state-of-the-art approaches for each dataset. We found that in the general domain, the ACE 2005 dataset is the most popular in the research community to evaluate event detection while in the biomedical domain, there are several datasets that have been created and mostly evaluated through the BioNLP shared tasks over the years. Based on the reported performance in the literature, the state-of-the-art model has surpassed the human baseline performance for the ACE 2005 dataset while in the biomedical datasets there is still room for improvement before it reaches the human-level performance, indicating that the biomedical event detection task is a difficult task and for the most part due to the complexity of the event structures involved. The survey also showed that most of the state-of-the-art approaches are neural-based. We then analysed the limitations of the different approaches in Section 3.3.3 and introduced our proposed approach in 3.4. Our proposed model is a novel search-based neural network structured prediction model that treats the task of event detection as a search problem on a relation graph of trigger-argument structures, which we described in Section 3.4.2. We also developed a baseline model (described in Section 3.4.3) using Tree-LSTM to classify event structures. As a whole, Chapter 3 of this thesis directly answers research question $Q_1$.

In order to address research question $Q_2$ (Is it possible to develop a deep learning-based method for biomedical event structure detection and representation without relying on hand-crafted features?), we established objective $O_2$.

$O_2$: To develop a neural-based method for detecting nested and overlapping events in the biomedical domain

We fulfilled this research objective by evaluating our proposed neural-based models on a biomedical event dataset in two experimental setups, namely, in a pipeline scenario (§4.1) and in comparison with the state-of-the-art on event detection (§4.2). Our results show that both in the pipeline and model evaluation experiments that the search-based
model yields state-of-the-art performance while being more computationally efficient. Furthermore, detailed analyses on the development set showed that our model yielded higher F1-score performance on both nested and overlapping event detection. We also have showed the flexibility of our model by varying the beam width which resulted in better performance. Moreover, in contrast to the state-of-the-art event detection model, our deep learning model did not use any hand-crafted features and syntactic features. As a whole, these findings fulfill our research objective \( O_2 \).

In order to address research question \( Q_3 \) (What is the state-of-the-art in sentence representation and its limitations?), we established objective \( O_3 \).

\( O_3 \): To conduct a review of existing work on text semantic representation in both general and biomedical domain and classify them into a framework that considers input granularity and model complexity and then investigate the limitations of the state-of-the-art

We began by examining the current approaches on representing the meaning of constituents in a sentence, as described in Section 5.1. While much research has been directed into constructing representations for constituents, we found that there has been far less consensus on how to represent larger semantic structures such as phrases and sentences. Since the best way to evaluate sentence meaning representations is to see how they perform on actual natural language inference tasks, we examined three related tasks and the corresponding datasets mainly in the general and biomedical/clinical domains. We then surveyed the state-of-the-art methods in each dataset, as detailed in Section 5.2. We found that methods vary depending on the dataset, from rule-based to feature-based machine learning to deep learning methods. Since our focus is on deep learning methods, we narrowed down our analysis and took a closer look at deep learning-based methods for semantic similarity estimation, as discussed in Section 5.3. We discovered that the approaches lie in two dimensions: input representation granularity and composition model complexity. Furthermore, using the
framework we developed, we found that higher level semantic structures such as event structures are under-explored as input representation. To gain a deeper understanding on the limitations of the state-of-the-art model, we implemented a simplified version of the state-of-the-art model for the semantic similarity task, as described in Chapter 6. We conducted error analysis and reported the limitations of the current state-of-the-art. We found that one of the datasets obtained low performance due to its noisy nature and we investigated the effect of the noisy data in Section 6.5 in a named entity recognition task. We found that adding weakly labelled for sparse entity types noticeably increased the overall performance. These results address our research question $Q_3$.

In order to address research question $Q_4$ (Can event structure representation be used as sentence representation on biomedical semantic textual similarity task and to what extent is it effective?), we established objective $O_4$.

$O_4$: To generate sentence representation from event representations and evaluate them on semantic textual similarity task in the biomedical domain

To fulfil this objective, firstly, we identified the semantic similarity dataset that we can apply our event representations. Based on the results in Chapter 5, to date there is only one semantic similarity dataset in the biomedical domain. Furthermore, the dataset is relatively small compared to the general domain datasets, due to the difficulty in the annotation process since this involves specialised domain knowledge. Since we want to evaluate the event representations discussed in the previous chapter, we outlined our methodology from event detection to sentence representation and semantic similarity evaluation, as described in Section 7.3. We compared our approach by creating two baselines which used BERT models, namely, BlueBERT and BioBERT. The baseline results showed that the BioBERT is a better alternative as an off-the-shelf tool for sentence representation compared to BlueBERT. For the event-based representations, we extracted the relations from the semantic similarity dataset using the state-of-the-art TEES system, however, the system returned only a small subset of the dataset with
entities or relations. This significantly limited the event representations that our event detection module can generate. At the same time, this is also expected as the dataset distribution on which the TEES system was trained on is different from the dataset distribution of the semantic similarity dataset. Particularly, the TEES system was only trained on cancer-related articles while the semantic similarity dataset was drawn from the whole biomedical domain. Furthermore, the event types are limited to what was defined on the training data that the TEES system was trained on. Recognising these limitations, we focussed our analysis on the sentence pairs which have predicted entities and relations. Using our event detection module, we predict event structures and generate event representations given the predicted relations on the semantic similarity dataset. We experimented with different threshold values of our model. With the limited number of samples, we compared the gold similarity scores against the cosine similarity scores obtained by comparing the sentence representations of each sentence pair. As discussed in Section 7.5, as expected, we found that higher threshold values resulted in higher Pearson correlation values. This indicates that the event representations generated cosine similarity scores proportional to the gold similarity scores. The BERT models also yielded the highest Pearson correlation score for threshold 0.5 which suggests that event representations contain the core semantic representation or at least equivalent to what is learned in the BERT models. Furthermore, this was confirmed in our analysis in Section 7.5.4. These results, though yet limited by the data available, indicate that event representations can be used as sentence representations in semantic similarity task, and thus answers our research question $Q_4$.

After having fulfilled the research objectives, the findings we obtained which was summarised above prove our research hypotheses which we formulated in the beginning:

$H_1$: Existing methods in event detection require hand-crafted features and do not consider nested and overlapping event structures simultaneously during learning.
CHAPTER 8. CONCLUSION

$H_2$: It is possible to develop a deep learning method without reliance on hand-crafted features in order to detect biomedical nested and overlapping event structures.

$H_3$: Existing methods in text semantics representation can be described in a framework which considers the input representation granularity and composition model complexity and this framework can point out the areas that are under-explored.

$H_4$: Event representations can be used as sentence representations in biomedical semantic similarity task.

8.2 Future Work

The work presented in this thesis can be extended by exploring the use of event representation in actual search-based systems. To accomplish that, the event detection model developed in this work must be trained on a wider set of event types and on a bigger dataset. Furthermore, having a jointly annotated dataset for events and semantic similarity will improve the performance because it opens the possibility to jointly trained a model based on these information.

The event detection model can also be extended to detect nested and overlapping event structures on the document level. Moreover, the event detection model itself can also be further improved. Model extensions can include enriching the input representation, exploring the use different aggregation functions for the relation representations, implementing a dynamic beam width searching and experimenting with additional actions.

While the focus of this work is on detecting and representing nested event structures, the model developed can also be applied to nested or discontiguous entities (Ju et al., 2018). Lastly, since our focus on this study is largely on the biomedical domain, another path of work is to apply and evaluate the event representations to semantic similarity tasks in the general domain. To accomplish this research, a dataset with nested
8.2. FUTURE WORK

event structures needs to be developed for the general domain to enable more complex event representations.
Appendix A

Computation of Partial Derivatives

Let $\delta^{(l)}_j$ be the error of node $j$ in layer $l$, the partial derivatives are given by Equation A.1 ignoring the regularisation term.

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = a^{(l)}_j \delta^{(l+1)}_i = D_{ij}^{(l)} \quad \text{(A.1)}$$

In this section, we show the derivation of the gradients $\delta^{(l)}$ referred in Lines 6-7 of Algorithm 1.

Let $J$ be our cost function, $z$ the linear function, $a$ the activation function, $\Theta$ the weights, $b$ the bias and $x$ the input. We start by computing the derivative of our cost function with respect to (w.r.t) each of the preceding elements in the neural network by using chain rule. We don’t take the derivative of our input $x$ since they are given.

$$\frac{\partial J}{\partial a} \frac{\partial J}{\partial z} \frac{\partial J}{\partial \Theta} \frac{\partial J}{\partial b} \quad \text{(A.2)}$$

A.1 Derivative of the cost function $J$ w.r.t the activation function $a$

We start by taking the derivative of the cost function w.r.t the activation function.
A.2 DERIVATIVE OF THE ACTIVATION FUNCTION $a$

We take the derivative of the activation function which is the sigmoid function defined as:

$$a = \frac{1}{1 + e^{-z}} \quad \text{(A.3)}$$

Since the $a$ can also be written as: $(1 + e^{-z})^{-1}$, we can now apply the chain rule to differentiate. Note that the derivative of inner function is as follows: $1 + e^{-z} = e^{-z} \cdot (e^{-z})' = e^{-z} \cdot (-1)$. 

The derivative of the activation function $a$ is:

$$\frac{\partial J}{\partial a} = - y \ln(a) - (1 - y) \ln(1 - a)$$

Since $\ln(x)' = \frac{1}{x}$, the derivative of the first term becomes: $-\frac{y}{a}$.

The second term is composed of two expressions $(1 - y)$ and $\ln(1 - a)$. The derivative of the first expression w.r.t to the activation is the same expression: $(1 - y)$ since it has no variable $a$. The second expression involves a composite function. Since $\ln(f(x))' = \frac{1}{g(x)}g'(x)$, so $\ln(1 - a) = \frac{1}{1 - a} (1 - a)'$. Since the derivative of $1 - a$ w.r.t to $a$ is -1, so the second expression results in: $-\frac{1}{1 - a}$. Plugging back the results of the first term and the second term, we have:

$$\frac{\partial J}{\partial a} = - y \frac{a}{a} - \left[ (1 - y) \frac{-1}{1 - a} \right]$$

$$\frac{\partial J}{\partial a} = - y \frac{1 - y}{1 - a}$$
We can expand the equation as a product of two functions:

\[
\frac{\partial a}{\partial z} = \frac{e^{-z}}{(1 + e^{-z})^2}
\]

Since the first term \( \frac{1}{1+e^{-z}} \) is the activation function \( a \), so we can simplify it as:

\[
\frac{\partial a}{\partial z} = a \cdot (1 - a)
\]

### A.3 Derivative of the cost function \( J \) w.r.t the linear function \( z \)

The derivative of the cost function w.r.t the linear function is equal to:

\[
\frac{\partial J}{\partial z} = \frac{\partial J}{\partial a} \frac{\partial a}{\partial z}
\]

Since we already have computed \( \frac{\partial J}{\partial a} \) in §A.1 and \( \frac{\partial a}{\partial z} \) in §A.2, we can just plug in the results and simplify:
A.4 Derivative of the linear function \( z \) w.r.t \( \Theta \)

We take the derivative of the linear function \( z = \Theta x + b \).

\[
\frac{\partial z}{\partial \Theta} = \Theta x + b = x
\]

A.5 Derivative of the linear function \( z \) w.r.t \( b \)

We take the derivative of the linear function \( z = \Theta x + b \).

\[
\frac{\partial z}{\partial b} = \Theta x + b = 1
\]

A.6 Derivative of the cost function \( J \) w.r.t the weights \( \Theta \)

The derivative of the cost function \( J \) w.r.t the weights \( \Theta \) is given as:

\[
\frac{\partial J}{\partial a} = \Theta x + b
\]
\[ \frac{\partial J}{\partial \Theta} = \frac{\partial J}{\partial a} \frac{\partial a}{\partial z} \frac{\partial z}{\partial \Theta} \]

The first two terms: \( \frac{\partial J}{\partial a} \frac{\partial a}{\partial z} \) is equal to \( \frac{\partial J}{\partial z} \) which we have computed in §A.3 while the third term \( \frac{\partial z}{\partial \Theta} \) we already have computed in §A.4. Hence, we used those results and simplify as follows:

\[ \frac{\partial J}{\partial \Theta} = \frac{\partial J}{\partial z} \frac{\partial z}{\partial \Theta} \]

\[ = \frac{\partial J}{\partial z} \frac{\partial z}{\partial \Theta} \]

\[ = \frac{\partial J}{\partial z} \frac{\partial z}{\partial \Theta} \]

\[ = \frac{\partial J}{\partial z} \frac{\partial z}{\partial \Theta} \]

\[ = (a - y)x \]

\[ = x(a - y) \]

**A.7 Derivative of the cost function J w.r.t the bias b**

The derivative of the cost function J w.r.t the bias b is given as:

\[ \frac{\partial J}{\partial \Theta} = \frac{\partial J}{\partial z} \frac{\partial z}{\partial b} \]

Since the \( \frac{\partial J}{\partial z} \) is already computed in §A.3 and \( \frac{\partial z}{\partial b} \) in §A.5, we can use those results and simplify.
\[
\frac{\partial J}{\partial \Theta} = \frac{\partial J}{\partial z} \cdot \frac{\partial z}{\partial b} = (a - y) \cdot 1 = a - y
\]


Agirre, E., Gonzalez-Agirre, A., Lopez-Gazpio, I., Maritxalar, M., Rigau, G., and


Björne, J. and Salakoski, T. (2018b). Biomedical event extraction using convolutional


Bibliography


workshop on natural language processing in biomedicine and its applications, pages 70–75. Citeseer.


Association for Computational Linguistics and the 7th International Joint Confer-
ence on Natural Language Processing (Volume 2: Short Papers), pages 365–371,
Beijing, China. Association for Computational Linguistics.

Nguyen, T. H. and Grishman, R. (2018). Graph convolutional networks with argument-
aware pooling for event detection. In Thirty-Second AAAI Conference on Artificial
Intelligence.

cedings of the 8th International Workshop on Parsing Technologies (IWPT), pages
149–160, Nancy, France.

Nivre, J. (2006). Inductive dependency parsing, volume 34 of Text, Speech and Lan-
guage Technology. Springer Netherlands, 1 edition.


York.

Ohta, T., Pyysalo, S., Rak, R., Rowley, A., Chun, H.-W., Jung, S.-J., Choi, S.-P.,
Ananiadou, S., and Tsujii, J. (2013). Overview of the pathway curation (PC) task

Ohta, T., Pyysalo, S., and Tsujii, J. (2011). Overview of the epigenetics and post-
translational modifications (EPI) task of BioNLP shared task 2011. In Proceedings


Pershina, M., Min, B., Xu, W., and Grishman, R. (2014). Infusion of labeled data into distant supervision for relation extraction. In *Proceedings of the 52nd Annual


Riedel, S. and McCallum, A. (2011b). Robust biomedical event extraction with dual
decomposition and minimal domain adaptation. In Proceedings of BioNLP Shared
Task 2011 Workshop, pages 46–50, Portland, Oregon, USA. Association for Com-
putational Linguistics.

mathematical statistics, pages 400–407.

auditory pathway in ferrets: receptive fields of visual neurons in primary auditory

Romera-Paredes, B. and Torr, P. (2015). An embarrassingly simple approach to zero-

Rosenblatt, F. (1957). The perceptron, a perceiving and recognizing automaton Project
Para. Cornell Aeronautical Laboratory.

abs/1609.04747.


323:533–536.

Ruppenhofer, J., Ellsworth, M., Schwarzer-Petruck, M., Johnson, C. R., and Schef-

Rychalska, B., Pakulska, K., Chodorowska, K., Walczak, W., and Andruszkiewicz,


pooling and unfolding recursive autoencoders for paraphrase detection. In Advances

Socher, R., Lin, C. C., Manning, C., and Ng, A. Y. (2011b). Parsing natural scenes

similarity estimation system for the biomedical domain. Bioinformatics, 33(14):i49–
i58.

semantic parser. In Proceedings of the Events and Stories in the News Workshop,


of the wnut16 named entity recognition shared task. In Proceedings of the 2nd
Workshop on Noisy User-generated Text, pages 138–144.

of the wnut16 named entity recognition shared task. In Proceedings of the Workshop
on Noisy User-generated Text (WNUT 2016), Osaka, Japan.

Sun, W., Rumshisky, A., and Uzuner, O. (2013). Evaluating temporal relations in
clinical text: 2012 i2b2 challenge. Journal of the American Medical Informatics


Venugopal, D., Chen, C., Gogate, V., and Ng, V. (2014b). Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features. In


