

4-28-2021 10:00 AM

# Generating Effective Sentence Representations: Deep Learning and Reinforcement Learning Approaches

Mahtab Ahmed, *The University of Western Ontario*

Supervisor: Mercer, Robert E., *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Computer Science

© Mahtab Ahmed 2021

Follow this and additional works at: <https://ir.lib.uwo.ca/etd>



Part of the [Artificial Intelligence and Robotics Commons](#)

---

## Recommended Citation

Ahmed, Mahtab, "Generating Effective Sentence Representations: Deep Learning and Reinforcement Learning Approaches" (2021). *Electronic Thesis and Dissertation Repository*. 7780.  
<https://ir.lib.uwo.ca/etd/7780>

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact [wlsadmin@uwo.ca](mailto:wlsadmin@uwo.ca).

# Abstract

Natural language processing (NLP) is one of the most important technologies of the information age. Understanding complex language utterances is also a crucial part of artificial intelligence. Many Natural Language applications are powered by machine learning models performing a large variety of underlying tasks. Recently, deep learning approaches have obtained very high performance across many NLP tasks. In order to achieve this high level of performance, it is crucial for computers to have an appropriate representation of sentences. The tasks addressed in the thesis are best approached having shallow semantic representations. These representations are vectors that are then embedded in a semantic space. We present a variety of novel approaches in deep learning applied to NLP for generating effective sentence representations in this space. These semantic representations can either be general or task-specific. We focus on learning task-specific sentence representations, where often these tasks have a good amount of overlap. We design a set of general purpose and task specific sentence encoders combining both word-level semantic knowledge and word- and sentence-level syntactic information. As a method for the former, we perform an intelligent amalgamation of word vectors using modern deep learning modules. For the latter, we use word-level knowledge, such as parts of speech, spelling, and suffix features, and sentence-level information drawn from natural language parse trees which provide the hierarchical structure of a sentence together with grammatical relations between the words. Further expertise is added with reinforcement learning which guides a machine learning model through a reward-penalty game. Rather than just striving for good performance, we always try to design models that are more transparent and explainable. We provide an intuitive explanation about the design of each model and how the model is making a decision. Our extensive experiments show that these models achieve competitive performance compared with the currently available state-of-the-art generalized and task-specific sentence encoders. All but one of the tasks dealt with English language texts. The multilingual semantic similarity task required creating a multilingual corpus for which we provide a novel semi-supervised approach to make artificial negative samples in the presence of just positive samples.

**Keywords:** Sequence Labelling, Long Short Term Memory, Neural Sequence to Sequence Learning, Word Sense Disambiguation, Tree structured Long Short Term Memory, Tree Attention, Protein-Protein Interaction, Structured Attention, Tree Transformer, Multi-head attention, Multi-branch attention, Natural language inference, Dependency parsing, Transformer, Semantic Similarity, Paraphrase Identification, Question Answer Pairing, Relational Memory Core, Multilingual Semantic Textual Similarity, Topic Modeling, Reinforcement Learning, Sentence Representation.

## **Lay Abstract**

The goal of Computational Linguistics is to analyze and process human language automatically by computers. To help achieve this goal, Natural Language Processing (NLP) based models, actualized by Artificial Intelligence (AI) algorithms, are being incorporated in increasingly intelligent computer applications at a rapid pace. These NLP models are being used in the language related aspects of publishing, healthcare, banking, advertising and insurance industries to improve their customer services and enterprise activities. Certain NLP tasks are fundamental to this thesis: paraphrase identification, sentence similarity, question answering, sentiment analysis, and sentence compression. Deep learning, an AI technique that is being applied more and more, improves the functionality and robustness of the solutions for these tasks. Sentence similarity analysis and paraphrasing is often used to check the originality of a document and prevent plagiarism as well as helping in natural language understanding. Question answering improves customer services and can enhance administrative activities by allowing end-users to ask and get responses about different services and products in their preferred language. The ability of deep learning-based models to begin to handle this kind of diversity is starting to make communication between people from various corners of the world possible in their own language. The automated answering models advance the administrative task of the enterprises by reducing customer service costs as well as saving office time. Sentiment analysis quantifies subjective information, extracts affective states, and is widely used in identifying customer feedback such as survey responses, movie reviews, and healthcare materials. Text compression tries to create a representative summary or abstract of a text piece by finding the most informative concepts.

Research in the field of AI is currently attempting to achieve human-level performance on these aforementioned tasks. In order to achieve this performance level, it is crucial for computers to have an appropriate representation of the sentences. The term representation in this case means a set of numbers (i.e., a vector). Sentences having similar meaning should be represented by almost similar sets of numbers. This thesis develops methods to find good representations of sentences using the modern AI techniques, deep learning and reinforcement learning.

## Co-Authorship Statement

This thesis is written in an integrated article format. A total of 12 papers comprise this thesis. All of the papers related to this thesis are either published, in press, or to be submitted. The author of this dissertation is the primary author of all of the papers.

Chapters 2, 3, 4, 5, and 8 are co-authored with Muhammad Rifayat Samee and Dr. Robert E. Mercer. Muhammad Rifayat Samee, a former MSc graduate student in Dr. Mercer's lab, performed the literature survey and helped with designing the model. Mahtab Ahmed, the author of this dissertation, performed the analysis and interpretation of the data, carried out the implementation, and drafted the manuscripts for the publication. Dr. Robert E. Mercer contributed with his valuable intuition, provided insights that helped in designing the model, and also helped in the preparation of the manuscripts for publication.

For Chapter 5, Jumayel Islam, the second co-author and former MSc graduate student in Dr. Mercer's lab, helped in pre-processing the datasets.

Chapters 6, 7, 9, 10, and 11 are co-authored with Dr. Robert E. Mercer. Mahtab Ahmed, the author of this dissertation, carried out the literature survey, developed the approach, formulated the design of the study, carried out the implementation, and drafted the manuscripts for the publication. As the supervisor of this research, Dr. Robert E. Mercer has participated in the design of the study, interpretation of its results, and preparation of the manuscripts for publication.

Chapters 12 and 13 result from a project under Mitacs Grant IT12388 with the company Messagepoint, Inc. These two works are co-authored with Chahna Dixit, Dr. Robert E. Mercer, Atif Khan, Muhammad Rifayat Samee, and Felipe Urrea. Mahtab Ahmed, the author of this dissertation, performed the background research, designed the sketch of the solution, performed the final implementation, and drafted the manuscripts for the publication. Chahna Dixit, helped in pre-processing the datasets, training the model, and drafting the manuscript. Muhammad Rifayat Samee assisted with implementing the negative sample generation. Felipe Urrea helped in exploring some other potential solutions to the problem. Atif Khan, Vice President, AI and Data Science at Messagepoint, and Dr. Robert E. Mercer helped with their valuable insights, provided some hard examples where the solution may fail, and proofread the final version of the manuscripts for publication.

## Acknowledgements

Above all, I am grateful to Allah for His blessings and mercy on me. I am undeniably His, and I will return to Him. He provided me with the power, experience, enthusiasm, perseverance, and stamina I needed to complete this project on time and with this much energy.

I would like to express my sincere appreciation and thanks to my supervisor Dr. Robert E. Mercer. He is among the best mentors I have found so far. He was always enthusiastic about my ideas, appreciating our achievements, and encouraging during all of the research blocks. Most importantly, he was always smiling! His methods have guided me towards the completeness as an independent researcher.

My sincere thanks to my lab colleagues and friends Sudipta, Samee, Jumayel, Felipe, Xindi, and Anemily. Their invaluable suggestions on my work during the lab meetings were very helpful. Especially, Samee helped me a lot through his problem solving skills and motivated me to go ahead and try every new idea that I suddenly came up with. Special thanks to Janice for providing me all the documents and help whenever I needed them. Thanks to all the “third-floor” ladies in the main office. You were simply superb!

I would also like to thank my Messagepoint colleagues Atif Khan and Chahna Dixit. They were there with me in most parts of my PhD journey. Their valuable insights and ideas helped me in many parts of my thesis. Working closely on an industry project during academic studies was a huge opportunity for me and I feel really blessed to have had this.

I also acknowledge the insightful thoughts and valuable feedback from my thesis examiners: Dr. Leila Kosseim, Dr. Katarina Grolinger, Dr. Anwar Haque, and Dr. Dan Lizotte.

In the end I would like to thank my parents and sister for their support and having faith in me. My sister has been a great support for me throughout this journey. During my rough times she used to recharge me through her funny jokes. And last but not the least, I want to thank my Fiancée for her patience and to be there with me.

## Dedication

This thesis is dedicated to my mother Shazia Begum and my maternal grandfather the Late Haji Md. Ibrahim.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Lay Abstract</b>	<b>ii</b>
<b>Co-Authorship Statement</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>iv</b>
<b>Dedication</b>	<b>v</b>
<b>List of Figures</b>	<b>xi</b>
<b>List of Tables</b>	<b>xiii</b>
<b>List of Appendices</b>	<b>xvi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Thesis Statement . . . . .	1
1.2 Problem Statement . . . . .	3
1.3 Motivation . . . . .	4
1.4 Objectives . . . . .	6
1.5 Contributions . . . . .	6
1.6 Chapter Mapping . . . . .	8
1.7 Thesis Organization . . . . .	9
<b>2 Improving Neural Sequence Labelling using Additional Linguistic Information</b>	<b>11</b>
2.1 Introduction . . . . .	11
2.2 Related work . . . . .	13
2.3 The Model: BLSTM-CRF . . . . .	13
2.3.1 Recurrent unit: Bidirectional LSTM . . . . .	13
2.3.2 Word Sense . . . . .	14

2.3.3	Convolutional Neural Network	15
2.3.4	Conditional Random Field	15
2.3.5	Morphology: Spelling and suffix features	16
2.3.6	BLSTM-CRF model	17
2.4	Experimental Setup	20
2.4.1	Dataset Description	20
2.5	Experimental Results	20
2.6	Conclusion	24
<b>3</b>	<b>A Novel Neural Sequence Model with Multiple Attentions for Word Sense</b>	
	<b>Disambiguation</b>	<b>26</b>
3.1	Introduction	27
3.2	Related Work	28
3.3	The model	29
3.3.1	Attention on Bigrams	30
3.3.2	Attention on Words and Parts of Speech (POS)	31
3.3.3	Attention on Words, POS and Bigrams with weighting	33
3.4	Experimental Setup	35
3.5	Experimental Results	36
3.6	Conclusion and Future Work	39
<b>4</b>	<b>Improving Tree-LSTM with Tree Attention</b>	<b>41</b>
4.1	Introduction	41
4.2	Related work	43
4.3	The Model	44
4.3.1	Incompatibility of standard LSTM and Tree structured data	45
4.3.2	Tree-LSTM	45
4.3.3	Attention	48
4.4	Experimental Setup and Analysis	52
4.5	Conclusion	56
<b>5</b>	<b>Identifying Protein-Protein Interaction using Tree LSTM and Structured</b>	
	<b>Attention</b>	<b>58</b>
5.1	Introduction	58
5.2	Related Work	59
5.3	The Model	62
5.3.1	Recurrent unit: Bidirectional LSTM	62



5.3.2	Tree LSTM	62
5.3.3	Structured Attention	64
5.3.4	Combining the modules	67
5.4	Experimental Analysis and Results	68
5.5	Conclusions	72
<b>6</b>	<b>Efficient Transformer-based Sentence Encoding for Sentence Pair Modelling</b>	<b>73</b>
6.1	Introduction	73
6.2	Related Work	74
6.3	The Model	75
6.4	Experimental Setup	79
6.5	Experimental Results and Analysis	81
6.6	Conclusion	85
<b>7</b>	<b>Investigating Relational Recurrent Neural Networks with Variable Length</b>	
	<b>Memory Pointer in Sentence Pair Modelling Tasks</b>	<b>86</b>
7.1	Introduction	87
7.2	Related Work	88
7.3	The Model	89
7.3.1	Using a Fixed Length Memory Pointer	90
7.3.2	Using a Variable Length Memory Pointer	92
7.3.3	Modelling Sentence Pairs using RMC	92
7.4	Experimental Setup	94
7.5	Experimental Results and Analysis	95
7.6	Conclusion and Future Work	97
<b>8</b>	<b>You Only Need Attention to Traverse Trees</b>	<b>98</b>
8.1	Introduction	98
8.2	Proposed Model	100
8.3	Experiments	102
8.4	Conclusion	106
<b>9</b>	<b>Encoding Dependency Information inside Tree Transformer</b>	<b>107</b>
9.1	Introduction	108
9.2	Related Work	108
9.3	Model	110
9.3.1	Design Components	110

9.3.2	General Architecture	112
9.3.3	Dataset Specific Design Details	113
9.4	Datasets and Experimental Setup	113
9.5	Results and Discussion	115
9.6	Conclusion	120
<b>10</b>	<b>Modelling Sentence Pairs via Reinforcement Learning: An Actor-Critic</b>	
	<b>Approach to Learn the Irrelevant Words</b>	<b>121</b>
10.1	Introduction	122
10.2	Model	123
10.3	Datasets, Experimental Setup, Results and Analysis	131
10.4	Conclusion	137
<b>11</b>	<b>Learning to Compare Sentence Pairs at the Phrase Level: An Actor-Critic</b>	
	<b>Approach</b>	<b>138</b>
11.1	Introduction	139
11.2	Related Work	140
11.3	Model	141
11.4	Datasets, Experimental Setup, Results and Analysis	147
11.5	Conclusion	152
<b>12</b>	<b>Multilingual Semantic Textual Similarity using Multilingual Word</b>	
	<b>Representations</b>	<b>153</b>
12.1	Introduction	153
12.2	The Model	154
12.2.1	Model	155
12.2.2	Dataset Collection and Preprocessing	156
12.2.3	Negative Sample Pairs Generation	157
12.3	Experimental Setup and Analysis	159
12.3.1	Dataset Preparation	159
12.3.2	Model Parameters and Training Details	159
12.3.3	Results and Analysis	159
12.4	Conclusion	162
<b>13</b>	<b>Multilingual Corpus Creation for Multilingual Semantic Similarity Task</b>	<b>163</b>
13.1	Introduction	163
13.2	Architecture	165

13.2.1 Positive Sample Selection . . . . .	165
13.2.2 Negative Sample Selection . . . . .	167
13.3 Corpus Details . . . . .	169
13.4 Evaluation Experiments . . . . .	171
13.4.1 Model Architecture, Parameters and Training Details . . . . .	171
13.4.2 Results and Analysis . . . . .	172
13.5 Conclusion . . . . .	175
<b>14 Conclusions</b>	<b>176</b>
14.1 Key Findings . . . . .	176
14.2 Major Contributions . . . . .	177
14.3 Limitations of the Study . . . . .	178
14.4 Recommendations for Future Research . . . . .	179
14.5 A Final Word . . . . .	181
<b>Bibliography</b>	<b>182</b>
<b>A Copyright Forms of the Papers</b>	<b>209</b>
<b>B Supporting Materials</b>	<b>231</b>
B.1 Software Packages . . . . .	231
B.2 Datasets . . . . .	232
<b>Curriculum Vitae</b>	<b>234</b>

# List of Figures

2.1	BLSTM-CRF model architecture	19
3.1	Encoder-decoder architecture for sequence-to-sequence WSD with attention on bigrams	29
3.2	Encoder-decoder architecture for sequence-to-sequence WSD, taking point-wise multiplication of POS and bigram attention weights	30
3.3	Encoder-decoder architecture for sequence-to-sequence WSD with weighted combination of multiple attentions	32
3.4	The effect of different attention weights on final attention matrix	39
4.1	Illustrations of different Tree-LSTM architectures	46
4.2	Probability of each node being selected by attentive child sum Tree-LSTM Model 2 with cross sentence attention	54
4.3	Probability of each node being selected by attentive binary Tree-LSTM Model 2 with cross sentence attention	55
5.1	Standard LSTM vs Tree LSTM	61
5.2	Work flow of a child sum tree LSTM on part of a dependency tree	65
6.1	Model architecture	77
6.2	Visualization of attention weights	84
7.1	Sample encoding of a sentence using RMC with variable length memory pointer	91
7.2	Change in attention weights as the attention window shifts over time	97
8.1	Attention over the tree structure	100
8.2	Attentive tree visualization (CTT)	105
9.1	Design components	111
9.2	Encoding edge information inside dependency tree transformer	112
9.3	Attentive tree visualization (DT-Transformer + edge label)	119
10.1	Effect of RL sample counts on validation accuracy	136

10.2 Change in gradient norm with training iterations while training with and with-	
out delay	137
11.1 Sample example of extracting phrases from a sentence given a stream of 0's	
and 1's	142
11.2 Attention weights assigned by our critic	150
12.1 Topic coherence score vs number of topics	157
13.1 Topic coherence score vs number of topics	168

# List of Tables

2.1	Dataset description	21
2.2	Hyper-parameters used for the experiments	21
2.3	Comparison of F scores of different models for chunking	22
2.4	Comparison of F scores of different models for NER	22
2.5	Comparison of accuracy of different models for POS tagging	22
2.6	Training time of our BLSTM-CRF model on the WSJ corpus	23
2.7	Ablation study of our BLSTM-CRF model for POS tagging	23
2.8	Change in $w$ 's for each module with epochs.	23
3.1	Hyper-parameters used for the experiments	36
3.2	F-scores for the English all-words coarse-grained WSD on the Senseval-3 corpus	36
3.3	F-scores of our top performing models for different parts of speech with different settings of SemCor and MASC corpus	38
3.4	The change in different attention weights with epochs for Seq2Seq + conv + POS	38
4.1	Hyper-parameters used for the experiments	52
4.2	Test set results on the SICK dataset	53
5.1	Basic statistics of the corpora	68
5.2	Hyper-parameters used for the experiments	70
5.3	Results of our tLSTM model from 10-fold cross-validation against other methods	70
5.4	Cross-corpus results	71
6.1	Dataset description	80
6.2	Hyper-parameters used for the experiments	80
6.3	Performance comparison of our model on different tasks against some existing top performing models	82
6.4	Example predictions from the test set	83
7.1	Hyper-parameters used for the experiments	95

7.2 Performance of our model on different tasks compared to some top performing models . . . . .	96
8.1 Performance comparison of the <i>Tree Transformer</i> against some state-of-the-art sentence encoders . . . . .	104
8.2 Effect of positional encoding . . . . .	105
8.3 Effect of different attention modules as a composition function . . . . .	105
9.1 Performance comparison of the <i>DT-Transformer</i> + <i>edge label</i> against some state-of-the-art LSTM, Transformer and Tree-structured models . . . . .	115
9.2 Effect of initializing edge embeddings with a fixed mean and changing standard deviation . . . . .	116
9.3 Effect of different ways to form forward and backward edges . . . . .	117
9.4 Ablation study . . . . .	118
10.1 Performance comparison of our model on different tasks against some existing top performing models . . . . .	133
10.2 The values that give the best results when combining the critic with the trained actor-critic . . . . .	133
10.3 Example predictions from the test set . . . . .	134
10.4 The original average length and the average length after filtering through RL . . . . .	135
10.5 Top 50 deleted words from the test set of all three corpora . . . . .	136
11.1 Type of structure according to $a_{t-1}$ and $a_t$ . . . . .	142
11.2 Performance comparison of our model on different tasks against some existing top performing models . . . . .	149
11.3 Example predictions from the test set . . . . .	150
11.4 The values that give the best results when combining the Critic with the trained Critic + RL . . . . .	151
11.5 Comparison on artificial phrase structures . . . . .	152
12.1 Details of collected datasets . . . . .	156
12.2 Sentence pair counts for dataset partitions . . . . .	159
12.3 Example predictions from the test set . . . . .	160
12.4 10-fold cross validation performance of different models . . . . .	160
12.5 Cross corpus performance on test set . . . . .	161
12.6 Performance comparison of our model on the MSRP dataset against some existing top performing models . . . . .	161

13.1 Positive and negative sentence pair examples	165
13.2 Examples from collected multilingual corpus	170
13.3 Sentence pair counts for dataset partitions	171
13.4 10-fold cross validation performance of different models	173
13.5 Cross corpus performance	173
13.6 Performance comparison on the MSRP dataset against some existing top per-	
forming models	174
13.7 Example predictions from the test set	174
14.1 Summary of all of the model performances on the overlapping datasets	177



# List of Appendices

A	Copyright Forms of the Papers . . . . .	209
B	Supporting Materials . . . . .	231

# Chapter 1

## Introduction

### 1.1 Thesis Statement

Natural language processing (NLP) underlies modern approaches to natural language tasks like document summarization, sentence similarity computation, part of speech tagging, machine translation, named entity recognition, relationship extraction, sentiment analysis, question-answering, and topic segmentation. A characteristic of natural language is that there are many different ways to express a statement: several meanings can be contained in a single text and the same meaning can be conveyed by different texts. This text can be a sentence, a paragraph, or an entire document. Because of this characteristic of natural language, having an effective text representation is fundamental for performing these tasks at a high level. An effective text representation is one which overcomes the difficulties posed by the complexities of natural language. For example, in the identification of paraphrase between two sentences, the second sentence can use a completely different set of words to say the same thing as the first sentence. An effective representation of these sentences should be able to capture this semantic aspect. For the named entity recognition task, it is very important to understand word morphology (i.e., uppercase, alphabetic and numeric characters, etc.). For part of speech tagging, it may be essential to know the meaning of the target word as well as its context information, because different word senses may have different parts of speech. For the sentence similarity comparison and question-answering tasks, an effective sentence representation of each sentence may benefit from information drawn from the other sentences. To sum up, an effective sentence representation should encode as much knowledge from the appropriate information sources as is required for the specific natural language task to perform at a high level.

Machine learning models have shown some success in performing these natural language tasks. To push the performance bar set by these machine learning models, Manning [147] suggests that additional linguistic knowledge is needed. Some recent research works have

shown the importance of this suggestion by carefully encoding extra linguistic knowledge in their models and getting significant performance boosts [67, 134, 215, 227, 254].

In written language, the smallest units are its characters. Sets of these characters are put together to form words. Similarly, a set of words builds sentences, a set of sentences are grouped into paragraphs, and a set of paragraphs form complete documents. Each of these text units convey meaning. In this thesis we investigate text representations that capture this meaning at the character, word, and sentence levels. Following on Manning’s sage remark, we go beyond having our models analyze only plain text and utilize a number of linguistic features in the design of our deep learning models, such as character level information, morphological features (suffix and spelling), part of speech, word sense, n-gram information, dependency parse information captured as a tree, and different forms of phrase structure. In addition to incorporating these linguistic resources, we also combine them with some important design concepts. For example, in sentence comparison tasks, we hypothesize that an effective sentence representation needs to be a blending of the information contained in each sentence. So we introduce a novel cross attention mechanism that can look at two sentences at a time and generate a sentence pair representation. We adopt this design concept in a number of our works in this thesis. Another important design concept that we suggest is adopting the actor-critic reinforcement learning framework, where we design two types of actors: the first one deletes irrelevant words from a sentence pair and the second one divides a sentence pair into phrases. We design a set of models, some are capable of addressing just one task, whereas some have the potential to be useful for many tasks.

One common strategy being followed throughout the thesis is generating effective sentence representations through a standard planning process for model design and appropriate utilization of various natural language features. We explore a range of tasks, each accompanied by publicly available benchmark datasets, and try to improve upon previous performances. We adopt two learning frameworks: supervised deep learning and reinforcement learning. Deep learning models use dataset specific labels as the supervision to solve well-defined closed form problems. In contrast, most of the tasks proposed to a reinforcement learning methodology are ill-defined. For deep learning, we design our models by first deciding what information is needed, then proposing the module to introduce the information and how these modules are to be combined, then debugging these design decisions, and finally investigating the need for each module. For reinforcement learning, we explore the actor-critic framework. In this framework the actor is attempting to solve an ill-defined problem. In order to evaluate actor’s performance, the suggested solution is used by a critic to solve another well-defined problem. The actor is trained using reward and penalty from the critic and the critic uses a supervised training methodology employing the gold standard labels of the well-defined problem. Their learning

objectives are different, actor wants to maximize the reward and critic wants to minimize the loss.

## 1.2 Problem Statement

As introduced in the previous section, a text has different levels which can be represented as vectors in a distributed manner. Beginning in 2003, NLP research was focussed on generating good distributed representations of words by fighting against the curse of dimensionality [33]. Later in 2013, this research interest refocusses on a new objective, i.e., finding representations that are learnable in a reasonable amount of time (i.e., Word2Vec [156] and GloVe [183]). Some leading researchers in the NLP community focused on improving distributed word representations since they recognized that every other information structure (i.e., sentence, paragraph and document) can be broken down into this smallest building block (i.e., “word”) [26, 56, 57, 160, 237]. Distributed word representation means describing a word with a set of numbers (i.e., a vector). And one can easily accumulate these word vectors into either sentence or paragraph level representations through some simple amalgamation of information (i.e., addition or averaging). But this focus soon shifted as the community realized that a higher level representation (i.e., the sentence level) is needed to generate better text representations. Recurrent Neural Networks (RNNs) [194] along with its variants, Long Short Term Memory (LSTM) [53, 92] and Gated Recurrent Units (GRUs), are found to work very well with text data as they can capture long distance dependencies. These RNN variants are widely used in generating good sentence representations where sentences are analyzed sequentially from both left to right and right to left as they are found to capture more in-depth features. Some of the models also looked at the syntax trees of sentences to capture even more complex features. Because of some builtin state dependency and parallelism issues, this trend then shifted towards a recent architecture called “Transformer” where those aforementioned problems are addressed with a newly introduced method named “Attention”. Recently, both RNN and Transformer-based [240] architectures have been looked at to address problems in the multilingual domain [51, 244, 245]. This language independent learning is interesting in a sense that we can learn the rich language features from a highly resourced language and apply that knowledge to get insight about a poorly resourced language. Reinforcement learning (RL), on the other hand, follows a paradigm that is completely different to deep learning. Few works have been done in NLP with RL because it is very hard to design a reward-based system following a continuous action as text is made of discrete elements. Following Manning’s [147] suggestions, we want to enrich these well established machine learning models with the information inherent in a variety of natural language features. We hypothesize that

having additional linguistic knowledge can bring another level of expertise into these modern artificial intelligence tools.

In this thesis, our goal is to obtain effective representations of sentences in a vector space by capturing both semantics and syntax. To capture the semantics we take advantage of good pre-trained word vectors. And to incorporate sentence syntax, we look at the ordering of the words in a plain sentence as well as the parent-child relationship in the tree structures used to represent syntactic relationships. Our novel models designed to generate effective sentence representations utilize both of these feature spaces.

The models previously mentioned are being used to solve many NLP tasks. Most of the recent works on computing sentence representations evaluate their model on the natural language inference (i.e., textual entailment) task (NLI). Understanding textual entailment becomes crucial to understanding natural language as it constitutes an effective way to evaluate machine learning algorithms. It is an interesting problem as it reflects our ability to extract and represent the meaning at the sentence level, and it takes into account several characteristics of natural language such as scope, syntactic structure, lexical ambiguity, and semantic compositionality. In addition to NLI, there are some other tasks that deal with sentence pairs: paraphrase identification and question-answer pairing. Recently, some research works are interested in the multilingual version of the aforementioned problems which we also explore in this thesis. We revisit a complex NLP task from the biomedical domain called the protein-protein interaction problem. The task is: in a sentence describing a set of proteins and their interactions, identify the protein entities that interact and those that don't. In addition to these, there are some other classic NLP problems that we address in this thesis: part of speech tagging, named entity recognition, chunking, sentiment classification, and word sense disambiguation.

## 1.3 Motivation

This thesis has been motivated by an interest in natural language processing using machine learning methodologies. In particular, the investigation of distributed representations of the previously mentioned language units is our primary objective. Recent interest in a deeper understanding of how modern machine learning models are making their decisions has also directed us toward exploring this aspect of the models that we design. In carrying out these investigations, we wish to make our contributions as general as possible for a wide range of natural language tasks. We will now look at each of these motivations in some more detail.

Natural language can be ambiguous which leads to difficulty in interpretation. An idea can be expressed in different statements and sometimes the same statement conveys different meanings when used in different contexts. For example, the sentences “*Those who only had*

*surgery lived an average of 46 months.” and “For those who got surgery alone, median survival was 41 months.”* are saying more or less the same thing even though they are using different sets of words. Another example is “*Visiting relatives can cause problems.*”. This sentence can be communicating that “*Relatives who visit us can cause problems.*” or “*Going to visit relatives can cause problems.*”. The syntactic structure of the sentence is ambiguous, leading to the different semantics. The two syntactic structures are dependent on the correct part of speech (i.e., present participle modifying “relatives” or present tense finite verb) given to the word “Visiting” which can be determined only if the sentence’s context is known. Capturing meaning is most important when working in the text domain. It has been shown that word embeddings can capture an effective semantic meaning of a word by looking at the words that occur together in the same context. But sentences are different. There is no fixed vocabulary for sentences, so how these text objects are represented depends on how we appropriately utilize the word embeddings to build up the sentence representation. We are motivated in bridging this gap between effective semantic representation of words and sentences.

A recent concern of the research community is the lack of interpretability in deep learning or machine learning models. Typical questions asked when researchers are attempting to understand a machine learning model are: why a model is performing well, why this complex architecture works, why not a simpler model, etc. These questions are not appropriate when trying to interpret the decisions made by the machine learning models. We are motivated in designing interpretable deep learning models by giving a proper justification of why this architecture works providing in depth design principles. If something works, we want to give an explanation why it works, and if something does not work, we want to properly explain why it is not working and what should we do to make it work. We use the attention mechanism as a form of explaining a model’s behavior and decision making strategy. In much of this thesis, we debug a given model based on how it is putting attention on different parts of the input. Our post hoc subjective analysis of this attentive focus also provides a possible interpretation of what portion of the text is influencing the model’s decision.

And finally we want to widen the scope of our exploration. We do not want to keep ourselves limited to solving just one NLP problem, instead we explore and study a set of problems, revisit their solutions, and provide our own way of solving the same problems with an objective to get better performance. This motivates us to go beyond the deep learning domain and to explore the reinforcement learning domain. We are also motivated to do a knowledge sharing type of learning. In this scenario, the dataset can be labelled for one task (e.g., sentiment classification) and we can use the gold label (i.e., the sentiment) as a marker to solve a completely different problem (e.g., sentence compression).

## 1.4 Objectives

In order to deal with any NLP problem, it is essential to have an effective representation of the entity of study. This entity can be the word, the sentence, or the entire document. Modern machine learning and deep learning models can extract abstract features from the data thanks to its providing a methodology for users to create powerful designs which use well-defined differentiable objective functions that map inputs to outputs. However, we are interested in pushing the state-of-the-art on some complex NLP tasks utilizing a combination of machine learning models and linguistic knowledge. More specifically, we are interested in creating task-specific sentence representations by designing machine learning models that take into account both pre-trained word embeddings as well as inherent linguistic features hidden beneath the text. As the thesis progresses, the reader will understand that we always keep the following objectives in mind.

**To show the effect of natural language syntax trees both as an external knowledge source as well as a primary feature space.** We design a set of deep learning models that work only with tree-structured data and need recursive traversal. We also use this tree-structured syntactic information as an external knowledge source in aiding a few tasks.

**To compare and justify the decision taking strategy of a machine learning model.** We are interested in justifying some pre-processing steps that computational linguistic researchers always do when preparing text for some machine learning models. This will allow us to see to what extent an algorithm can mimic human language ability.

**To see the impact of additional linguistic features in a machine learning model.** Other than the plain textual features, we want to verify whether having additional linguistic features from an external knowledge source with a different design strategy helps a machine learning model.

**To push the state of the art results on some NLP tasks.** We design a range of machine learning models that can solve a set of NLP tasks. We are interested in designing models with explainable architecture and at the same time can provide state-of-the-art performance.

**To provide explainability for the decisions made by the machine learning models.** Across all the tasks, one thing is common: we provide a skeleton of how the model is taking a decision on a particular sample. We always try to give some sort of explainability about the thought process of a model.

## 1.5 Contributions

The thesis provides a number of important contributions. As we explore a large area of the NLP domain, our contributions may seem somewhat diverse. But here, we provide the threads

that pull them into coherent themes. This discussion also gives a short synopsis of how the contribution is accomplished and what it provides. Details are forthcoming in later chapters.

First, we provide some insight into which linguistic features can aid task specific NLP models. This external knowledge source realizes a qualitative change in the model design either as a module or by providing the foundation of the design. Quantitatively, the empirical results confirm that these additional features definitely help to give the model relevant linguistic information when analyzing the text piece. Having these linguistic features contributes to faster convergence of our models, as well.

The employment of attention is a strong theme in the thesis. We provide a way of making deep learning models more explainable through the use of attention. This qualitative improvement allows us to view what part of the text piece a model is focussing on when making a decision. It also helps debugging of the model architecture and assists identifying the contribution of specific modules.

We make the attention module compatible with the tree structured models. To the best of our knowledge, no prior work encodes attention inside a tree structured LSTM cell. By encoding attention inside the tree structured LSTM cell, we show which child contributes more in computing the representation of the parent node. We also show how different design strategies affect the amount of contribution by different children.

We successfully encode natural language grammar trees solely using attention. We provide a way of interpreting natural language grammar trees by explaining facts such as which nodes are more important, which branch to look at, and which path to traverse. We also justify both the correct and incorrect decision provided by the designed model.

We provide a way to generate latent trees without needing any supervision. Typically, one of a set of common natural language parsers is used to generate the parse trees which are then encoded by a machine learning model. Through our proposed model we show that the automatically generated task-dependent trees work as well as the fixed linguistically generated trees. This qualitative change means that end-to-end task learning can be performed without the need of an external knowledge source.

We also justify a major pre-processing step (i.e., task specific stop word removal) of any NLP task through the use of reinforcement learning. We further show that other than the stop words, there are some content words as well that are not needed when performing certain tasks. Our model successfully identifies them as well.

We create a multilingual semantic textual similarity (STS) corpus focused on enterprise business content. In this multilingual STS setting, we provide a semi-supervised algorithm to generate negative examples in the presence of only positive samples. Our generalized algorithm can work on any business text given a few basic constraints.



Last but not the least, our models have made significant quantitative strides. Through extensive experimentation, we show the competitiveness of each of our models against the currently available state-of-the-art models across various tasks. We push the state-of-the-art performance boundary on a range of NLP tasks, i.e., paraphrase identification, natural language inference, protein-protein interaction, part of speech tagging, and question answering.

## 1.6 Chapter Mapping

The chapters in this thesis are organized to provide the reader with a clear view of the evolution of our ideas. At the beginning of our research we were mainly focused on adding linguistic information as an external knowledge source. This idea is reflected in Chapters 2 and 3. Then, we decided to design machine learning models that are compatible with handling specific types of natural language syntactic structure, i.e., dependency and constituency trees. We incorporated an attention module inside a tree-structured long short term memory module and later on made a significant performance improvement to a complex NLP problem in the biomedical domain, i.e., protein-protein interaction. These are thoroughly explained in Chapters 4 and 5. Further, in order to show the effectiveness of the attention module, we design a model that is capable of looking at a pair of sentences and providing a decision about their relatedness. To accomplish this decision making, we suggest adding a module which provides a completely different point of view when comparing a pair of sentences. We also enhance the memory mechanism of the standard long short term memory module through a variable length memory pointer mechanism utilizing the attention mechanism. The former idea is depicted in Chapter 6 and the latter one is discussed in Chapter 7. Later, we became interested in using this attention module as a composition function to encode natural language syntax trees. We also extend this work to encode dependency arc information. Chapters 8 and 9 explain this architecture. We also wanted to explore applying the reinforcement learning paradigm to NLP problems. Since stop word removal is customary for various NLP tasks, we investigate whether removing them manually or through a machine learning model makes any difference. We have chosen reinforcement learning to do this because there is no dedicated dataset on which we can design a specialized supervised machine learning model. Supervised learning requires an exact mapping from input to output where the agent receives a feedback as correct set of actions. On the other hand with reinforcement learning, the positive or negative behavior of an agent is defined by the reward and penalty signals. We also show the power of the reinforcement learning-based framework by having it provide artificial phrase structuring without needing any natural language grammar tools. We use the ground truth label from the datasets specific to the semantic similarity task in reward and penalty computation to solve the word removal and phrase generation tasks as

mentioned above. Chapters 10 and 11 explain these ideas. Finally, we explore the effectiveness of multilingual word representations on an industry-related problem and then address a benchmark sentence pair matching task using the English sentence representations that were learned in this multilingual space. This problem resulted from a Mitacs Accelerate internship. To carry out this work, we also create our in-house multilingual sentence pair comparison dataset utilizing a mixture of a traditional model (i.e., Latent Dirichlet Allocation) and a modern large scale language model (i.e., OpenAI-GPT). Chapters 12 and 13 explain these projects.

## 1.7 Thesis Organization

The thesis is organized as follows:

**Chapter 1** provides an introduction to the thesis. The chapter outlines the problem, motivation for doing this research, thesis objectives, and contributions to the research community.

**Chapter 2** provides the effect of external linguistic information in a deep learning model and shows different ways of adding it in addressing a range of tasks like part of speech tagging, named entity recognition, and chunking. It involves extracting different syntactic features through careful investigation as well as some semantic features through an external algorithm.

**Chapter 3** continues the same hypothesis of checking the effect of linguistic features except on a different problem, word sense disambiguation. It follows a well-known deep learning paradigm, i.e., sequence to sequence learning.

In **Chapter 4** we move our focus towards tree-structured deep learning models. Rather than using natural language syntax trees as an external knowledge source, our motivation was to somehow encode an entire tree as a representation and solve some specific tasks. We also utilize an attention module in our design strategy. This chapter explores just the semantic relatedness task.

In **Chapter 5** we solve a complex NLP problem titled protein-protein interaction using the tree-structured models that we explore in the previous chapter.

In **Chapter 6** we wanted to explore further the effect of the attention module. So, we design a sentence pair model based solely on attention and extend our task domain to paraphrase identification and question answer pairing.

In **Chapter 7** we again follow the same principle as we did in Chapter 4 except we use the attention module to enhance the memory mechanism of a long short term memory cell.

After doing a thorough analysis of the attention module in both sequential as well as tree-structured recurrent neural networks, in **Chapter 8** we shift our focus towards designing a tree structured Transformer module using various attention mechanisms as the composition function. We limit ourselves to the same set of problems that we investigate in the previous

chapters.

**Chapter 9** is an extension of Chapter 8 where we encode labels from dependency parsing along with the tree structure.

In **Chapter 10** we shift our focus to reinforcement learning as a means to distinguish the important portions of a text from the unimportant. In contrast to the attention module which provides a probability distribution over the input, this chapter initiates the idea of achieving a 1/0 binary distribution with the meaning whether to keep an input piece or delete it.

In **Chapter 11** we continue with the reinforcement learning framework with the same objective of getting a distribution of 1s and 0s. However, the motivation is different. We explain how we use this distribution to suggest an artificial phrase structure.

**Chapter 12** explains our idea of semantic similarity across multiple languages. We explain how we utilize multilingually aligned word embeddings to solve this problem.

**Chapter 13** explains the procedure to create a corpus for the multilingual semantic similarity task that we introduce in the previous chapter.

Finally, **Chapter 14** briefly summarizes the thesis and explores its limitations and possible avenues for extension.

## Chapter 2

# Improving Neural Sequence Labelling using Additional Linguistic Information

This chapter is based on the paper titled “Improving Neural Sequence Labelling using Additional Linguistic Information” co-authored with Muhammad Rifayat Samee and Robert E. Mercer that appeared in the 17th IEEE International Conference on Machine Learning and Applications (ICMLA 2018) [11].

Sequence labelling is the task of assigning categorical labels to a data sequence. In Natural Language Processing, sequence labelling can be applied to various fundamental problems, such as Part of Speech (POS) tagging, Named Entity Recognition (NER), and Chunking. In this study, we propose a method to add various linguistic features to the neural sequence framework to improve sequence labelling. Besides word level knowledge, sense embeddings are added to provide semantic information. Additionally, selective readings of character embeddings are added to capture contextual as well as morphological features for each word in a sentence. Compared to previous methods, these added linguistic features allow us to design a more concise model and perform more efficient training. Our proposed architecture achieves state-of-the-art results on the benchmark datasets of POS, NER, and chunking. Moreover, the convergence rate of our model is significantly better than the previous state-of-the-art models.

### 2.1 Introduction

Linguistic sequence labelling is one of the first tasks focusing on natural language processing using deep learning and it has been well examined over the past decade [56, 133, 267]. Part of speech (POS) tagging, named entity recognition (NER), and chunking are subclasses of sequence labelling. They play a vital role in fulfilling many downstream applications, such as

relation extraction, syntactic parsing, and entity linking [19, 190, 250]. POS tagging assigns a tag to each word in a text, where a tag represents the lexical category of a word. NER is a sub-task of information extraction that seeks to locate and classify named entities in text. Chunking identifies the POS and short phrases in a sentence by doing shallow parsing and also groups words into syntactically correlated phrases. These labelled texts can later be used for different applications such as machine translation, information retrieval, word sense disambiguation, and natural language understanding etc.

Before neural sequence models, algorithms were based on Hidden Markov Models (HMMs) [75, 269] and Conditional Random Fields (CRFs) [121, 168]. The problem with these models is they are dependent on manually hand-crafted features, so it becomes difficult to apply them in real life applications.

To overcome these drawbacks, Neural Network (NN) based models have been proposed in which the models are responsible for extracting higher level features from the data [31, 107]. Recurrent Neural Networks (RNNs) along with its variants, Long Short Term Memory (LSTM) [53, 92] and Gated Recurrent Units (GRUs) are found to work very well with sequence data as they can capture long distance dependencies [53, 83, 224]. Nevertheless, considering the overwhelming number of their parameters and the relatively small size of most human annotated sequence labelling corpora, annotations alone may not be sufficient to train complicated models. So, guiding the learning process with extra knowledge could be a wise choice [147, 200]. For example, before tagging the word ‘*flies*’ as either a verb or a noun in the sentence ‘*Time flies like an arrow*’, having its semantic meaning would make a correct tagging straightforward.

Knowing the sense of a word prior to sequence labelling (POS or NER) often gives the best tag for that word. Word senses can be obtained from a variety of sources: WordNet [157], a lexical database that can be queried for the sense of a word given its context; the simplified LESK algorithm [25, 29] which uses the dictionary definition of each word in a sentence as extra context to suggest the word sense; and linear algebraic methods [20] which uses a random walk on a discourse model and represents the vector of the base word as a linear combination of its probable sense vectors.

In this chapter, we propose a novel deep neural architecture for doing sequence labelling incorporating not only semantic features through word senses but also the rich morphology of the words. We provide an in depth analysis of the design of this architecture giving some insights regarding how each feature is introduced into the architecture. Our sequence model achieves state of the art results on the three sequence labelling tasks, POS, NER, and chunking, and has a training time at least four times faster than the currently available state of the art models.

## 2.2 Related work

Huang et al. [96] propose a few models for the sequence tagging task. Apart from just word embeddings, they use morphological, bigram and trigram information as their input features. Later, they use LSTM and Bidirectional LSTM (BLSTM) with CRF to do the final tagging. Lample et al. [123] extract character embeddings from both the left and right directions, concatenate these with word embeddings and use a stacked LSTM with CRF to do the tagging. Liu et al. [138] propose a model leveraging word and character level features. It includes a language model to represent the character level knowledge along with a highway layer to avoid the feature collision. Finally it is trained jointly as a multitask learning.

Yu et al. [255] propose a general purpose tagger using a convolutional neural network (CNN). First, they use CNNs to extract the character level features and then concatenate it with word embeddings, position embeddings and binary features. Finally they use another CNN to get the contextual features as well as to do the tagging. Ma and Hovy [145] propose an end-to-end sequence labelling model using a combination of BLSTM, CNN and CRF. They use a CNN to get the character level information, concatenate it with word embeddings and then apply BLSTM to model the contextual information. Finally they generate the tags by using a sequential CRF layer. Rei [192] trains a language model type objective function using BLSTM-CRF to predict the surrounding words for every word in the corpus and utilizes it for sequence labelling.

The contribution of this chapter combines the common themes found in these previous works (morphology encoded as character embeddings, and word embeddings) with word senses in a new architecture that integrates these embeddings and the outputs of a CNN, a BLSTM, and a CRF in novel ways.

## 2.3 The Model: BLSTM-CRF

In this section, we describe our work in detail. We first explain each of the pieces of the complete architecture and then we explain how we combine those pieces to build our model. This section also explains the morphological and semantic features that we have added with our model to get the improved performance that is discussed in Section 2.5.

### 2.3.1 Recurrent unit: Bidirectional LSTM

Recurrent neural networks (RNNs) are the best known and most widely used NN model for sequence data as they go over the entire sequence through time and try to remember it in a

compressed form. Although its variant, LSTM, is very good with long term dependencies, for many sequence labelling task, it is important to keep track of these dependencies from the future as well as from the past. But LSTM has just one hidden state from the past and changes that hidden state recursively through time. An elegant solution to this problem is going over the sequence in both forward and backward directions with two hidden states and finally concatenating the output from both directions. This bidirectionality has proven to be very effective in some prior works [73, 84, 229]. The resulting network, the Bidirectional LSTM (BLSTM), is the RNN variant that is used by the model described below.

### 2.3.2 Word Sense

Knowing the sense of a word prior to tagging makes the tagging task more straightforward. Generally, polysemy is captured in standard word vectors, but the senses are not represented as multiple vectors. So we have trained an adaptive skip gram model, AdaGram, [28] which gives a vector for each sense of a word. It is a non-parametric Bayesian extension of the skip-gram model and is based on the constructive definition of Dirichlet process (DP) [76]. It can learn the required number of representations of a word automatically.

In our model, we denote a set of input words as  $X = \{x_i\}_{i=1}^N$  and their context as  $Y = \{y_i\}_{i=1}^N$ . The  $i$ th training pair  $(x_i, y_i)$  consists of words  $x_i = o_i$  with context  $y_i = (o_t)_{t \in c(i)}$ , where  $c(i)$  is the index of the context words. Then, instead of maximizing the probability of generating a word given its contexts [156], we maximize the probability of generating the context given its corresponding input words [28]. Our final objective function becomes,

$$p(Y|X, \theta) = \prod_{i=1}^N p(y_i|x_i, \theta) = \prod_{i=1}^N \prod_{j=1}^C p(y_{ij}|x_i, \theta) \quad (2.1)$$

where,  $\theta$  is the set of model parameters. The drawbacks of this objective function is that it captures just one representation of a word which goes against a word having different senses depending on the context [20]. To counter this, AdaGram introduces a new latent variable  $z$  which captures the required number of senses even though the number of structure components of the data is unknown a priori. In AdaGram, if the similarities of a word vector with all its existing sense vectors are below a certain threshold, a new sense is assigned to that word with a prior probability  $p$ . The prior probability of the  $k$ th meaning of word  $w$  is

$$p(z = k|w, \beta) = \beta_{wk} \prod_{r=1}^{k-1} (1 - \beta_{wr}), p(\beta_{wk}|\alpha) = \text{Beta}(\beta_{wk}1, \alpha), k = 1 \dots \quad (2.2)$$

where  $\beta$  is a latent variable and  $\alpha$  controls the number of senses. Theoretically, it is possible

to have an infinite number of senses for each word  $w$ . However, as long as we have a finite amount of data, the number of senses can not be more than the number of occurrences of that word. With more data, it can increase the complexity of the latent variables thereby allowing more distinctive meanings to be captured. Taking all the facts into account, our final objective function becomes,

$$p(Y, Z, \beta | X, \alpha, \theta) = \prod_{w=1}^V \prod_{k=1}^{\infty} p(\beta_{wk} | \alpha) \prod_{i=1}^N [p(z_i | x_i, \beta)] \prod_{j=1}^C [p(y_{ij} | z_i, x_i, \theta)] \quad (2.3)$$

where  $Z = \{z_i\}_{i=1}^N$  is a set of senses for all the words.

### 2.3.3 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are good for extracting n-gram features from a sentence [130]. They consist of kernels (i.e., a weight matrix) which are used to go over the input word embedding matrix with a variable stride length and extract some higher level features. In our model, we use a CNN to get the bigram features. First we pad the input sentence<sup>1</sup> and pass it to an embedding layer. This layer represents the sentence as a matrix of size  $(m + 1) \times d$ , where  $m$  is the actual sentence length and  $d$  is the embedding dimension. Next, we initialize a kernel of size  $2 \times d$  with stride length 1 and convolve it with the input sentence matrix. This results in a bigram embedding matrix  $B$  of size  $m \times d$  using Eqn. 2.4.

$$B_{i,:} = \sum_{j=1}^2 I_{i+j,:} * K_{j,:} \quad (2.4)$$

where  $I$  is the input sentence matrix,  $K$  is the convolution kernel and  $n$  is the maximum sequence length for the current batch. Later, this bigram embedding is passed to a BLSTM layer to extract more abstract features.

### 2.3.4 Conditional Random Field

Each of the tasks that we are modelling requires a tag to be assigned to each word. In addition to using the current word to predict its tag, it is also possible to use the information about the neighboring words' tags. There are two main ways to do this. One way is to calculate the distribution of tags over each time step and then use a beam search-like algorithm, such as maximum entropy markov models [153] and maximum entropy classifiers [191], to find the

<sup>1</sup>To keep the size of the feature matrix uniform through the model, we padded a start token <start> at the beginning of the input sentence.



optimal sequence. Another way is to focus on the entire sentence rather than just the specific positions which leads to Conditional Random Fields (CRFs) [121]. CRFs have proven to give a higher tagging accuracy in cases where there are dependencies between the labels. Like the bidirectionality of BLSTM networks a CRF can provide tagging information by looking at its input features bidirectionally.

In our model we denote a generic input sequence as  $x = \{x_i\}_{i=1}^N$ , generic tag sequence as  $y = \{y_i\}_{i=1}^N$ , and set of possible tag sequences of  $x$  as  $F(x)$ . Then we use CRF to calculate the conditional probability over all possible tag sequences  $y$  given  $x$  as

$$p(y|x; W, b) = \frac{\prod_{i=1}^n \phi_i(y_{i-1}, y_i; x)}{\sum_{y' \in F(x)} \prod_{i=1}^n \phi_i(y'_{i-1}, y'_i; x)} \quad (2.5)$$

where  $\phi(\cdot)$  is the score function for the transition between the tag pair  $(y', y)$  given  $x$ . We train this CRF model using maximum likelihood estimation (MLE) [102]. For a training pair  $(x_i, y_i)$  we maximize

$$L(W, b) = \sum_i \log p(y|x; W, b) \quad (2.6)$$

where  $W$  is the weight matrix and  $b$  is the bias term. While decoding, we search for the best tag  $\hat{y}$  with the highest conditional probability using the Viterbi algorithm [201].

$$\hat{y} = \arg \max_{y \in F(x)} p(y|x; W, b) \quad (2.7)$$

### 2.3.5 Morphology: Spelling and suffix features

For the morphological features, we have focused on spelling and suffix features. We extract 14 spelling features for a given word and store it as a binary vector  $S V_{1 \times 14}$ :

- Composed only of alphabets or not
- Contains non-alphabetic characters except ‘.’ or not
- Starts with a capital letter or not
- Composed only of upper case letters or not
- Composed only of lower case letters or not
- Composed only of digits or not
- Composed of alphabets and numbers or not
- The starting word in the sentence or not
- The last word in the sentence or not
- In the middle of the sentence or not
- Ends with an apostrophe s (’s) or not

- Has punctuation or not
- The sentence starts with a capital letter or not
- Composed mostly of digits or not

Apart from extracting these features, we also replace all the numbers in the corpus with the `<number>tag`.

We have assembled a list of 137 suffixes from <https://www.learnthat.org/pages/view/suffix.html> and have used the ten that occur most often in our corpus for this study. Then for each of these suffixes, we have collected the words that end with that suffix and have recorded their POSs as well as the frequency. Next, we made an assumption that if a word  $w$  with POS  $x$  ends with a specific suffix  $s$  exceeds a frequency threshold in the training set, then  $s$  is the true suffix of word  $w$ . We record the pair as  $(w, s)$ . Finally, we create a one hot vector  $SUV_{1 \times 10}$  for each word where a 1 at index  $k$  means the word has the  $k$ th suffix.

### 2.3.6 BLSTM-CRF model

In this sub-section, we combine the BLSTM and CRF models with some feature connection techniques to form our final BLSTM-CRF model. We divided this final model into some modules and the description of each of these modules is as follows:

**Module 1: Word Level Features** This module starts with an embedding layer. In detail, we initialize the embedding layer randomly as well as using pre-trained embeddings (GloVe / word2vec). Next we represent each sentence as a column vector  $I_{m \times 1}$  where each element of the vector is a unique index of the corresponding word. Then we pass this vector to an embedding layer which gives a matrix representation  $W_{m \times d}$ . Here,  $d$  is the embedding dimension.

**Module 2: Character Embedding** In this module, first we split a word into its characters and then transform it into a column vector  $C_{k \times 1}$ , where  $k$  is the word length and each element of the vector is a unique index of the corresponding character. Next we initialize an embedding layer randomly and pass the character vector into it. This will change the representation to a matrix of size  $k \times n$  where  $n$  represents the embedding dimension. Then we use an LSTM on this matrix and store the last hidden state of this LSTM as the character level representation  $C_{1 \times n}$  of the word. Finally, for a sentence with  $m$  words, it is stored as a matrix  $C_{m \times n}$ .

**Module 3: Selective Pickup from Char LSTM (SP-CLSTM)** In this module, we introduce a new way of capturing the morphological features as well as the context features. The word embeddings from module 1 gives the contextual features in both directions and the character embeddings from module 2 gives the lexical information. We capture both sets of information by first representing each sentence in terms of its characters  $I_{(k \times m) \times 1}$  and then turn this into a matrix of size  $k \times m \times d$  through a random embedding layer. Then we apply a

BLSTM over this representation and finally we pick those indices from the output where each word ends. This selective pickup provides the morphological information of a word as well as information about the previous words in the sequence.

$$\tilde{C}_{m \times d} = \text{SELECT}(\text{BLSTM}(I_{(k \times m) \times d})) \quad (2.8)$$

**Module 4: Sense Features** This module calculates the sense level contextual features of a sentence. First, we initialize a sense embedding layer using the pre-trained sense embeddings from AdaGram. Then we tag each word in the input sentence using the module `disambiguate` from AdaGram (the word ‘apple’ with sense 2 is tagged as ‘apple\_2’). This modified input sentence is then passed to the embedding layer initialized before and finally the resultant output is passed to a BLSTM layer. The output of this BLSTM layer gives the sense level contextual feature  $S_{m \times d}$ .

**Module 5: Bigram Features** This module calculates the bigram embedding features  $B_{m \times d}$  of a sentence as described in the Subsection [2.3.3](#)

**Module 6: The Connection Technique** In this module, we combine all the features and the modules using some novel connection techniques and build our final BLSTM-CRF model as shown in Figure [2.1](#). First we concatenate the word embedding from module 1 with the character embedding from module 2 and the suffix vector from Subsection [2.3.5](#) as  $[W_{m \times d}, C_{m \times n}, SUV_{m \times 10}]$ . Following this, we apply a BLSTM on this new embedding matrix, calling this output  $O^1_{m \times d}$ . The outputs of modules 3, 4 and 5 are called  $O^2_{m \times d}$ ,  $O^3_{m \times d}$ , and  $O^4_{m \times d}$ , respectively. Then we initialize four scalar weights  $w_1$ ,  $w_2$ ,  $w_3$  and  $w_4$  with initial value 1.0 and add them as model parameters. We form a linear combination of the  $w_i$  weighted  $O^i$ s to form the final output.

$$O = \sum_{i=1}^4 O^i w_i \quad (2.9)$$

The final output ( $O_{m \times d}$ ) have pieces of information from all the features that we calculated above. We choose linear addition rather than concatenation of these output features, because concatenation will result in a very large feature matrix and the network have to tune each of the cell of this matrix during back-propagation. Following this, we initialize an LSTM layer where we pass the final output from Eqn. [2.9](#) at each time step  $\tilde{O}^i_{1 \times z} = \text{LSTM}(O^i_{1 \times d}, h^{i-1}_{1 \times d})$  and store the outputs separately  $\tilde{O}_{m \times z} = [\tilde{O}^1, \tilde{O}^2, \dots, \tilde{O}^m]$ . This LSTM layer unfolds at each time step taking the hidden state of the previous time step to initialize the hidden state of the current time step. The previous hidden state has the information about the previous tag and initializing the current hidden state with the previous one explicitly gives this information. Next we pass the output from each time step to a  $\tanh$  layer  $T_{1 \times d} = \tanh(\tilde{O}^i)$ , which squeezes the values

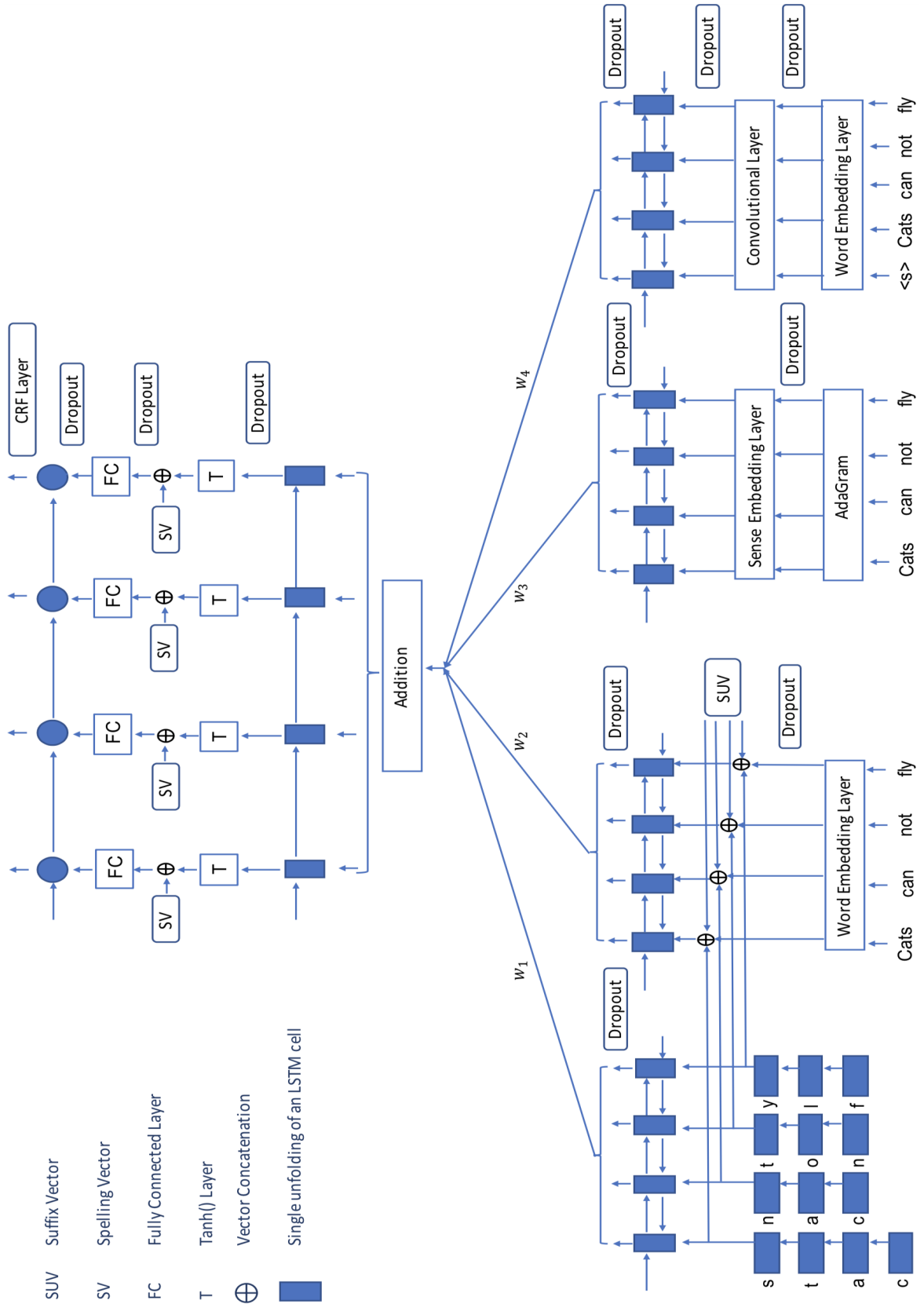


Figure 2.1: BLSTM-CRF model architecture

between  $[-1, 1]$ . Then we concatenate this  $\tanh$  output  $T_{1 \times d}$  with the spelling features  $SF_{1 \times 14}$  calculated in subsection 2.3.5 and pass this to a fully connected (FC) layer. This FC layer maps the output to the number of tag classes  $Y_{1 \times c} = \text{FC}([T_{1 \times d}, SF_{1 \times 14}])$ , where  $c$  represents number of classes. We do this for each time step and concatenate the results to make a final tensor  $Y_{m \times c}$ . Finally, we pass this tensor to the CRF layer and calculate the possible tag sequence for the given input sequence.

## 2.4 Experimental Setup

In this section, we describe the detailed experimental setup for the evaluation of our study. We first explain the dataset statistics for each tagging task. Following this, we explain the working environment details along with the hyper-parameter settings of our architecture.

### 2.4.1 Dataset Description

We test our BLSTM-CRF model on three NLP tagging tasks: Penn TreeBank (PTB) POS tagging [149], CoNLL 2000 chunking [232], and CoNLL 2003 NER [233]. Table 2.1 shows the number of sentences in the training, validation and test sets respectively for each corpus. We utilize the BIOESx explanation standard for the chunking and NER tasks.

Table 2.2 shows the detailed hyper-parameter settings of our model and some of the hyper-parameters for AdaGram (the remaining parameters are set to their default values [27]). We train our model on Nvidia GeForce GTX 1080 GPU with both the ‘Adam’ and ‘SGD’ optimizers. All of the results in the next section are reported using ‘SGD’ as it was giving the best results. The ‘Learning rate decay’ parameter was only used with the ‘SGD’ optimizer. We used PyTorch 0.3.1 to implement our model and Julia 0.4.5 for running AdaGram under the Linux environment.

## 2.5 Experimental Results

In this section, we describe in detail the results obtained with our proposed architecture. As the evaluation metrics, we use accuracy for the WSJ corpus and F score (micro averaged) for the CoNLL00 and CoNLL03 tasks. This section also contains the results of the top performing models for all three sequence labelling tasks. Additionally, we show the rate of convergence of our model compared to the state of the art one. Finally, we conclude this section by giving an ablation study by removing certain modules as well as features and mixing them in different combinations.

	<b>WSJ</b>	<b>CoNLL00</b>	<b>CoNLL03</b>
Train	39831	8936	14987
Valid	1699	N/A	3466
Test	2415	2012	3684

Table 2.1: Dataset description

<b>BLSTM-CRF</b>	
<b>Hyper-parameter</b>	<b>Range Selected</b>
Learning rate	<b>0.001</b> / 0.015 / 0.01
Batch size	<b>10</b> / 50 / 100
No. of LSTM layers	1 / <b>2</b> / 3
Momentum	0.9
Dropout	0.5 / <b>0.2</b> / 0.1
Word embedding size	<b>300</b> / 200 / 100
Character embedding size	<b>50</b> / 30
Initial scalar weight value	1.0
Gradient clipping	<b>5</b> / 20 / 50
Weight decay	$10^{-5}$
Learning rate decay	0.05
CNN kernel size	$2 \times (\mathbf{300}/200/100)$
<b>AdaGram</b>	
Epoch	1000
Window size	<b>5</b> / 7 / 10
No. of prototypes	5
Sense embedding size	300
Prior prob. of new sense	0.1
Initial weight on first sense	-1
Word embedding size	<b>300</b> / 200 / 100

Table 2.2: Hyper-parameters used for the experiments (in boldface) and the ranges that were searched during tuning.

Table 2.3 shows the performances of all of the chunking systems. An SVM based classifier [119] won the CoNLL 2000 challenge with an F score of 93.48%. However, later they improved their result up to 93.91% [118]. Recently, most of the models incorporate CRF in their architecture to capture the tag dependencies and achieve very good performance [154][201][223]. However, none of them surpass the performance of [203] which uses an HMM to capture the dependencies and a voting scheme to increase the confidence interval of the model. Our model outperforms all the existing models and achieves a state of the art F score of **96.76%**.

Table 2.4 shows the results of the existing models on the NER task. Huang et al. [96] did many experiments using random and pre-trained embeddings on their model. For random embeddings, they achieved a very low score of 84.26%. However, when they use pre-trained

Model	F-score
SVM classifier [119]	93.48
SVM classifier [118]	93.91
BI-LSTM-CRF [96]	94.13
Second order CRF [154]	94.29
Second order CRF [201]	94.30
Conv. network tagger [57]	94.32
Second order CRF [223]	94.34
BLSTM-CRF (Senna) [96]	94.46
HMM + voting [203]	95.23
BLSTM-CRF (Ours)	<b>96.76</b>

Table 2.3: Comparison of F scores of different models for chunking

Model	F-score
Conv-CRF [57]	81.47
BLSTM-CRF [96]	84.26
MaxEnt classifier [52]	88.31
HMM + Maxent [77]	88.76
Semi-supervised [18]	89.31
Conv-CRF + Senna [57]	89.59
BLSTM-CRF [96]	90.10
CRF + LIE [178]	90.90
BLSTM-CRF (Ours)	<b>91.63</b>

Table 2.4: Comparison of F scores of different models for NER

Model	Accuracy
Conv-CRF [57]	97.29
5wShapesDS [147]	97.32
Structure regularization [222]	97.36
Multitask learning [192]	97.43
Nearest neighbor [217]	97.50
LSTM-CRF [123]	97.51
LSTM-CNN-CRF [145]	97.55
LM-LSTM-CRF [138]	97.59
BLSTM-CRF (Ours)	97.51
BLSTM-CRF (Ours) without CNN	<b>97.58</b>

Table 2.5: Comparison of accuracy of different models for POS tagging

SENNA embeddings [57] along with a gazetteer feature, their F-score jumped up to 90.10% surpassing the Conv-CRF model [57] which uses window and sequence approach networks to do the tagging. Our model achieves a state of the art result of **91.63%**.

Table 2.5 shows the performance of our architecture in comparison with some top perform-

Model		Acc.	Time
[138]	LSTM-CRF	97.35	37
	LSTM-CNN-CRF	97.42	21
	LM-LSTM-CRF	97.53	16
	LSTM-CRF	97.44	8
	LSTM-CNN-CRF	96.98	7
Ours	BLSTM-CRF	97.51	4
	BLSTM-CRF without CNN	97.58	3.5

Table 2.6: Training time (hours) of our BLSTM-CRF model on the WSJ corpus compared with all models of [138] using the same hardware configuration (GPU: Nvidia GTX 1080)

Word emb	Sense	SP CLSTM	Bigram	Suffix		Spelling			Char Embed		Prev. POS	Acc.
				CW	CO	R	CW	CO	CW	CO		
Rand.	x	x	x	-	-	-	-	-	-	-	x	95.42
Glove	x	x	x	-	-	-	-	-	-	-	x	96.13
Glove	✓	x	x	-	-	-	✓	-	-	✓	x	97.08
Glove	✓	✓	x	✓	-	✓	-	-	✓	-	x	97.15
Glove	✓	✓	x	✓	-	-	✓	-	✓	-	x	97.22
Glove	✓	x	x	-	✓	-	-	✓	✓	-	x	97.32
Glove	✓	✓	x	-	-	-	-	✓	✓	-	x	97.45
Glove	✓	✓	x	✓	-	-	-	✓	✓	-	x	97.48
Glove	✓	✓	✓	✓	-	-	-	✓	✓	-	x	97.50
Glove	✓	✓	x	✓	-	-	-	✓	✓	-	✓	<b>97.58</b>

Table 2.7: Ablation study of our BLSTM-CRF model for POS tagging. (R - Residual connection, CW - Concatenate with word embedding, CO - Concatenate with second last output layer)

Module	100th epoch	200th epoch	300th epoch	400th epoch	500th epoch
Word emb	0.91	0.84	0.80	0.77	0.78
Sense	0.85	0.76	0.69	0.64	0.65
SP-CLSTM	0.81	0.66	0.48	0.35	0.34
Bigram	0.75	0.49	0.27	0.01	0.01

Table 2.8: Change in  $w$ 's for each module with epochs.

ing ones for the POS tagging task. As can be seen, a number of models use Convolution or LSTM or BLSTM to get the contextual features and CRF to do the tagging. They achieve very good accuracies of 97.29% [57], 97.51% [123] and 97.55% [145]. Some of the models use multitask learning, doing two or more tasks at the same time. They also achieve very good accuracies: 97.43% [192] and **97.59%** [138]. Our model achieves an accuracy of **97.58%** which is higher than all of the existing models except LM-LSTM-CRF [138] which leverages a language model for the tagging tasks. LM-LSTM-CRF, however, has a mean accuracy of



**97.53%** (reported accuracy:  $97.53 \pm 0.03$ ) which is lower than the our model’s mean accuracy ( $97.57 \pm 0.01$ ). Also, as shown in Table 2.6 our model’s training time is one quarter that of LM-LSTM-CRF with on par performance.

Table 2.7 gives the ablation study of our model where we show how we apply different combinations of features in different parts of our BLSTM-CRF architecture to get an optimal configuration. With so many features and parameters, these sequence models are very much prone to overfit. But with careful tuning as well as with proper feature connections, it is possible to leverage those features. We extract a set of morphological as well as semantic features from our dataset such as spelling, suffix and char-level features. We experiment on applying various combinations of these features in different segments of our model. Our extensive experimentation shows that optimal results are achieved when these features are added in the model through residual connection (R), concatenation with word embeddings (CW) and concatenation with the second last output layer (CO). Focusing on which segment to connect each feature, our experiments found that the spelling feature works best when concatenated with the second last output layer, and the suffix feature as well as the character embeddings work well when concatenated with the word embeddings. This configuration is what is kept in our final model. We further continue our experiments by turning on / off different modules such as word embedding, sense embedding, selective pickup from LSTM and bi-gram embedding. We found significant contribution of word embeddings, sense embeddings and selective pickup from LSTM compared to the bigram modules as shown by the weights at the 500th epoch in Table 2.8. The bigram module gives better performance without considering previously generated POS and vice versa. However, linguistically, the information about the previous tag has a huge influence in generating the current one. So we kept the first three modules along with the previously generated POS and discarded the bigram module from our final model. Our best model as shown in the last row of Table 2.7 gives state of the art results.

## 2.6 Conclusion

In this chapter, we propose an improved neural sequence labelling architecture by leveraging from additional linguistic information such as polysemy, bigrams, character level knowledge and morphological features. Benefitting from such adequately captured linguistic information, we can assemble a considerably more compact model, hence yielding much better training time without loss of effectiveness. To avoid feature collision we performed an extensive ablation study where we produced an optimal model structure along with an optimal set of features. Our best model achieved state of the art results on the POS tagging, NER and chunking benchmark datasets and at the same time remains four times faster to train than the best performing model

currently available. Our experimental results show that multiple linguistic features and their proper inclusion significantly boosted our model performance.

## Chapter 3

# A Novel Neural Sequence Model with Multiple Attentions for Word Sense Disambiguation

This chapter is based on the paper titled “A Novel Neural Sequence Model with Multiple Attentions for Word Sense Disambiguation” co-authored with Muhammad Rifayat Samee and Robert E. Mercer that appeared in the 17th IEEE International Conference on Machine Learning and Applications (ICMLA 2018) [12].

Word sense disambiguation (WSD) is a well-researched problem in computational linguistics. Various research works have approached this problem in different ways. Some state-of-the-art results that have been achieved for this problem in terms of accuracy use supervised models, but they often fall behind flexible knowledge-based solutions which use engineered features as well as human annotators to disambiguate every target word. This chapter focuses on bridging this gap using neural sequence models incorporating the well-known attention mechanism. The main gist of our work is to make a weighted combination of multiple attentions on different linguistic features and to provide a unified framework to accomplish this. This weighted attention allows the model to easily disambiguate the sense of an ambiguous word by attending to a suitable portion of a sentence. Our extensive experiments show that weighted multiple attention enables a more versatile encoder-decoder model leading to state-of-the-art results.

## 3.1 Introduction

Word sense disambiguation (WSD) is the task of assigning an appropriate meaning to a target word when the target word sense is clearly distinguishable from its other word senses subject to the attributes of the target word’s context. As one of the challenging problems in the field of computational linguistics, WSD has received considerable attention over the past decade [11, 165] due to its various application potentials such as information retrieval, text mining, machine translation, speech synthesis as well as question answering. Some of the classical approaches are the LESK algorithm, which uses the overlap of the words in the dictionary definition sense and the words in the target sentence; Naive Bayes, which looks at the conditional probability of each word sense along with the contexts with the assumption that word order as well as inclusion in a bag of words is independent; and Neural Networks, where nodes representing the senses of words in a sentence stabilize on one sense more than the others.

Much research has attempted to solve this classic WSD problem, evaluating new algorithms [43, 236, 256] on some of the well-known benchmarks [161, 166, 206]. This recent work focuses on two known WSD difficulties: order of the words in the context and the use of handcrafted features. Most of the traditional supervised WSD methods are based on extracting features from the surrounding words and then training a classifier for each of the ambiguous words [268].

Recently, deep neural network-based approaches have gained state of the art results in many widely examined classical problems in computational linguistics. In this chapter, we propose a new neural network architecture for WSD by taking linguistic features of the surrounding context words into account. WSD has been viewed as assigning the correct word sense to a word as the task of translating the target sentence to a sentence containing the sense-tagged words. The neural architecture developed here is a variant of the sequence to sequence (Seq2Seq) architecture that has been successfully applied to machine translation. As candidate features, we investigate three: surrounding word vectors, surrounding context bigrams and the parts of speech (POS) sequence of the whole sentence. Although Raganato et al. [189] show that Seq2Seq is sub-optimal for doing WSD, this chapter revisits this finding and explores the effectiveness of various attentive encoder-decoder architectures for this task. The novelty of our method is that we use multiple attentions to decide the significance of each of the three features and amalgamate these features in a vector as a weighted linear combination where the weights either scale up or scale down every dimension of the corresponding features accordingly. Even though we haven’t been able to take the entire corpus into account because of resource limitations, our Seq2Seq architecture with multiple attentions have obtained state of the art performance on some of the benchmarks.

## 3.2 Related Work

WSD can be viewed as a machine translation task. Neural machine translation (NMT) is a recently proposed approach for the translation task [24, 106, 224]. Unlike the conventional phrase-based translation framework [114] which comprises numerous small sub-segments that are tuned independently, NMT uses a vast encoder-decoder based neural network that takes a sentence and yields a translation. Bahdanau et al. [24] proposed the attention mechanism, a breakthrough in NMT, to do translation as well as alignment jointly. It calculates how much attention the network should give to each source word to generate a specific translated word. Luong et al. [144] then proposed two new attention mechanisms: one looks for the global context and the other one looks for the local context, i.e., a subset of the words in the sentence. Although these Seq2Seq models capture word level features very well, Sennrich et al. [200] achieved better results by adding some linguistic features such as part-of speech tags, morphological features, and syntactic dependency.

A number of recent works have adopted the Seq2Seq concept for WSD. Among them, Raganato et al. [189] have experimented with some neural sequence models which include a bidirectional long short term memory (BLSTM) based architecture in a many to many format with and without attention for sequence tagging. They also experimented on Seq2Seq architectures with attention to do multitask learning where the words in a sentence are tagged with their sense as well as their parts of speech. Their best performing model is an attentive BLSTM tagger rather than the Seq2Seq architecture. Melamud et al. [155] trained a BLSTM architecture to get the context representation of each sense annotation on an unlabeled corpus. Kaageback et al. [105] relied on a BLSTM-based approach where they divided the sentence into two pieces, the left and right contexts, based on the position of the target word. They applied two long short term memories (LSTMs) from opposite directions on these contexts, concatenated their last hidden states and used a multi-layer perceptron to classify the target word sense. Yuan et al. [256] proposed a powerful neural language model to obtain a latent representation for an entire sentence containing a target word  $w$ . They then compare this representation with those sentences which have other candidate senses of word  $w$ . Ahmad et al. [184] suggested an architecture to calculate the cosine similarities between the sense embedding of the center word and the word embedding of every other word in a sentence. Then they applied two LSTMs on this vector of similarities, one from the left direction and one from the right, concatenated them and finally applied a fully connected layer to classify the word sense as a one class classification problem.

In this chapter, we incorporate ideas from this previous work in a few encoder-decoder architectures for WSD by taking linguistic features of the surrounding context words into ac-

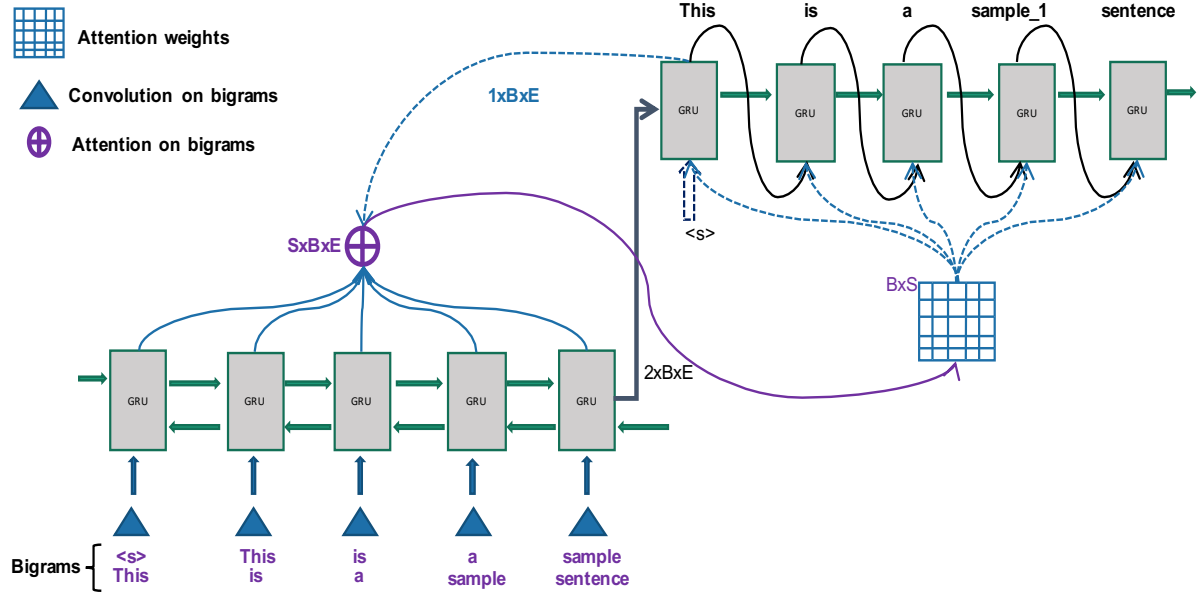


Figure 3.1: Encoder-decoder architecture for sequence-to-sequence WSD with attention on bigrams. ( $S$  = Sentence length,  $B$  = Batch size,  $E$  = Embedding dimension, and  $\langle s \rangle$  = Start token)

count, combining them and at the same time capturing the order of the context words as well.

### 3.3 The model

In this section, we describe our work in detail. The Seq2Seq architecture is being used as a sequence tagger and the attention mechanism is being used to inform the network how much attention needs to be paid on each linguistic feature to identify the specific meaning of an ambiguous word. We use this supervised attention-based method to generate a linear combination of different features as well as to generate a final linguistic feature based attention matrix. Three linguistic features, surrounding word vectors, surrounding context bigrams and the POS sequence of the whole sentence, have been added to these models to improve their performance. We first explain our basic Seq2Seq model having attention on bigrams. Following this, we explain a way of doing multiple attentions on different features and finally, we describe a way of combining these multiple attentions using some weighted value in the latter part of this section.

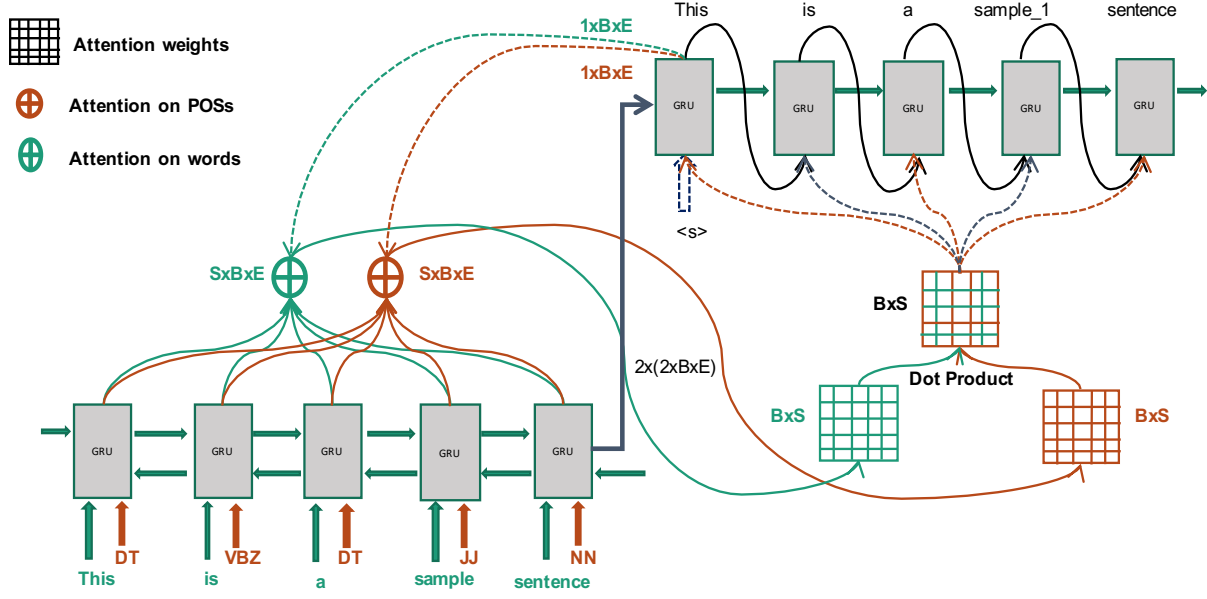


Figure 3.2: Encoder-decoder architecture for sequence-to-sequence WSD, taking point-wise multiplication of POS and bigram attention weights. ( $S$  = Sentence length,  $B$  = Batch size,  $E$  = Embedding dimension, and  $\langle s \rangle$  = Start token)

### 3.3.1 Attention on Bigrams

Our first architecture is based completely on the attention based Seq2Seq model with encoders and decoders as shown in Figure 3.1. The encoder inputs the source sentence as a sequence  $[x_1, x_2, \dots, x_t, \dots, x_S]$ , with  $x_t$  as the ambiguous word. The decoder tries to generate a sequence  $[y_1, y_2, \dots, y_t, \dots, y_S]$ , where  $x_i = y_i$  for all  $i$ , except the target word at index  $t$  is replaced by the corresponding sense-tagged word. Adding an attention mechanism with this architecture allows the model to take one word at a time from the decoder and look for which of the words in the input sentence are useful for generating this target word. However, rather than using word by word attention, we use bigram attention. This makes sense because to generate a particular sense of a word, the contribution of the context words matters most [126]. Next, we pad a start token  $\langle s \rangle$  at the beginning of this sentence and then pass this modified sentence to an embedding layer. The embedding weights are initialized randomly as well as using pre-trained word vectors and both are trained along with other parameters of the network. Next is a convolution layer with a kernel of size  $2 \times \text{embedDim}$  which goes over the bigram embedding with a stride length of 1 as shown in Eqn. 3.1

$$B_{t,:} = \sum_{j=1}^2 E_{t+j,:} * K_{j,:} \quad (3.1)$$

where  $E$  is the embedding matrix,  $K$  is the convolution kernel and  $S$  is the maximum sequence length for the current batch. This will generate a convolved embedding of bigrams,  $B$ , which is then fed to the bi-directional gated recurrent unit (BiGRU) layer.

$$h_t = \text{BiGRU}(B_t, h_{t-1}) \quad (3.2)$$

The last hidden state  $h_S$  is the encoded representation of the sentence and we term this as  $h_{enc}$ . Next, in the decoder section, a uni-directional GRU is initialized with  $h_{enc}$  as the hidden state and  $\langle s \rangle$  as input. This generates a new hidden state  $\tilde{h}_t$ .

$$\tilde{h}_t = \text{GRU}(\langle s \rangle, h_{enc}) \quad (3.3)$$

This  $\tilde{h}_t$  and the encoder output  $O$  at each time step is passed to an ‘Attention’ model which returns an attention matrix  $A$  of size  $1 \times S$ .

$$A = \text{Attention}(O, \tilde{h}_t) \quad (3.4)$$

Next, batch-wise matrix multiplication is applied on  $A$  and  $O$  and the context  $C_{1 \times d}$  is computed.

$$C = A.bmm(O) \quad (3.5)$$

The  $\tilde{h}_t$  and  $C$  are concatenated and passed to an MLP followed by a Softmax layer which maps the result back to the vocabulary size.

$$\begin{aligned} \tilde{Y}_t &= \text{MLP}([\tilde{h}_t, C]) \\ Y_t &= \text{Softmax}(\tilde{Y}_t) \end{aligned} \quad (3.6)$$

In the next time step, the decoder GRU again unfolds. But this times it takes the hidden state and the word generated from the previous time step into account as shown in Eqn. 3.7

$$\tilde{h}_t = \text{GRU}(\tilde{Y}_{t-1}, \tilde{h}_{t-1}) \quad (3.7)$$

The rest of the training is done in an end-to-end fashion.

### 3.3.2 Attention on Words and Parts of Speech (POS)

In this model, we introduce a multiple attention mechanism where individual attention is applied on different features of the data which are later combined through point-wise multiplication. We start with a traditional Seq2Seq architecture having an encoder and decoder at both



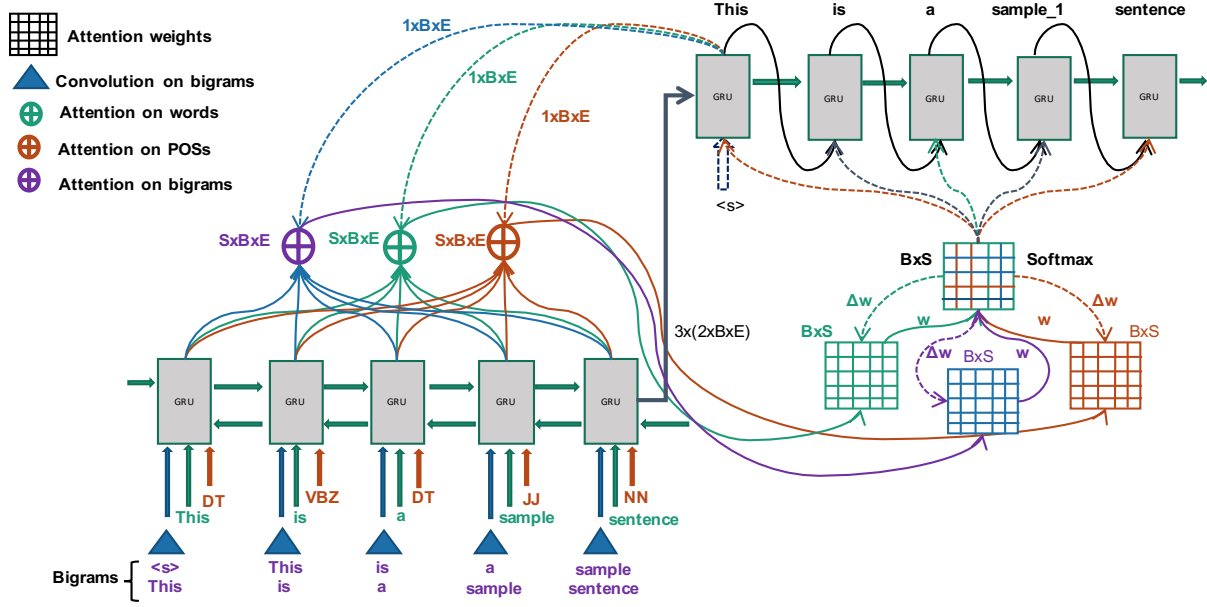


Figure 3.3: Encoder-decoder architecture for sequence-to-sequence WSD with weighted combination of multiple attentions. ( $S$  = Sentence length,  $B$  = Batch size,  $E$  = Embedding dimension, and  $\langle s \rangle$  = Start token)

ends. As shown in Figure 3.2, the encoder has a GRU layer which takes word embeddings as well as POS embeddings as input. Rather than having a different GRU layer for words and POS, we use a single GRU layer whose weights are shared for both of these inputs. This also allows the gradients to be shared as well during back-propagation. Next, using Eqn. 3.2, the encoded word and POS features are calculated as given by Eqn. 3.8.

$$\begin{aligned} h_t^w &= \text{GRU}(W_t, h_{t-1}) \\ h_t^p &= \text{GRU}(P_t, h_{t-1}) \end{aligned} \quad (3.8)$$

where  $h^w$  and  $h^p$  represent the hidden state for the word and the POS, respectively. From each of these hidden state vectors, we consider the one at the last index as the encoded version. Following this, Eqns. 3.3 and 3.4 are applied to these encoded hidden state vectors giving two attention matrices  $A_w$  and  $A_p$  for the word and the POS, respectively. The final attention matrix  $\tilde{A}$  is obtained with the point-wise multiplication of  $A_w$  and  $A_p$  in Eqn. 3.9.

$$\tilde{A} = A_w * A_p \quad (3.9)$$

The point-wise multiplication changes the amplitude of each dimension of these vectors and also allows encoding the positions of the target word neighbors according to their vector amplitude. If the word POS attention matrices both put more focus on the  $i^{\text{th}}$  word then the magnitude

of the final attention vector at the  $i^{th}$  dimension becomes very large. And if two attention vectors put focus on two different words, then the final attention gets distributed over those two possible words. This makes it similar to a soft-hard attention model, where the model decides which one gets activated and when. Next, to make the final attention matrix  $A$  as a probability distribution, a Softmax layer is applied as shown in Eqn. 3.10

$$A = \text{Softmax}(\tilde{A}) \quad (3.10)$$

Finally, Eqn. 3.5 is used to calculate the context followed by Eqn. 3.6 to generate the next predicted word. Similar to the method in Subsection 3.3.1, Eqn. 3.3 is replaced by Eqn. 3.7 from the second time step to generate the hidden state vector for the decoder. The decoder continues to decode until it generates all words in the target sentence or an EOS token is encountered. The rest of the training is done in an end-to-end manner.

### 3.3.3 Attention on Words, POS and Bigrams with weighting

In this model, we combine the concept of both bigrams and POS attention architecture from Subsections 3.3.1 and 3.3.2 and introduce a term called weighted attention. As shown in Figure 3.3, the input GRU in the encoder takes word embeddings, POS embeddings and bigram embeddings. Similar to our previous model, the GRU weights are shared among these three inputs. Eqn. 3.2 is used on the three inputs independently and three encoded vectors  $h_w$ ,  $h_p$  and  $h_b$  are calculated for the word, POS and bigram, respectively. Then, Eqns. 3.3 and 3.4 are applied on these encoded vectors independently and three attention vectors  $A_w$ ,  $A_p$  and  $A_b$  are calculated. Next, three weights  $w_1$ ,  $w_2$  and  $w_3$  are generated and a weighted linear addition of the three attention vectors is performed as shown in Eqn. 3.11.

$$A = (A_w * w_1) + (A_p * w_2) + (A_b * w_3) \quad (3.11)$$

Following this, Eqn. 3.10 is used to turn these attention weights into probabilities. Finally, calculating the context and generating the next probable word is similar to the one described in Subsections 3.3.1 and 3.3.2. The decoder continues to generate until an EOS token is found or the entire sequence is generated.

**Generating weight values** Currently, most research uses attention to decide which portion of a feature the model needs to pay more attention to. To the best of our knowledge, none has investigated putting attention on attentions. In this study, we propose a few ways to do this. Firstly, if a model has multiple features with a separate attention for each, the simplest method is to combine all the attentions and perform a Softmax on them. Another way, formalized

in Eqn. 3.12 is to apply a local gating mechanism, where each of the individual attention vectors is first passed to a Sigmoid layer. This generates a value in  $[0, 1]$  for each attention vector. Then a point-wise multiplication is performed on these individual attention vectors and their sigmoid values. Finally, a Softmax is applied on their sum, and the result is used as the attention vector.

Apart from local gating, it is also possible to apply a global gating mechanism, formalized in Eqn. 3.13. First, the attention vectors are concatenated and then Softmax is applied on individual columns of this concatenated matrix. The best features for each position are chosen via  $\text{argmax}$  applied on each column and then concatenate them to create a vector. Following this, Softmax is applied on this vector to get the final attention vector of probabilities.

$$\begin{aligned}
 \tilde{A}_1 &= \text{Sigmoid}(A_1) \\
 \tilde{A}_2 &= \text{Sigmoid}(A_2) \\
 \tilde{A}_3 &= \text{Sigmoid}(A_3) \\
 W &= (\tilde{A}_1 * A_1) + (\tilde{A}_2 * A_2) + (\tilde{A}_3 * A_3) \\
 A &= \text{Softmax}(W)
 \end{aligned} \tag{3.12}$$

Another way to generate the weights is to scale each of the attention vectors with a scaling factor and then add them. The factor can be a vector or a scalar. To do this, the factor is first initialized with some random values. Then it is added to the model parameters where its gradient is calculated based on loss. If the factor is a vector then we can choose an MLP without a bias for doing the transformation. In this study, we choose scalar factors as weights, initialize them to 1.0 and then perform a linear weighted addition of the attention vectors. After the addition, a Softmax is applied on the resultant vector in order to make it a vector of probabilities. In the next section we show that even though we start with weight values of 1.0, by taking gradients during training, the model adjusts the values accordingly and we end up with a different set of values.

$$\begin{aligned}
 \hat{A} &= [A_1; A_2; A_3] \\
 \tilde{A}_{:,i} &= \text{Softmax}(\hat{A}_{:,i}), \forall i \\
 W_i &= \text{argmax}(\tilde{A}_{:,i}), \forall i \\
 A &= \text{Softmax}(W)
 \end{aligned} \tag{3.13}$$

## 3.4 Experimental Setup

In this section we detail the experimental setup used for evaluation. We first describe our training corpora as well as all of the benchmarks used in other standard WSD studies. Following this, we explain the technical details of our proposed architectures along with their hyper-parameter settings.

**Training and Evaluation Benchmarks:** SemCor 3.0, MASC and Senseval task 3 corpora are used for evaluating our models. Many existing works have used these corpora as their benchmark [86, 105, 128, 219, 226, 256]. As our models are configured to work with WordNet senses, we use the mapping algorithm proposed by [256] to map the SemCor and MASC corpora from NOAD senses to WordNet senses. We have evaluated our architectures in the normal way using standard splits of these corpora for the training-testing sets. As well, we performed a cross domain evaluation—training on one corpus and testing on another. For this evaluation, the whole corpus from one domain is used for training and the entire corpus from another is used for testing.

Model selection is done using the validation set of each corpus. The best model is then trained on a combined training and validation set and evaluated on the test set. As we have not found any standard splits for SemCor and MASC corpora, we perform a manual split: train, test and validation with a ratio of 80%, 10%, and 10%, respectively. The hyper-parameters are tuned on the validation set only.

During testing, our models calculate the probability distribution over the output words  $O$  given a target word  $w$ . The output  $O$  at each time step is given to a Softmax layer which gives the probability for each class. It is then used to rank the candidate senses of  $w$  and the top ranked candidate is selected as the output of the model.

**Architecture details and network parameters:** For all three architectures, we use GRU as the basic building block. For the first architecture we use single attention. For the other two architectures we use multiple attentions. We use the ‘dot’, ‘concat’, and ‘linear’ attention models from [144] to calculate the attention energies. As all of the models were giving the best results with the ‘dot’ attention model, we report our final experimental results in the next section only with the ‘dot’ attention model.

Table 3.1 shows the detailed hyper-parameter settings used during the evaluation for all three of our architectures. We trained our models on a GeForce GTX 1080 GPU with both ‘Adam’ and ‘SGD’ optimizers. All the results in the next section are reported using ‘SGD’ as it was giving comparatively better results. We used PyTorch 0.3.1 for implementing our models under the Linux environment.

Hyper-parameter	Range Selected
Learning rate	<b>0.01</b> / 0.02 / 0.001
Context size	<b>50</b> (25 on each side)
Batch size	10 / <b>50</b> / 100
No. of GRU layers	Encoder - <b>2</b> / 3 / 4 Decoder - <b>2</b> / 3 / 4
Type of GRU layer	Encoder - bi-directional Decoder - uni-directional
Dropout	<b>0.1</b> / 0.2 / 0.3
Word embedding size	<b>100</b> / 200 / 300
Initialization of scalar weights on Attentions	<b>Random number</b> $\in$ uniform (-0.1, 0.1)
Decoder learning ratio	<b>5.0</b>
Gradient clipping	<b>50</b> / 25 / 10

Table 3.1: Hyper-parameters used for the experiments (in boldface) and the ranges that were searched during tuning.

Model	SE03	nn.	vb.	adj.	adv.
SEQ2SEQ	66.3	47.2	45.1	59.2	68.1
SEQ2SEQ + CONVOLUTION (BIGRAMS)	59.2	-	-	-	-
SEQ2SEQ + PART OF SPEECH (POINT-WISE MULTIPLY)	57.1	-	-	-	-
SEQ2SEQ + PART OF SPEECH (WEIGHTING)	67.5	49.1	58.3	61.2	68.1
SEQ2SEQ + CONVOLUTION + PART OF SPEECH (WEIGHTING)	<b>73.9</b>	58.2	<b>70.4</b>	71.3	<b>85.4</b>
BLSTM + ATTENTION [189]	70.2	71.0	58.4	75.2	83.5
SEQ2SEQ + ATTENTION [189]	69.6	69.5	57.2	74.5	81.8
SEQ2SEQ + ATTENTION + SEMANTIC LABEL + POS [189]	68.5	70.1	55.2	75.1	84.4
CONTEXT TO VECTORS (CONTEXT2VEC) [155]	69.1	71.2	57.4	75.2	82.7
IT MAKES SENSE + EMBEDDING (IMS+EMB) [97]	70.4	<b>71.9</b>	56.6	<b>75.9</b>	84.7
RANDOM WALK + KNOWLEDGE BASE (UKBGLOSS-w2w) [2]	55.4	64.9	41.4	69.5	69.7
BABELFY [162]	67.0	68.9	50.7	73.2	79.8
BLSTM [105]	73.4	-	-	-	-
IT MAKES SENSE + ADAPTIVE CONTEXT WIDTH [226]	73.4	-	-	-	-
Htsa3 [86]	72.9	-	-	-	-
IRST-KERNELS [219]	72.6	-	-	-	-
NUSELS [128]	72.4	-	-	-	-

Table 3.2: F-scores (%) for the English all-words coarse-grained WSD on the Senseval-3 corpus. Best performing models are given in boldface. POSs: **nn.** = Noun, **vb.** = Verb, **adj.** = Adjective, and **adv.** = Adverb.

### 3.5 Experimental Results

In this section, we describe the evaluation of our models. For the Senseval-3 corpus, a comparison with the previously best performing models in terms of F(%) score is provided in Table

[3.2](#). We show the consistency of our models across different part of speech by reporting F scores on four major classes in Table [3.3](#). We show which linguistic feature has more impact on determining the sense of a word through the weights described in Table [3.4](#). Finally, we conclude this section by showing how multiple attention can penalize feature collision in the model by doing a controlled amalgamation of the scaled attention vectors in Figure [3.4](#). For a more complete evaluation, we implemented Seq2Seq with word attention as a baseline model. Also, to see the impact of a weighted combination of different linguistic features, we implemented Seq2Seq+POS(WEIGHTING) model which is similar to the model described in Subsection [3.3.3](#) with the bigram attention module removed.

Table [3.2](#) compares the F score achieved by our proposed models against some of the existing state of the art models on the Senseval-3 task. Our Seq2Seq+CONV+POS(WEIGHTING) is the best performing among the five models that we experimented with. It outperformed the previously top performing neural sequence model from [\[105\]](#) on the Senseval-3 task and achieves a state of the art F score of **73.9%**. An interesting aspect is that when the POS feature is added through point-wise multiplication, the performance drops (from **66.3%** to **57.1%**) because it causes inconsistent scaling of different dimensions of the attention vectors. However, adding the same feature through weighted addition causes a performance boost (from **66.3%** to **67.5%**) as each dimension of the final attention vectors now is stretched uniformly based on the two individual attention vectors. Table [3.2](#) also reports the F scores of individual POS classes for the Senseval-3 task. Our best model achieves state of the art results on *vb.* and *adv.* classes with F scores of **70.4%** and **85.4%**, respectively. The best results on *nn.* (**71.9%**) and *adj.* (**75.9%**) are achieved with an IMS framework along with word embeddings to generate features and classification with a support vector machine (SVM) [\[97\]](#). Apart from that, it can easily be seen that the Seq2Seq architectures perform very well against the statistical and knowledge-based methods like IMS+ADAPTED CW [\[226\]](#), HTSA3 [\[86\]](#), UKBGLOSS-w2w [\[2\]](#), BABELFY [\[162\]](#) as well as IRST-KERNELS [\[219\]](#) achieving results that are superior or equivalent to the best models mentioned above. One interesting evaluation is that the Seq2Seq baseline from [\[189\]](#) is **69.6%** while our Seq2Seq baseline performance is **66.3%**. When [\[189\]](#) added POS, where it is meant to be learned as one of their tasks, their performance degrades from **69.6%** to **68.5%**. When we added POS as a feature, our performance jumped from **66.3%** to **67.5%** which clearly shows that adding POS information as a feature has an influence on identifying polysemy of a word. It is to be noted that the variation in baseline model performance may be due to different hyper-parameter settings or different hardware configuration.

Table [3.3](#) shows an extensive evaluation of our models on the SemCor and MASC corpora with various training and testing environments. It also shows our model performance on different parts of speech classes on these two corpora. It is clear that whenever testing on the same

Model	Train	Test	nn.	vb.	adj.	adv.	all
Seq2Seq	MASC	MASC	38.2	57.1	60.8	66.5	66.4
	SemCor	SemCor	36.1	45.2	41.3	49.6	70.1
	SemCor	MASC	23.2	44.1	48.1	48.9	60.3
	MASC	SemCor	22.2	36.6	32.4	53.7	59.0
Seq2Seq + Part of Speech(weighting)	MASC	MASC	39.9	59.7	57.5	68.8	68.1
	SemCor	SemCor	37.7	47.6	41.5	50.9	76.4
	SemCor	MASC	26.6	43.3	48.3	49.0	62.6
	MASC	SemCor	24.1	37.0	35.2	52.8	63.5
Seq2Seq + Convolution + Part of Speech(weighting)	MASC	MASC	46.5	65.2	67.0	74.1	72.7
	SemCor	SemCor	42.1	52.1	53.9	72.5	76.3
	SemCor	MASC	29.2	45.7	42.8	53.4	65.2
	MASC	SemCor	26.1	40.0	38.2	53.6	65.8

Table 3.3: F-scores (%) of our top performing models for different parts of speech with different settings of SemCor and MASC corpus. POSs: **nn.** = Noun, **vb.** = Verb, **adj.** = Adjective, and **adv.** = Adverb.

Attention weights	Initial value	1000 epoch	2000 epoch	4000 epoch	5000 epoch	3200 epoch
On words ( $w_1$ )	0.41	0.74	0.48	0.35	0.34	0.32
On bigrams ( $w_2$ )	0.33	2.02	2.69	3.12	3.65	3.45
On POS ( $w_3$ )	0.20	2.14	2.23	3.38	3.37	3.58

Table 3.4: The change in different attention weights with epochs for Seq2Seq + conv + POS (weighting).

domain (Train-MASC, Test-MASC and vice versa), all the models perform quite well with maximum F-scores up to **72.7%** and **76.4%** for MASC and SemCor, respectively. However, while testing on different domains (Train-MASC, Test-SemCor and vice versa), performance of all the models decreases. It does make sense because even though we are in different domains, we are not tuning any of the hyper-parameters; instead it's been set according to their prior training environment. Table 3.3 also depicts how well our best models perform on some frequent parts of speech classes. For almost all of the POS classes, our top performing model, SEQ2SEQ+CONV+POS(WEIGHTING), outperforming the other models by a good margin. Variance in the performance on different parts of speech is mainly due to some statistically significant differences between the models. However, one certainty is that the results for training and testing in different domains is very much correlated with what is shown in Table 3.2. We have not included the results for the comparatively weak models, SEQ2SEQ+CONV(BIGRAMS) and SEQ2SEQ+POS(POINT-WISE MULTIPLY).

Table 3.4 depicts the pattern of change across epochs in scalar weights on different attentions which shows how the model decides the amount of attention it needs to pay on different



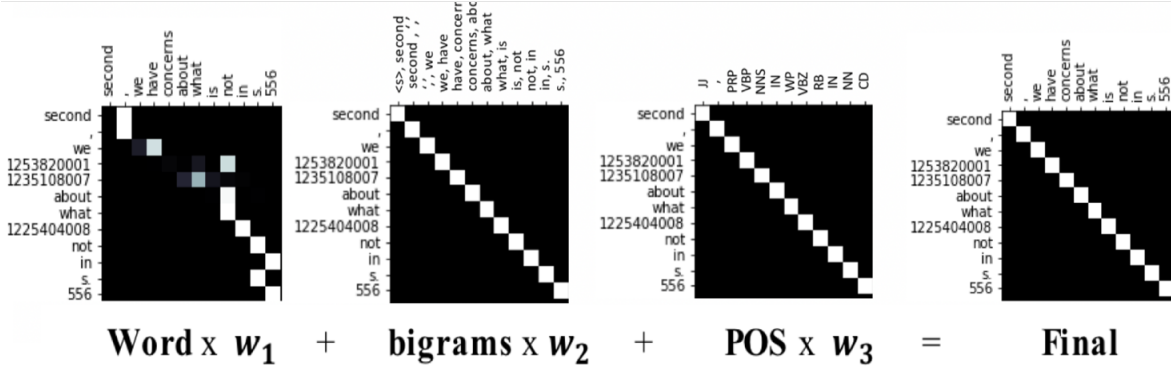


Figure 3.4: The effect of different attention weights on final attention matrix. (Numbers on the left of each Figure indicates a unique sense id of the corresponding true word on that position)

linguistic features. We start with random scalar weights on three possible features as shown in the first column of Table 3.4. It can easily be seen that as the model sees more data, it gives more importance to POS with weight **3.58** compared to the other two (**0.32** for words and **3.45** for bigrams). The weight on word attention is not stable but the weights on the others are increasing monotonically until the loss gets very small. We randomly sample 10% of the data and use this as our development set. Our model achieves the highest development set F score at the 3200<sup>th</sup> epoch. Evaluation of our model uses these weights.

Figure 3.4 shows how the model has different attentions on different linguistic features. It is clearly evident that with just word attentions, the model is confused and gives attention to more than one word at a given time. This makes sense for a translation model as one particular word in one language at a specific index position can depend on more than one word of another language. But as we are dealing with the same language at both ends, the decoded word attention has to be on the same word from the encoder; in other words, it should be a one to one mapping. The deviation is due mainly to the lack of context. By making a new attention matrix with the linear weighted combination of multiple attentions on different linguistic features, we penalize this lack of context. This linearly combined attention matrix is finally passed through a Softmax layer to make each attention weight a probability.

### 3.6 Conclusion and Future Work

In this chapter, we adopted a new approach for doing WSD using neural sequence models by applying multiple attentions on different linguistic features of a sentence. The single attention approach with sequence models is very effective with machine translation however this study focuses on using multiple attentions and taking their linear weighted combinations. By making these weights a network parameter, the model can easily fit itself to a suitable combination



of them. Our best model achieves state of the art results on Senseval-3 corpus. Also our in depth analysis on POS classes of all the corpora gives us insight into how polysemy relates to POS. The multiple attention approach largely impacts the model by giving it more flexibility to choose the right combination of the suitable features.

This approach can easily be applied to other applications of neural sequence models such as question answering. The most related fact can be chosen through an attention over all the fact sentences. The answer is predicted through another attention on the question. As future research, we are currently working on this idea.

# Chapter 4

## Improving Tree-LSTM with Tree Attention

This chapter is based on the paper titled “Improving Tree-LSTM with Tree Attention” co-authored with Muhammad Rifayat Samee and Robert E. Mercer that appeared in the 13th International Conference on Semantic Computing (ICSC 2019) [13]. The paper was awarded "Best Paper Award" and "Best Student Paper Award".

In Natural Language Processing (NLP), we often need to extract information contained in the topology of trees. Sentence structure can be represented via a dependency tree or a constituency tree. For this reason, a variant of LSTM, named Tree-LSTM, was proposed to work on tree topology. In this chapter, we design a generalized attention framework for both dependency and constituency trees by encoding variants of decomposable attention inside a Tree-LSTM cell. We evaluate our models on a semantic relatedness task and achieve notable results compared to Tree-LSTM-based methods with no attention as well as other neural and non-neural methods, and good results compared to Tree-LSTM-based methods with attention.

### 4.1 Introduction

Long Short Term Memory (LSTM) units are very effective when working on sequential data [81, 92]. For some Natural Language Processing (NLP) tasks, we often need to find a distributed representation of phrases and sentences [62, 125, 135]. One obvious way of doing this is to use a sequential LSTM which captures word order in a sentence [172, 262]. But we can also have information about sentence structure from a dependency parse tree or about phrase structure from a constituency tree [112]. Despite the fact that RNN based models work well with sequence information, they frequently neglect to catch any sort of semantic compositionality if the information is structured rather than in the sequential frame [213]. For example, the

syntactic principles of natural language are known to be recursive, with noun phrases containing relative clauses that themselves contain noun phrases, e.g., “*I went to the church which has nice windows*” [210]. The term compositionality can also be explained in terms of a *car*. A *car* can be recursively decomposed into smaller *car* parts, for example, tires and windows and these parts can occur in different contexts, like tires on airplanes or windows in houses.

Attention [24, 144, 240] was first introduced for doing machine translation where the target word generated by the decoder at each time step is aligned with all the words in the source sentence. In its general form, attention allows a model to put importance on certain parts of the sentence for doing any specific downstream task [71, 247]. In a dependency tree, the relationship between the entities (head and dependent) are organized as a structure where a head word can have multiple dependents under it. In the case of a constituency tree, a phrase is represented by one of the subtrees with the root being the phrase type and words or subtrees being the children. In both tree structured LSTMs, the derivation of the vector representation of the entire tree does not depend on all of the subtree components uniformly. Some parts of the tree have a larger influence on the root vector and some parts may have less. This contribution from subtrees for the building of the whole tree depends on the underlying task that the model is performing. For example, in a sentiment analysis task the sentiment of a tree depends on the sentiment of all of its children and how this information propagates. There may be scenarios where a single word (such as “not”) flips the sentiment of the whole subphrase. These words should get more attention when deciding the sentiment of a subphrase containing them. On the other hand, when the problem is a regression problem with the task of assigning a score based on the semantic similarity of two sentences, this attention can be calculated as a cross sentence attention. In this case the representation of one sentence can guide the structural encoding of the other sentence on the dependency as well as constituency parse tree [270].

Capturing semantic relatedness means recognizing the textual entailment between the hypothesis and the premise [152]. The general approach of modeling sentence pairs (i.e., measuring the relatedness between sentences) using neural networks includes two steps: represent both of the sentences as vectors via a sentence encoder and then initializing a classifier with these vectors to do the classification. The sentence encoder can be viewed as a compositional function which maps a sequence of words in a sentence to a vector. Some of the common compositional functions are sequential LSTM [270], Tree-LSTM [227, 270] and CNN [89].

In this chapter, we propose two models to encode attention inside tree structured LSTM cells and verify their effectiveness by evaluating them on the semantic relatedness task where the model needs to give a score depending on how similar two sentences are. The tree data structure allows a set of dependents in the dependency tree or constituents in the constituency tree to be children of an immediately higher level (parent) tree node. Our tree attention model

applies attention over the set of children in a subtree and decides which of them are important to reconstruct their parent node vector. We apply this attention with respect to four pieces of information: the vector representation of the sentence currently being represented as a tree, the vector representation of the sentence being compared with, dependent vectors (dependency tree) or phrase vector (binary constituency tree), and concatenated vectors of the dependents or the constituents. Our extensive evaluation proves the effectiveness of our attentive Tree-LSTM with respect to the plain Tree-LSTM models as well as some top performing models on the benchmark dataset.

## 4.2 Related work

Socher et al. [213] propose a number of recursive neural network (rNN) based models which take phrases as input rather than entire sentences. Phrases are represented as a vector as well as a parse tree. Vectors for higher level nodes in the tree are computed using a tensor-based composition function. Their best model was Matrix Vector rNN (MVRNN) where each word is represented as a vector as well as a matrix. In this model the children in a subtree interact more through their vectors rather than being influenced by some weights during the calculation of the parent’s vector and matrix representation.

Tai et al. [227] developed two different variants of standard linear chain LSTMs: child sum Tree-LSTM and N-ary Tree-LSTM. The underlying concept of using input, output, update and forget gates in these variants is quite similar to how these gates are used in standard LSTMs, however there are few important changes. The standard LSTM works over the sequence data whereas these variants are compatible with tree structured data (constituency tree or dependency tree). Also, unlike standard LSTMs, the hidden and cell states of a word at the current time step does not depend on the entire sequence seen before. Instead, the hidden and cell state of a parent node depends only on its children hidden and cell states. Recently, Chen et al. [50] combined LSTM with Tree LSTM for natural language inference task and empirically proved that these two models complement each other very well.

Zhou et al. [270] extend the concept of standard Tree-RNNs and propose a number of attention based Tree-RNN models to perform the semantic relatedness task. Their insight was quite novel: in order to compute the semantic similarity of two sentences, one can encode attention in the tree structure of one sentence with respect to the vector representation of the other sentence. However, their proposed attention model only works with child sum Tree-LSTMs and GRUs. Attention with Tree LSTM has also been studied by Liu et al. [140] for text summarization task where they use two different kinds of alignment : block alignment for aligning phrases and word alignment for aligning inter-words within phrases.

Turning to machine translation, the attention mechanism is used to align the source and target sentences in the decoding phase. More formally, the attention mechanism allows the model to attend to some elements with the intention of emphasizing different elements. The well-known attention models from [24] and [144] use recurrent models to attend over a set of source words during the generation of each target word. Using recurrent models to generate an attention score incorporates a memory mechanism inside the network which helps the model at run time to traverse and decide what to attend over. Also, this recurrency allows some positional information in the sequence to help ordering the generated words.

Parikh et al. [176] propose a decomposable attention model for natural language inference tasks by removing the modules with recurrent behavior during the calculation of attention. First, they pick a single vector from a set of vectors representing the source sentence and then compare its point-wise similarity with every element of each word vector from the target sentence. Following this, they compare these alignments using a function which is a feed forward neural network and finally perform an aggregation through summation before doing the final classification. Gehring et al. [80] propose a sequence to sequence learning framework utilizing a convolutional neural network which completely avoids recurrent models allowing their architecture to be parallelizable. In order to capture the positional information, they include a positional embedding layer which gives their model a sense of the portion of the sequence in the input or output it is currently dealing with. They encode *sine* and *cosine* frequencies for each dimension of every position in the sentence to create the positional embeddings and finally combine them with word embeddings. Vaswani et al. [240] combine the previous two works and propose a powerful machine translation framework utilizing attention without recurrence and positional embeddings. They also extend the decomposable attention mechanism by attending over the input sequence multiple times stating it as a multi-head attention where the target is to extract different features by different attentional heads.

## 4.3 The Model

In this section, we describe our work in detail. We first explain how the two variants of Tree-LSTM work. Following this, we describe our universal attention mechanism that is applicable for these two Tree-LSTM variants. Additionally, we give an in-depth analysis of adding this attention with respect to various information as discussed in Section 4.2

Recurrent neural networks (RNNs) are the best known and most widely used neural network (NN) model for sequence data as they sequentially scan the entire sequence and generate a compressed form of it. Although in theory RNNs are capable of remembering long distance dependencies, practically, as the sequence becomes longer, RNNs suffer from the vanishing

gradient problem [34, 177]. To overcome this drawback some RNN variants have been introduced such as Long Short Term Memory (LSTM) [92] and Gated Recurrent Unit (GRU) [53]. These variants use a gating mechanism to propagate new information further and at the same time to forget some previous information allowing the gradients to propagate further.

### 4.3.1 Incompatibility of standard LSTM and Tree structured data

Recurrent neural networks (RNNs) are the best known and most widely used neural network (NN) model for sequence data as they sequentially scan the entire sequence and generate a compressed form of it. Although in theory RNNs are capable of remembering long distance dependencies, practically, as the sequence becomes longer, RNNs suffer from the vanishing gradient problem [34, 177]. To overcome this drawback some RNN variants have been introduced such as Long Short Term Memory (LSTM) [92] and Gated Recurrent Unit (GRU) [53]. These variants use a gating mechanism to propagate new information further and at the same time to forget some previous information allowing the gradients to propagate further. Performance-wise, LSTMs are superior to GRUs because they have more parameters but in terms of computational complexity GRUs often surpass LSTMs.

Even though these gating variants effectively solve the RNN vanishing gradient problem, they are limited to linear data; however, a natural language sentence encodes more than a sequence of words. This extra information is usually represented in a tree structure. The tree structure shows how the words combine through different sub-phrases to reflect the overall meaning. If a sentence gets traversed by a standard LSTM, the latter part of the sentence gets more importance comparatively as the traversal moves left to right. But if the tree structure of the sentence gets traversed from bottom to top then the information from different constituent or dependents first gets combined to represent the root at the upper level and then this roots gradually gets traversed as children and combined to represent the root at next level and so on. So in both cases an LSTM cell will forget previous information which for plain LSTM, is related to the length of the sentence and for Tree-LSTM, is related to the depth of the tree. Also in plain LSTM, the hidden and cell state of a word at time step  $t$  depends on hidden and cell state of all the words from time step  $1 \dots t - 1$ . But in Tree-LSTM, the hidden and cell state of a root word depends only on the hidden and cell state of all of its children rather than all the words before it.

### 4.3.2 Tree-LSTM

There are two possible tree representations of a sentence: Dependency tree and Constituency tree [49]. As previously presented, the standard linear chain LSTM and BLSTM cannot cor-

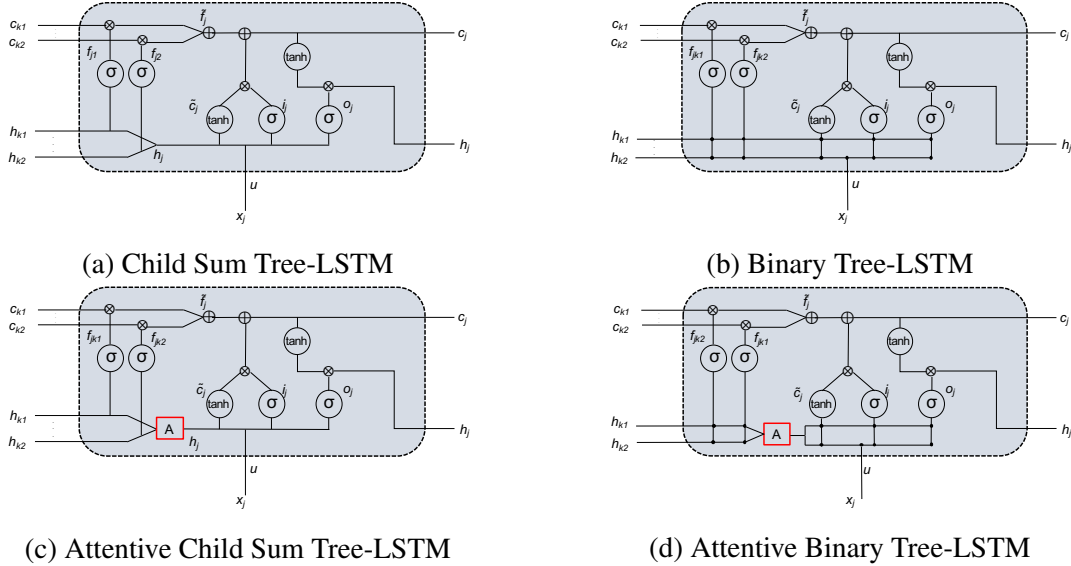


Figure 4.1: Illustrations of different Tree-LSTM architectures

rectly analyze this structured information. To properly deal with this structured data, Tai et al. [227] proposed two LSTM models which can analyze a tree structure preserving every property of the standard LSTM gating mechanisms. They called the first one child sum Tree-LSTM and the second one N-ary Tree-LSTM. Child sum Tree-LSTM fits well with dependency trees as it is well suited for high branching child-unordered trees. On the other hand N-ary Tree-LSTM (with  $n = 2$ ) works better with the binarized (Chomsky Normal Form) constituency trees.

Traditional LSTM generates a new hidden and cell state from the previous hidden state  $h_{t-1}$ , previous cell state  $c_{t-1}$  and current sequential input  $x_t$ . In the child sum Tree-LSTM, a component node state is generated based on the states of its children in the tree as shown in Figure 4.1(a). To do this, the internal gates (i.e., the input, output and intermediate cell states) are updated using the sum of the hidden states of the children of the component node as follows:

$$\tilde{\mathbf{h}}_j = \sum_{k \in C(j)} h_{jk} \quad (4.1)$$

where  $C(j)$  denotes the children of node  $j$ . Next, using this modified hidden state,  $\tilde{\mathbf{h}}$ , the input, output and intermediate cell states are calculated as follows:

$$\mathbf{i}_j = \sigma(\mathbf{W}^{(i)} x_j + \mathbf{U}^{(i)} \tilde{\mathbf{h}}_j + \mathbf{b}^{(i)}) \quad (4.2)$$

$$\mathbf{o}_j = \sigma(\mathbf{W}^{(o)} x_j + \mathbf{U}^{(o)} \tilde{\mathbf{h}}_j + \mathbf{b}^{(o)}) \quad (4.3)$$

$$\tilde{\mathbf{c}}_j = \tanh(\mathbf{W}^{(c)}x_j + \mathbf{U}^{(c)}\tilde{\mathbf{h}}_j + \mathbf{b}^{(c)}) \quad (4.4)$$

where  $W^{(i)}$ ,  $W^{(o)}$ ,  $W^{(c)}$ ,  $U^{(i)}$ ,  $U^{(o)}$ ,  $U^{(c)}$ ,  $b^{(i)}$ ,  $b^{(o)}$ , and  $b^{(c)}$  are the parameters to be learned. Instead of having just a single forget gate, child sum Tree-LSTMs have  $k$  forget gates where  $k$  is the number of children of the target node. This multiple forget gate allows child sum Tree-LSTM to incorporate individual information from each of the children in a selective manner. Each forget gate is calculated as follows:

$$\mathbf{f}_{jk} = \sigma(\mathbf{W}^{(f)}x_j + \mathbf{U}^{(f)}h_{jk} + \mathbf{b}^{(f)}) \quad (4.5)$$

Next, the individual forget gate outputs are multiplied with corresponding cell state values and then combined to get a single forget vector which is used to get the final cell state of the model as follows:

$$\tilde{\mathbf{f}}_j = \sum_{k \in C(j)} f_{jk} \cdot c_k \quad (4.6)$$

$$\mathbf{c}_j = i_j \cdot \tilde{\mathbf{c}}_j + \tilde{\mathbf{f}}_j \quad (4.7)$$

Finally, the update equation for the hidden state of a child sum Tree-LSTM cell is similar to the traditional LSTM:

$$\mathbf{h}_j = o_j \cdot \tanh(c_j) \quad (4.8)$$

Each of the parameter matrices represents a correlation among the component vector, input  $x_j$  and the hidden state  $h_k$  of the  $k^{th}$  child of the component unit. For example, the sigmoid function at the input gate represents semantically important words at input by giving values close to 1 (e.g., a verb) and relatively unimportant words by giving values close to 0 (e.g., a determiner). Since the hidden state and cell state values of the parent node are generated based on the hidden state and the cell state of its children, child sum Tree-LSTM is well suited for dependency trees.

The N-ary Tree-LSTM is used where there are at most  $N$  ordered children. Unlike the child sum Tree-LSTM, it has a different set of parameters for each child having its own cell and hidden state, shown in Figure 4.1(b). The update equations for deriving input, output and update gate values are as follows:

$$\mathbf{i}_j = \sigma(\mathbf{W}^{(i)}x_j + \sum_{l=1}^N \mathbf{U}_l^{(i)}h_{jl} + \mathbf{b}^{(i)}) \quad (4.9)$$

$$\mathbf{o}_j = \sigma(\mathbf{W}^{(o)}x_j + \sum_{l=1}^N \mathbf{U}_l^{(o)}h_{jl} + \mathbf{b}^{(o)}) \quad (4.10)$$



$$\tilde{\mathbf{c}}_j = \tanh(\mathbf{W}^{(c)}x_j + \sum_{l=1}^N \mathbf{U}_l^{(c)}h_{jl} + \mathbf{b}^{(c)}) \quad (4.11)$$

where  $W^{(i)}$ ,  $W^{(o)}$ ,  $W^{(c)}$ ,  $U_l^{(i)}$ ,  $U_l^{(o)}$ ,  $U_l^{(c)}$ ,  $b^{(i)}$ ,  $b^{(o)}$ , and  $b^{(c)}$  are the parameters to be learned. As can be seen, for each gate, the N-ary Tree-LSTM has a set of  $N$  parameter matrices associated with the  $N$  hidden states whereas the child sum Tree-LSTM has just one. Next, for each of the children, forget gate values are calculated separately, as done in the child sum Tree-LSTM as follows:

$$\mathbf{f}_{jk} = \sigma(\mathbf{W}^{(f)}x_j + \sum_{l=1}^N \mathbf{U}_{kl}^{(f)}h_{jl} + \mathbf{b}^{(f)}) \quad (4.12)$$

Similar to the child sum Tree-LSTM, these new forget gate values are multiplied with corresponding cell state values and then summed to get the final values for the forget gate:

$$\tilde{\mathbf{f}}_j = \sum_{l=1}^N f_{jl} \cdot c_{jl} \quad (4.13)$$

Finally, the cell state and new hidden state values are updated using Equations [4.7](#) and [4.8](#)

### 4.3.3 Attention

The two tree structured LSTM models described in Section [4.3.2](#) treat every word within a sub-tree with equal probability. More specifically, in an N-ary Tree-LSTM, every word contributes uniformly to the building of the higher-level constituent. Likewise, the child sum Tree-LSTM architecture suggests that, within a dependency tree branch, a head word influences all of its dependent words in a similar way. When viewing the tree as a semantic representation of a sentence, this may not be the case in many scenarios. For a constituency tree, if a sub-tree contains some negative sentiment words, then it is not always the case that the sentiment of that particular constituent is negative. If the negative sentiment word is preceded by a negation, then the higher-level constituent becomes semantically positive because of the location of the negation word. To capture this type of information, attention is applied over the sub-tree components to apportion the importance of each sub-tree component when building the entire tree either semantically or syntactically. In this study, we are interested in applying semantic attention over the sub-tree components to see how they contribute to building a sub-tree.

Attentive Tree-LSTM was proposed by [\[270\]](#) for doing the semantic relatedness task. They state that the effect of semantic relevance could be implemented as part of the sentence representation construction process using a Tree-LSTM where each child should be assigned a different weight. In their proposed model, a soft attention mechanism assigns an attention

weight on each child in a subtree. Given a collection of hidden state  $h_1, h_2, \dots, h_n$  and an external vector  $s$ , their proposed attention mechanism assigns a weight  $\alpha_k$  on each of these hidden states and produces a weighted vector  $g$ . To achieve this, first they perform an affine transformation on each of the child hidden states and calculate a vector  $m_k$  as follows:

$$m_k = \tanh(\mathbf{W}^{(m)}h_k + \mathbf{U}^{(m)}s), \quad (4.14)$$

where  $\mathbf{W}^{(m)}$  and  $\mathbf{U}^{(m)}$  are the parameter matrices of size  $d \times d$  and  $s$  is the vector representation of the sentence learned by a sequential LSTM. Next, using this transformed hidden states  $m_k$ , the attention probabilities  $\alpha_k$  are calculated as follows

$$\alpha_k = \frac{\mathbf{w}^T m_k}{\sum_{j=1}^n \mathbf{w}^T m_j} \quad (4.15)$$

where  $w$  is a parameter vector of size  $1 \times d$ . Following this, a weighted combination of the hidden states is calculated using,

$$g = \sum_{1 \leq k \leq n} \alpha_k h_k \quad (4.16)$$

This  $g$  is of size  $1 \times d$ . Finally, an affine transformation is applied on this  $g$  to get the new hidden state  $\tilde{h}$  as follows:

$$\tilde{h} = \tanh(\mathbf{W}^{(a)}g + \mathbf{b}^{(a)}) \quad (4.17)$$

This soft attention mechanism from [270] introduces four new parameters to derive the final attentive hidden state; two matrices in Eqn. 4.14 one vector in Eqn. 4.15 and one matrix in Eqn. 4.17. This attention mechanism is only applicable to the child sum Tree-LSTM. It is not possible to apply this attention on N-ary Tree-LSTMs since the structure of the N-ary Tree-LSTM is such that it needs  $N$  separate hidden states to work with whereas a child sum Tree-LSTM collapses all the hidden states to a single vector through summation. In this study, we develop two generalized attention models by adopting the decomposable attention framework proposed by [176] and the soft attention mechanism proposed by [270].

**Model 1:** Our first model is based on the self attention mechanism where we make some subtle changes to calculate the attention probability with respect to different segments of the sentence. Calculating attention in this way involves three matrices *key*, *query*, and *value*. The *key* matrix represents on which child to attend over, the *query* matrix represents “with respect to what” is attention to be applied and the *value* matrix extracts the final attention-able vector using attention probability. The *key* matrix is calculated as follows:

$$key = \mathbf{W}^{(k)}M^{(k)} \quad (4.18)$$

where,  $W^{(k)}$  is a parameter matrix of size  $d \times d$  and  $M^{(k)}$  is the matrix on which to attend over. For child sum Tree-LSTMs, this matrix is the concatenation of the vectors of all the words under a particular head word. For N-ary Tree-LSTMs, it is the concatenation of all the word vectors in a constituent. So in both cases the formal representation is  $M^{(k)} = [h_1; h_2; \dots; h_n]$ . In order to encode self attention in the sub-tree, the *query* and *value* matrices also get calculated with respect to  $M^{(k)}$  ( $M^{(k)} = M^{(q)} = M^{(v)}$ ) but with a different set of parameter matrices  $W^{(q)}$  and  $W^{(v)}$  as follows:

$$query = W^{(q)} M^{(q)} \quad (4.19)$$

$$value = W^{(v)} M^{(v)} \quad (4.20)$$

Once the *key* and *query* get calculated, the next step is to align each of them by looking at the similarity at each dimension of their representation. This is done using:

$$align = (query)^T key \cdot \frac{1}{\sqrt{d}} \quad (4.21)$$

where the *align* matrix is of size  $n \times n$  with  $n$  representing the number of children within this sub-tree. The  $d$  is being used here as a normalizing factor. Finally, the attention probability is calculated by applying *softmax* over it as follows:

$$\alpha = \text{softmax}(align) \quad (4.22)$$

Here  $\alpha$  is the matrix of attention probabilities where each row represents how much attention needs to be given on each of the children within that sub-tree according to the word at that row. As there are  $n$  children within a sub-tree, the size of this matrix is  $n \times n$ . Finally, we calculate a new attention encoded hidden state  $\tilde{h}$  through a batch-wise matrix multiplication between the  $\alpha$  and *value* matrices as follows:

$$\tilde{h} = \text{bmm}(\alpha, value) \quad (4.23)$$

The shape of this new  $\tilde{h}$  is  $n \times d$ . It contains attention encoded hidden state values of all the children sequentially one on top of another. So in order to locate a specific hidden state value, the row number corresponding to the position of that child in the sub-tree is used. For child sum Tree-LSTMs, all of the hidden state vectors are summed to get a single vector and for N-ary Tree-LSTMs, one row of  $\tilde{h}$  is selected as the hidden state of a child.

For the semantic relatedness task, where the objective is to assign a score based on the similarity between two sentences, it is better to calculate the query matrix with respect to the vector representation of the second sentence. Specially, given a pair of sentences, our generalized attentive encoder uses the representation of one sentence generated via a sequential

LSTM to guide the structural encoding of the other sentence on both the dependency as well as the constituency tree. In that case,  $M^{(q)}$  is a vector rather than a matrix thus changing the shape of *query* from Eqn. 4.19 into  $1 \times d$ . This results in an alignment vector from Eqn. 4.21 of size  $1 \times n$ . When `softmax` is applied over this vector, a vector of probabilities,  $\alpha$ , is produced. Finally, instead of doing a matrix multiplication as in Eqn. 4.23, a point-wise multiplication  $\tilde{h} = \alpha * \text{value}$  is performed resulting in a new hidden state vector. For child sum Tree-LSTMs, we use this new hidden state vector in place of the one generated in Eqn. 4.1 and for N-ary Tree-LSTMs, we use this hidden state vector as the hidden state of both the left and right children. This way of calculating self attention requires three additional matrices as parameters from Eqn. 4.18, 4.19 and 4.20, a smaller number of parameters than found in [270]. We further continue our experiments by calculating a phrase vector representation using an additional LSTM cell and use it as the *query* vector. Then, we adopt the same procedure as above to calculate the attention probability  $\alpha$  and the final hidden state vector  $\tilde{h}$ . However, this requires more parameters than what is required in [270].

**Model 2:** In our second model, we combine the concepts of decomposable attention mechanism with a soft attention layer. Here, we have two matrices *key* and *query* and their derivation are the same as Eqns. 4.18 and 4.19. We further align and transform these matrices into probabilities using the same set of equations, Equations 4.21 and 4.22. We again make some subtle changes which result in four different versions of this model. In Eqn. 4.19, when  $M^{(q)} = M^{(k)}$ , the dimension of the attention probability becomes  $n \times n$  and when  $M^{(q)}$  is either a sentence vector  $M^{(q)} = \text{LSTM}(\text{sentence}_2)$  or phrase vector  $M^{(q)} = \text{LSTM}(M^{(k)})$ , the dimension of this attention probability changes to  $1 \times n$ . Then,  $\tilde{h}$  is calculated as follows,

$$\tilde{h} = \begin{cases} \text{bmm}(\alpha, M^{(k)}), & \text{if } \alpha \text{ is a matrix} \\ \alpha * M^{(k)}, & \text{if } \alpha \text{ is a vector} \end{cases} \quad (4.24)$$

Next we perform an affine transformation of this  $\tilde{h}$  by multiplying it with a parameter matrix  $W$  and passing it through a `tanh` layer as follows:

$$\hat{h} = \tanh(W\tilde{h} + \mathbf{b}) \quad (4.25)$$

In the case of child sum Tree-LSTMs, if  $\hat{h}$  is a matrix, we do a summation of all the rows and use that as the final vector and if  $\hat{h}$  is a vector, we use that as it is. In the case of N-ary Tree-LSTMs, if  $\hat{h}$  is a matrix, then each row corresponds to the hidden state of a child and if  $\hat{h}$  is a vector, then we just copy this vector as the hidden states of the children.

Hyper-parameter	Range Selected
Learning rate	<b>0.01</b> / 0.025 / 0.05
Batch size	<b>10</b> / 25 / 30
Momentum	0.9
Memory dimension	150
MLP hidden dimension	50
Attention layer dimension	150
Dropout	0.5 / <b>0.2</b> / 0.1
Word embedding size	300
Gradient clipping	<b>5</b> / 20 / 50
Weight decay	$10^{-5}$
Learning rate decay	0.05

Table 4.1: Hyper-parameters used for the experiments (in boldface) and the ranges that were searched during tuning.

## 4.4 Experimental Setup and Analysis

In this section, we describe the detailed experimental setup for the evaluation of our study. We first explain the dataset statistics for evaluating our generalized attention frameworks. Following this, we explain the working environment details along with the hyper-parameter settings of our architecture.

We evaluated our model for the semantic similarity task on the Sentences Involving Compositional Knowledge (SICK) dataset [152]. The task is to give a likeness score for a pair of sentences and then compare it to a human produced score. The SICK dataset contains 9927 sentence pairs configured as: 4500 training pairs, 500 development pairs and 4927 test pairs. Each sentence pair is annotated with a similarity score ranging from 1 to 5. A high score shows that the sentence pair is strongly related. All sentences are derived from existing image and video comment datasets. The assessment measures are Pearson’s  $\rho$  and mean squared error (MSE).

Table 4.1 shows the detailed hyper-parameter settings of our model. We trained our model on an Nvidia GeForce GTX 1080 GPU with ‘Adam’, ‘SGD’ and ‘Adagrad’ optimizers. All of the results in the next section are reported using ‘Adagrad’ as it was giving the best results. The ‘Learning rate decay’ parameter was only used with the ‘SGD’ optimizer. We used PyTorch 0.4 to implement our model under the Linux environment.

Table 4.2 shows the overall evaluation of our model in terms of Pearson’s  $\rho$  and Mean Squared Error (MSE). This table also contains the results of some top performing models on the SICK dataset. Among these models, [227] and [270] did their evaluation with plain Tree-LSTMs, whereas the rest of the models use some different composition functions such as CNN [111], ECNU [264] and Combine-skip + COCO [89]. However [270] also experimented with

	Model		<b>r</b>	<b>MSE</b>
Previous Models	ECNU [264]		0.8414	—
	Combine-skip+COCO [111]		0.8655	0.2561
	ConvNet [89]		0.8686	0.2606
	Seq-GRU [270]		0.8595	0.2689
	Seq-LSTM [270]		0.8528	0.2831
	Dep. Tree-GRU [270]		0.8672	0.2573
	Dep. Tree-GRU + Attn. [270]		0.8701	0.2524
	Const. Tree-LSTM [227]		0.8582	0.2734
			0.8460 †	0.2895 †
	Dep. Tree-LSTM [227]		0.8676	0.2532
			0.8663 †	0.2612 †
	Dep. Tree-LSTM + Attn. [270]		0.8730	0.2426
			0.8635 †	0.2591 †
Child Sum Tree LSTM	Model 1	Self	0.7466	0.4545
		Sentence 1	0.7305	0.4849
		Sentence 2	0.7939	0.3801
		Phrase	0.7889	0.3877
	Model 2	Self	0.8577	0.2695
		Sentence 1	0.8620	0.2634
		Sentence 2	0.8686	0.2518
		Phrase	0.8623	0.2615
Binary Tree LSTM	Model 1	Self	0.8648	0.2567
		Sentence 1	0.8692	0.2486
		Sentence 2	0.8686	0.2507
		Phrase	0.8676	0.2517
	Model 2	Self	0.8698	0.2476
		Sentence 1	0.8698	0.2476
		Sentence 2	0.8720	0.2435
		Phrase	0.8696	0.2479

Table 4.2: Test set results on the SICK dataset. The first group lists previous results, and the remainder are the results of our models. We mark models that we re-implemented with a †.

attentive Tree-LSTMs and GRUs, but they have only been able to design models compatible with the child sum variant. On the other hand, among our two proposed models, Model 2 performs very well on both Tree-LSTM variants showing significant improvements with every configuration. For both child sum as well as binary Tree-LSTMs, our second model with cross sentence attention has superior performance compared to the plain Tree-LSTM variants getting MSE of 0.2518 and 0.2435 respectively. For the child sum Tree-LSTM, Model 1 performs poorly compared to all the other models. This poor performance is due to the hard attention that it applies. If a subtree has  $n$  children, this hard attention forces  $n - 1$  children to have probability close to 0 which causes the domination of just one child hidden state in the summation. The rest will not contribute at all. On the other hand, the reason behind Model 2 performing better

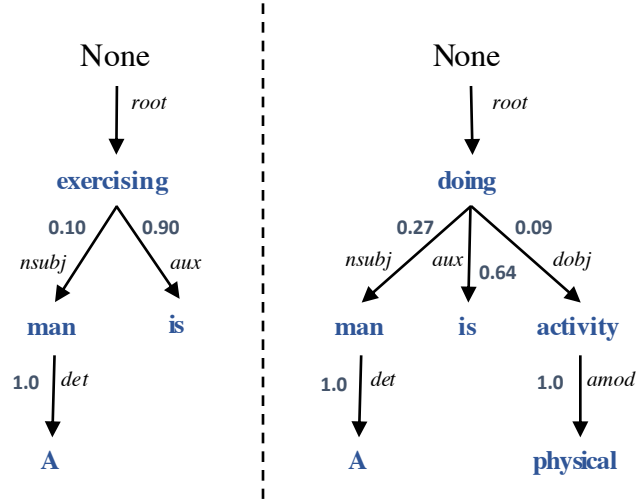


Figure 4.2: Probability of each node being selected by attentive child sum Tree-LSTM Model 2 with cross sentence attention (**Left**: *A man is exercising* **Right**: *A man is doing physical activity* **Label**: *Entailment* ).

in every configuration with both variants is that even though a hard attention causes one of the children to get close to 0, the normalization of N-ary tree into binary tree causes much more flexibility for the information to flow from bottom to top. During normalization, a branch with  $n$  children gets split up to  $n - 1$  full binary trees resulting in  $(n - 1)/2$  nodes that are always chosen. Our best performing attentive child sum Tree-LSTM model with cross sentence attention achieves a better result (0.2518 MSE) than the plain child sum tree variant from [227] (0.2532 MSE). Our score did not surpass the reported result (0.2426 MSE) of the attentive child sum variant from [270]. However, our implementation of their model with their reported hyper-parameters gave a 0.2591 MSE which is significantly worse than their claimed MSE. This suggests to us that the implementation environment has a strong impact on model performance. Our child sum Tree-LSTM Model 2 with cross sentence attention achieves better performance than our implementation of [270] using their hyper-parameter settings. To the best of our knowledge, our work is the first to encode attention inside a binary Tree-LSTM cell. In terms of binary tree LSTM, our best performing model with cross sentence attention achieves 0.2435 MSE which is significantly better than the one reported in [227] (0.2734 MSE) for the non-attentive version. In our implementation of plain binary Tree-LSTM without attention from [227] we were not able to reproduce their reported result and ended up with 0.2895 MSE which is much worse than the one we got with every configuration of our Model 1 and Model 2. This performance analysis does show the effectiveness of our generalized attention model.

Figure 4.2 depicts the probability assigned to each node in the dependency tree by our Model 2 with cross sentence attention. Unlike standard child sum Tree-LSTM, where the hid-

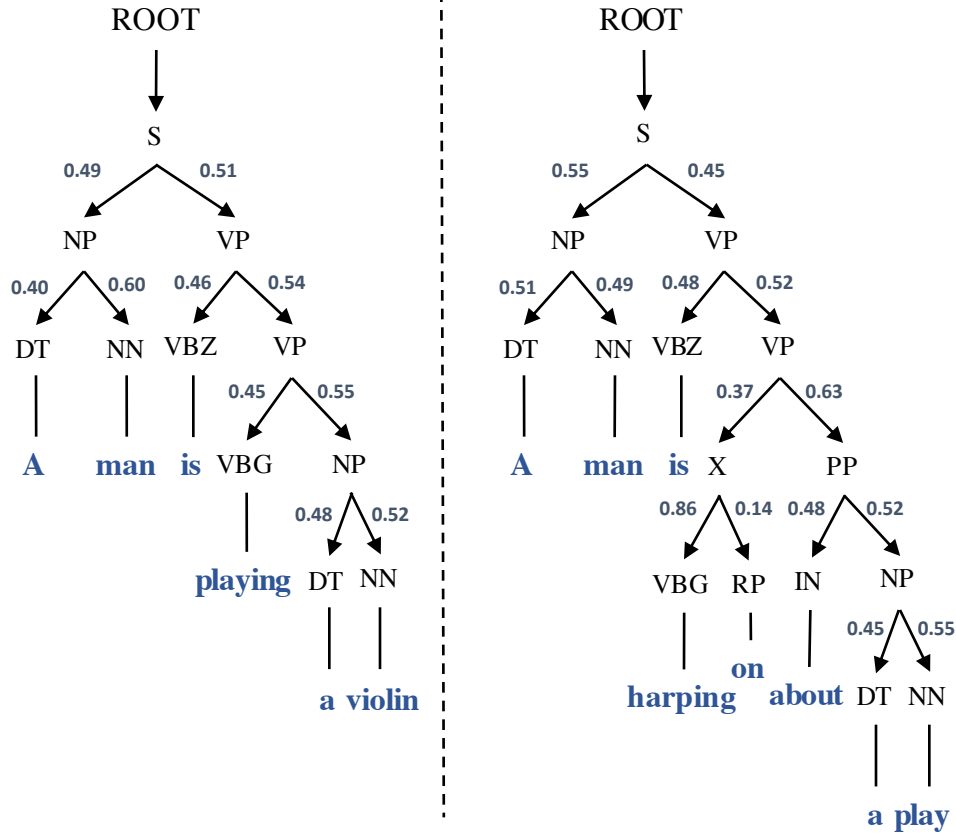


Figure 4.3: Probability of each node being selected by attentive binary Tree-LSTM Model 2 with cross sentence attention (**Left:** *A man is playing a violin* **Right:** *A man is harping on about a play* **Label:** *NEUTRAL* ).

den states of all the children nodes are combined with a plain summation, our attentive child sum Tree-LSTM assigns a weight to each node and then does a weighted summation. The example used in this Figure has “A man is exercising” as the left sentence, “A man is doing physical activity” as the right sentence and “Entailment” as their relationship. As usual, the main verb from both of the sentences is selected as the *root* node. The auxiliary verb (*is*) gets high attention in both the left and right trees because of the word similarity. However, their absolute influence varies because of the presence of semantically related words in other branches as discussed above. Both of these trees share the same nominal subject (*nsubj*) however with different probabilities (in the left tree its probability is significantly lower). The reason behind this is the cross sentence attention allows the word *man* from the left sentence to align with two words *man* and *physical* from the right sentence. As they share a similar semantic meaning in the vector space, the branch in the left sentence that contains *man* is diminished because the right sentence divides the attention between two branches (left sentence: *exercising*  $\xrightarrow{nsubj}$  *man*; right sentence: *doing*  $\xrightarrow{nsubj}$  *man* and *doing*  $\xrightarrow{dobj}$  *activity*).



Figure 4.3 depicts the probability assigned to each node in a binary (Chomsky Normal Form) constituency tree using an attentive binary Tree-LSTM with cross sentence attention. In this setting, the attention on the structure of the left sentence is computed with respect to the vector representation of the right sentence and vice versa. As a result, the words in a specific phrase from the left sentence are aligned with very high attention probability if the same words appear anywhere in the right sentence. However, as softmax was operating with small values from Eqn. 4.21, it forced both children to have the same probabilities (0.5). In order to verify whether this probability has any effect or not, we have confirmed that replacing  $\alpha$  in Eqn. 4.23 with pairs of the same value other than (0.5) results in the model giving comparatively poor performance. Finally, for the inference of attention probabilities, we replaced softmax from Eqn. 4.22 with plain normalization. For the example in Figure 4.3, we have “A man is playing a violin” as the left sentence, “A man is harping on about a play” as the right sentence and “Neutral” as their relationship. The phrase *NP* gets almost the same probabilities in both the left (0.49) and right (0.55) trees because of having the same set of words: “A man”. The subphrase *VBZ* under *VP* in both trees gets very high attention due to having the same word “is” at exactly the same position. Due to the Chomsky normalization, the tree on the right side gets an extra dummy node *X* which contains *VBG* and *RP* as the child nodes. In the vector space, the words “playing” and “harping” are semantically connected which allows both of the models to align them with moderately high as well as equal probabilities. The left tree does not have any particle (“*RP*”) words which causes the model to put low attention probability when it appears on the right tree. The left tree has *NP* as the right child of *VP* at level 3 with probability 0.55 which is quite close to the amount of attention *PP* gets (0.63) as the right child of *VP* at the same level in the right tree. Again in both of these trees, at the right most branch, the words “play” and “violin” share the same semantic space which causes them to get aligned with almost the same probabilities. The *DT* in this branch gets the same high probability because of appearing in both sentences at relatively similar positions.

## 4.5 Conclusion

Previous attempts to encode the attention mechanism in Tree-LSTMs were only successful for the child-sum tree variant as the techniques used are not easily adaptable to binary trees like the Chomsky Normal Form constituency tree. In this chapter, we have introduced two different ways of applying attention on tree structures. The second of these two methods gives superior performance for both tree variants. The proposed techniques can be used on both dependency as well as constituent tree structure. Our experimental results verify the superiority of the attentive variant of Tree-LSTMs over traditional Tree-LSTMs and linear chain LSTMs on the

semantic relatedness task. With our extensive in depth analysis, we showed that our proposed attention models provide a good representation of how a sentence builds semantically from the words. Our generalized attention framework is adaptable to any tree like structures.

## Chapter 5

# Identifying Protein-Protein Interaction using Tree LSTM and Structured Attention

This chapter is based on the paper titled “Identifying Protein-Protein Interaction using Tree LSTM and Structured Attention” co-authored with Jumayel Islam, Muhammad Rifayat Samee and Robert E. Mercer that appeared in the 13th International Conference on Semantic Computing (ICSC 2019) [6].

Identifying interactions between proteins is important to understand underlying biological processes. Extracting a protein-protein interaction (PPI) from the raw text is often very difficult. Previous supervised learning methods have used handcrafted features on human-annotated data sets. In this chapter, we propose a novel tree structured long short term memory network with structured attention architecture for doing PPI. Our architecture achieves state-of-the-art results (precision, recall, and F-score) on the AIMed and BioInfer benchmark data sets. Moreover, our models achieve a significant improvement over previous best models without any explicit feature extraction. Our experimental results show that traditional recurrent networks have inferior performance compared to tree recurrent networks for the supervised PPI problem.

### 5.1 Introduction

With extensive ongoing research currently happening in the bio-medical field, there is an exponentially growing amount of information available in textual form requiring expert knowledge to extract the important information contained therein. As doing this manually with human expertise only is time consuming and expensive, there has been a lot of interest in develop-

ing computational approaches for automatically inferring some hidden information from this vast source of knowledge such as protein-protein interactions (PPIs), drug-drug interactions (DDIs) and chemical-disease relation information. Researchers have successfully applied natural language processing (NLP) techniques and machine learning (ML) methods for doing these tasks [95,127,182,205].

The task of identifying protein-protein interactions (PPIs) is to extract relations between protein entities mentioned in a document [117]. While PPI relations can cross over sentences and even across corpora, current work is centered mostly on PPIs in single sentences [185,231]. For example, in the sentence “*LEC induces chemotaxis and adhesion by interacting with CCR1 and CCR8.*”, *LEC-CCR1* and *LEC-CCR8* are in PPI relations, whereas there is no relation between *CCR1* and *CCR8*.

Whereas previously, pattern-based methods have been very popular for doing this bio-medical relation extraction, in this chapter, we propose a novel neural net architecture for identifying protein-protein interactions from bio-medical text using a Tree LSTM [227] with Structured Attention [234]. We provide an in depth analysis of traversing the dependency tree of a sentence through a child sum tree LSTM and at the same time learn this structural information through a parent selection mechanism by modeling non-projective dependency trees. We also provide an extensive evaluation of our model by doing a detailed comparison with the currently available state of the art methods applied on the standard PPI corpora (AImed, BioInfer, IEPA, HPRD50, and LLL). Our architecture achieves state of the art results on four of the five corpora. Our experiments suggest that our model is more generalized and is better capable of capturing long distance information than existing feature and kernel based methods.

## 5.2 Related Work

In previous work, pattern-based methods have been very popular for doing PPI relation extraction, where patterns as well as rules were crafted and defined based on lexical and syntactic features [61,129,199]. For example, Leeuwenberg et al. [129] propose the syntactic tree pattern structure (STPS) for DDI extraction from a sentence in bio-medical text based on the syntax tree of the sentence. Also much research has been done on bio-medical relation extraction using Kernel-based methods which allow learning rich structural data in the form of syntactic parse trees and dependency structures [15,48,108,158]. Miwa et al. [158] propose a system which embeds rich feature vectors in a Support Vector Machine with corpus weighting where the weights are learned from one corpus and the other corpora are used for support. Kim et al. [108] propose a walk-weighted sub-sequence kernel for the extraction of PPIs. It captures the non-contiguous syntactic structures by matching the v-walk and e-walk on the

shortest dependency path. Chang et al. [48] propose an interaction pattern tree kernel method in which they extract PPIs by integrating the PPI patterns with a convolution tree kernel. Airola et al. [15] propose a method to extract PPIs by looking at the information from both dependency as well as linear subgraphs. For this they adopted an all-path kernel approach where they weighted all the edges on the shortest paths by a high value and all other edges with a low value. Peng et al. [180] propose an Extended Dependency Graph (EDG) based approach by incorporating a few simple linguistic features beyond syntax information. Finally they evaluated this EDG approach with edit distance and an APG kernel on the five benchmark corpora. Zhang et al. [260] propose a neighborhood hash kernel based method for PPI extraction. They started by transforming each node label of the dependency graph for two target sentences into a bit label and then replaced this bit label by a new label produced by order-independent logical operations on the bit labels of the current node and its neighboring nodes. They continued this process for the two target sentences and finally ended up with a high order substructures over the dependency graph. Finally, they computed the similarity of the two dependency graphs based on the intersection ratio of the updated label sets.

Recently, deep neural network (DNN) based methods have successfully applied and achieved promising results in bio-medical relation extraction from bio-medical literature [139, 187, 257, 266]. Mikolov et al. [156] propose an approach which gives a distributed representation (i.e., embeddings) of words capturing both the syntactic and the semantic similarity. Nowadays almost all of the DNN-based approaches in linguistics have this embedding layer at the top either in a pre-trained or randomly initialized form.

Peng et al. [181] adopt a convolutional neural network (CNN) based approach in which they utilize two channels of CNN for high level feature extraction. In one channel, they use raw words along with some syntactic features such as parts-of speech, chunk parsing information, named entities, syntactic dependencies and two distance vectors for each word representing the distance from the word to the two proteins being considered as interacting. In another channel they use parent word information for each word and pass this to an embedding layer to get a distributed representation of the sentence in terms of parent words. Following this, they apply convolution on these two channels separately and map them via a fully connected layer to the required number of classes.

Zhang et al. [261] also utilize a multi-channel CNN for this task. In the first channel, they use a sequence of raw words along with positional embedding features. In the second channel, they use shortest dependency path information: the arcs visited in the shortest paths from the first protein to the root and the second protein to the root. This information is arranged as a sequence and passed to an embedding layer. In the third channel, they use dependency relation embedding, storing the embedding of the words encountered in the shortest dependency path.

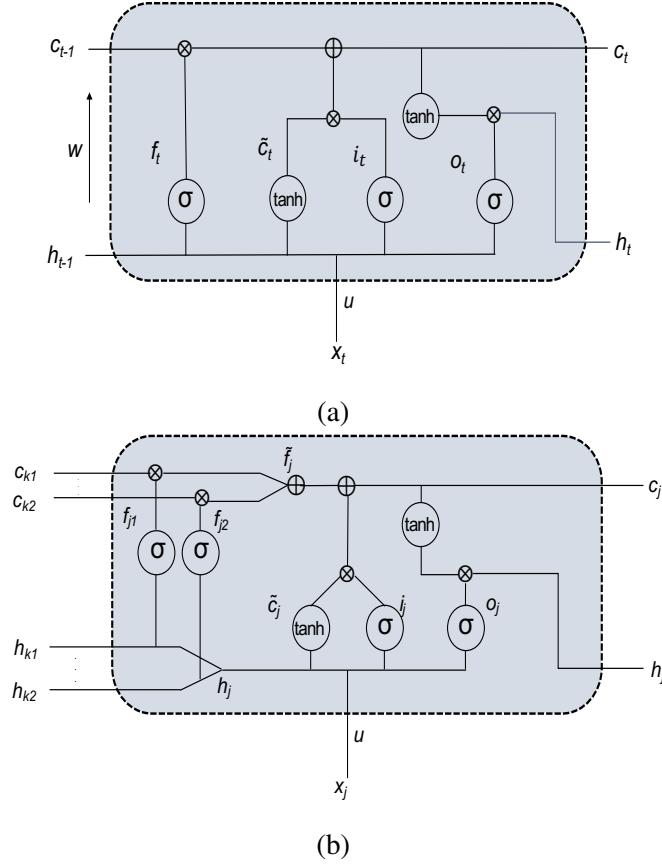


Figure 5.1: Standard LSTM (a) vs Tree LSTM (b)

Finally they apply convolution on it and map the extracted features to two classes using multi-layer perception (MLP) followed by a softmax layer.

Zhao et al. [266] propose a greedy layer-wise unsupervised learning-based approach to extract PPIs from bio-medical literature. They first divided their corpus into train, validation and unlabelled set and applied an auto encoder (AE) on the unlabelled set of data to initialize the parameters of a deep multi-layer neural network. Finally they applied a gradient descent method using back propagation to train their whole model.

Hsieh et al. [93] utilize recurrent neural network (RNN) for extracting the PPI form bio-medical literature. Without doing any additional feature extraction, they used long short term memory (LSTM) (a variant of RNN) to encode the dependency information through time from forward and backward direction over the sentence. Finally, they took the left most and right most output vector of LSTM, concatenated them and applied MLP followed by softmax for the classification.

## 5.3 The Model

In this section, we describe our work in detail. We first explain the working mechanism of a tree LSTM cell. Then we explain the Structured Attention mechanism for learning the dependency tree through Kirchhoff's Matrix-Tree Theorem. Finally we explain how we combine the tree LSTM architecture with structured attention to obtain a performance boost that we describe in the next section.

### 5.3.1 Recurrent unit: Bidirectional LSTM

In this chapter, we use recurrent neural network (RNN) which is the best known and most widely used NN model for sequence data. Its long-short term memory variant (LSTM) gives all of the advantages of the basic RNN with an elegant solution to RNN's vanishing gradient problem. Figure 5.1(a) shows a sample LSTM cell and the construction of its internal gates are as follows:

$$\begin{aligned}
 \mathbf{i}_t &= \sigma(\mathbf{W}^{(i)}x_t + \mathbf{U}^{(i)}h_{t-1} + \mathbf{b}^{(i)}) \\
 \mathbf{o}_t &= \sigma(\mathbf{W}^{(o)}x_t + \mathbf{U}^{(o)}h_{t-1} + \mathbf{b}^{(o)}) \\
 \mathbf{f}_t &= \sigma(\mathbf{W}^{(f)}x_t + \mathbf{U}^{(f)}h_{t-1} + \mathbf{b}^{(f)}) \\
 \tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}^{(c)}x_t + \mathbf{U}^{(c)}h_{t-1} + \mathbf{b}^{(c)}) \\
 \mathbf{c}_t &= \mathbf{i}_t \cdot \tilde{\mathbf{c}}_t + \mathbf{f}_t \cdot \mathbf{c}_{t-1} \\
 \mathbf{h}_t &= \mathbf{o}_t \cdot \tanh(\mathbf{c}_t)
 \end{aligned} \tag{5.1}$$

Although LSTMs are very good with sequence data, most often it is important to have information from the past as well as from the future. However, LSTM allows only one hidden state from the past and changes that hidden state recursively through time. An elegant resolution to this problem is going over the sequence in both forward and backward directions using two hidden states and finally concatenating the output from both directions. This method, called Bidirectional LSTM (BLSTM), has proven to be very effective in some prior works [73, 84, 229]. BLSTM has the same internal structure as LSTM except one of the output dimensions is twice that of the LSTM output.

### 5.3.2 Tree LSTM

The main limitation of the basic LSTM is that it can only be used for analyzing sequential information. However, a natural language sentence encodes more than a sequence of words. This extra information is usually represented in a tree structure. One such structure is the dependency tree [49]. LSTM and BLSTM cannot analyze this structured information correctly.

A variant of standard LSTM cell, called tree LSTM (tLSTM) [227], traverses the sentence by following a tree-structured network topology rather than going over the sequence as a linear chain. The underlying idea of an LSTM cell remains the same except here each tLSTM unit is capable of incorporating information from multiple child units as well. Figure 5.1(b) shows a sample tLSTM cell. In this study, we use child sum version of tree LSTM, as it is more suitable with dependency trees.

Traditional LSTM takes the previous hidden state  $h_{t-1}$ , the previous cell state  $c_{t-1}$  and the current time step input  $x_t$  into account and generates a new hidden state and cell state. However in the child sum tree LSTM, the main gist remains the same except component node states are now generated based on the states of its all possible children in the tree structure. To do this, first, the hidden states at the previous time step is summed up for all of the children of the component node and the internal gates (i.e., input, output and intermediate cell state) are updated using this new hidden state.

$$\tilde{\mathbf{h}}_j = \sum_{k \in C(j)} h_{jk} \quad (5.2)$$

where  $C(j)$  denotes the set of children of node  $j$ . Next using this modified hidden state  $\tilde{\mathbf{h}}$ , input, output and intermediate cell states are calculated as follows,

$$\mathbf{i}_j = \sigma(\mathbf{W}^{(i)}x_j + \mathbf{U}^{(i)}\tilde{\mathbf{h}}_j + \mathbf{b}^{(i)}) \quad (5.3)$$

$$\mathbf{o}_j = \sigma(\mathbf{W}^{(o)}x_j + \mathbf{U}^{(o)}\tilde{\mathbf{h}}_j + \mathbf{b}^{(o)}) \quad (5.4)$$

$$\tilde{\mathbf{c}}_j = \tanh(\mathbf{W}^{(c)}x_j + \mathbf{U}^{(c)}\tilde{\mathbf{h}}_j + \mathbf{b}^{(c)}) \quad (5.5)$$

where  $W^{(i)}$ ,  $W^{(o)}$  and  $W^{(c)}$  are the parameters to be learned. Instead of having just a single forget gate, tLSTMs have  $k$  forget gates where  $k$  is equal to the number of children of the target node. This multiple forget gate allows tLSTM to incorporate individual information from each of the children in a selective manner. Each forget gate is calculated as follows:

$$\mathbf{f}_{jk} = \sigma(\mathbf{W}^{(f)}x_j + \mathbf{U}^{(f)}h_{jk} + \mathbf{b}^{(f)}) \quad (5.6)$$

Next, the individual forget gate outputs are multiplied with corresponding cell state values and then combined to get a single forget vector which is further used to get the final cell state of the model as follows:

$$\tilde{\mathbf{f}}_j = \sum_{k \in C(j)} f_{jk} \cdot c_k \quad (5.7)$$

$$\mathbf{c}_j = \mathbf{i}_j \cdot \tilde{\mathbf{c}}_j + \tilde{\mathbf{f}}_j \quad (5.8)$$



Finally, the update equation for the hidden state of a child sum tree LSTM cell is similar to the one used in traditional LSTM,

$$\mathbf{h}_j = o_j \cdot \tanh(c_j) \quad (5.9)$$

Each of the parameter matrices represents a correlation among the component vector, input  $x_j$  and the hidden state  $h_k$  of the  $k^{th}$  child of the component unit. For example, the sigmoid function at the input gate represents semantically important words at input by giving values close to 1 (e.g., a verb) and relatively unimportant words by giving values close to 0 (e.g., a determiner). Since the hidden state and cell state values of the parent node are generated based on the hidden state and the cell state of its children, child sum Tree LSTM is well suited for trees with a high branching factor or whose children are unordered. Because of this phenomenon, it is a good choice for dependency trees where the number of dependents of a parent can be highly variable.

### 5.3.3 Structured Attention

The attention mechanism [24] has been a breakthrough in neural machine translation (NMT) in recent years. This mechanism calculates how much attention the network should give to each source word to generate a specific translated word. The context vector calculated by the attention mechanism mimics the syntactic skeleton of the input sentence precisely given a sufficient number of examples. Recent work suggests that incorporating explicit syntax alleviates the burden of modeling grammatical understanding and semantic knowledge from the model [234]. However, these features are designed by evaluating the model on some downstream tasks without having any representation [136].

Sentences in bio-medical texts can be comparatively quite complex. For instance, information about a protein relation sometimes extends over more than one syntactic constituent, or a modifier following a protein name sometimes names a new protein. As a consequence, most of the research uses dependency graph information as an external feature or carefully engineers more compact features extracted from the dependency tree arcs [181, 261]. On the other hand, some research adopts input latent graph parsing [87] as the syntax representation. Inducing the dependency tree in a principled manner while training allows the model to learn the internal representation of the sentence very well [115, 234].

In our structured attention model, the input sentence is first fed to a BLSTM which gives output at each time step for each word in the sentence.

$$\mathbf{S} = BLSTM(x) \quad (5.10)$$

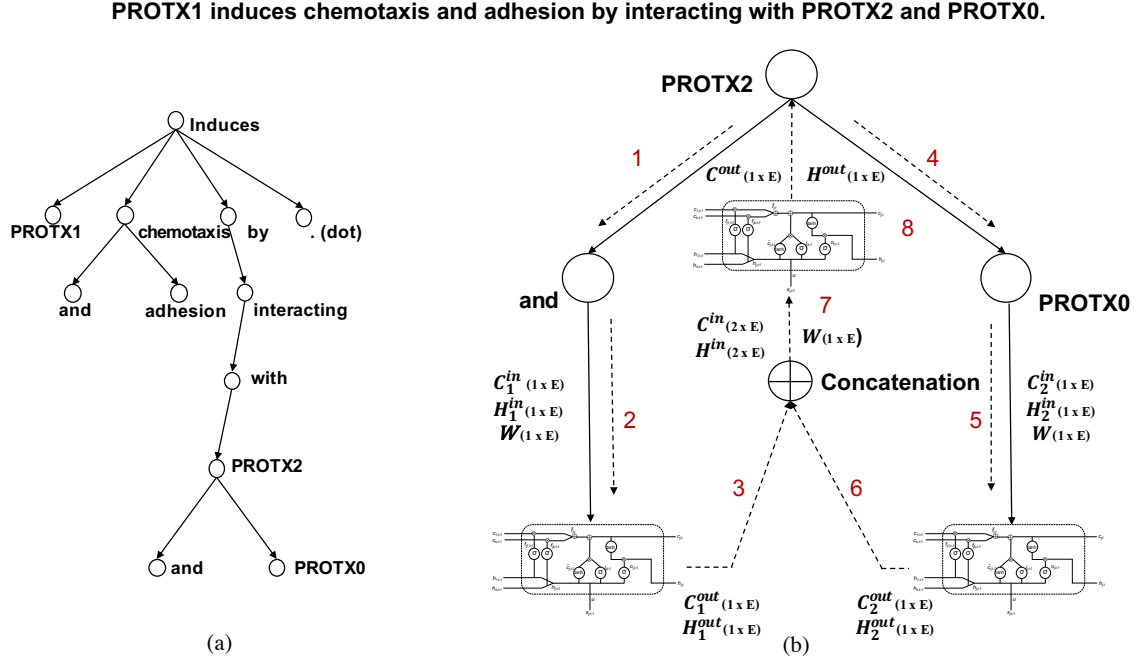


Figure 5.2: Work flow of a child sum tree LSTM on part of a dependency tree

where the term  $S$  is the content annotation. Next, this  $S$  is transformed into a structured annotation as a matrix through structured attention. To do this, first, we initialize three matrices  $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d \times d}$  and add them as trainable model parameters. Here  $d$  is the hidden dimension of the BLSTM. Then using these matrices we map  $S$  into query, key and value matrices  $S_q = \mathbf{W}_q \mathbf{S}, S_k = \mathbf{W}_k \mathbf{S}, S_v = \mathbf{W}_v \mathbf{S} \in \mathbb{R}^{n \times d}$  respectively. Here  $n$  is the length of the source sentence. Next we use Kirchhoff's matrix-tree theorem for computing marginals of non-projective dependency parsing and calculate a structured attention matrix on BLSTM output  $S$  [238]. At first, we multiply the query and key matrices to get an intermediate score matrix .

$$\text{score}_i = S_q S_k^T \quad (5.11)$$

Next, we initialize a query matrix  $R_q \in \mathbb{R}^{1 \times d}$  for the root node and add it as a model parameter. Following this, we multiply this  $R_q$  with the key matrix  $S_k$  to get a vector of length  $n$

$$\text{root} = S_k R_q^T \quad (5.12)$$

Next we pick the diagonal elements of  $\text{score}_i$  and add it with the  $\text{root}$  vector to get a final score and then we normalize it using a partition function. Finally, we arrange this vector in the form of a block-diagonal matrix of size  $n \times n$ . We call this matrix  $\phi$ . The cell  $\phi_{ij}$  means how likely the word  $x_i$  is to be the parent of word  $x_j$  as it captures all pairwise word dependencies.

We are interested in selecting a soft parent word for each word and to do this we can transform the matrix  $\phi$  into an attention matrix  $A$  where each cell  $A_{ij}$  is the posterior probability  $p(\mathbf{x}_i = \text{parent}(\mathbf{x}_j) | \mathbf{x})$ . We define an adjacency matrix  $z \in \{0 \times 1\}^{n \times n}$  in order to encode the source's dependency tree. We can transform our posterior into a marginal by defining it as  $p(\mathbf{z}_{ij} = 1 | \mathbf{x}; \phi)$  which is interpreted as the probability of word  $x_i$  to be the parent of word  $x_j$  given the input  $x$  and matrix  $\phi$ . So the term  $A$  becomes

$$A_{ij} = p(\mathbf{z}_{ij} = 1 | \mathbf{x}; \phi) = \sum_{z: z_{ij}=1} p(\mathbf{z} | \mathbf{x}; \phi) \quad (5.13)$$

Next, we calculate the marginal of non-projective dependency structures using a framework proposed by [115] which utilizes Kirchhoff's Matrix-Tree Theorem [238]. In order to fill all the cells of the attention matrix  $A$ , we need to calculate the spanning tree from each source word in the sentence along with the probability of reaching every target node. To do this we first define a Laplacian matrix  $\mathbf{L} \in \mathbb{R}^{n \times n}$  as follows:

$$\mathbf{L}_{ij}(\phi) = \begin{cases} \sum_{\substack{k=1 \\ k \neq j}}^n \exp(\phi_{kj}), & \text{if } i = j \\ -\exp(\phi_{ij}), & \text{otherwise} \end{cases} \quad (5.14)$$

Next we define another matrix  $\tilde{\mathbf{L}}$  for root word selection as follows:

$$\tilde{\mathbf{L}}_{ij}(\phi) = \begin{cases} \exp(\phi_{jj}), & \text{if } i = 1 \\ \mathbf{L}_{ij}(\phi), & \text{if } i > 1 \end{cases} \quad (5.15)$$

The marginals are then calculated as,

$$\begin{aligned} p_1 &= (1 - \delta_{1j}) \left\{ \exp(\phi_{ij}) [\tilde{\mathbf{L}}^{-1}(\phi)]_{jj} \right\} \\ p_2 &= (1 - \delta_{i1}) \left\{ \exp(\phi_{ij}) [\tilde{\mathbf{L}}^{-1}(\phi)]_{ji} \right\} \end{aligned} \quad (5.16)$$

$$\mathbf{A}_{ij} = p_1 - p_2 \quad (5.17)$$

where  $\delta_{ij}$  is the Kronecker delta. Finally the marginals for the root node is calculated as,

$$\mathbf{A}_{k,k} = \exp(\phi_{k,k}) [\tilde{\mathbf{L}}^{-1}(\phi)]_{k,1} \quad (5.18)$$

This marginal computation is fully differentiable, thus we can train the model with the standard back-propagation algorithm [82].

### 5.3.4 Combining the modules

In this subsection, we combine tLSTM and structured attention as discussed above to build our final model. To the best of our knowledge no work has combined the independently produced gold standard dependency structure information with learning the structure through the model without accessing the actual dependency tree. The method described below accomplishes this fusion.

For the tree LSTM module, we first take the raw sentence and apply the Stanford dependency parser to represent it as a vector of parents where the value  $j$  at index  $i$  means word  $x_j$  is the parent of word  $x_i$  in the dependency tree. We call this vector  $\mathbf{P}$ . Next, using this  $\mathbf{P}$  we compute a tree for each sentence and as an attribute we store all of its child information. This allows us to recursively traverse the entire tree if we start from the root. Apart from this, we have another matrix  $\mathbf{W}$  which is the embedded representation of each of the words in the sentence. Next, we pass the root of this tree and  $\mathbf{W}$  to a recursive module which returns a hidden state and a cell state value for the entire sentence by traversing in a tree-structured manner. Figure 5.2 shows the work flow of tLSTM model on the dependency tree of a sentence. Figure 5.2(a) shows a sample dependency tree of one of the sentences from the corpus and Figure 5.2(b) shows how the hidden state and cell state of the root node of a sub-tree gets calculated. As shown in Figure 5.2(b), for a sub-tree with two children, the work flow is as follows:

While doing a traversal from the root, the tLSTM calculates the hidden state and the cell state of a node using its child hidden state, child cell state which have already been calculated recursively,

$$\mathbf{H}, \mathbf{C} = \text{tLSTM}(\mathbf{W}, (H_i)^c, (C_i)^c) \quad (5.19)$$

here,  $i$  represents the  $i^{\text{th}}$  child,  $H$  is the hidden state and  $C$  is the cell state.  $H$  and  $C$  marked with  $c$  refers to the child hidden and cell state. For our example, Eqn. 5.19 gets called for the word ‘and’ and for the word ‘PROTX0’ as leaf nodes and returns two sets of hidden state and cell state vectors. Next we concatenate the two hidden state vectors and the two cell state vectors and again apply Eqn. 5.19 on the resulting vector. But as a parameter, this time we pass the word vector for ‘PROTX1’, the concatenated hidden state vector and the concatenated cell state vector. This gives us a new hidden state and cell state vectors for the word ‘PROTX1’. We continue to traverse the whole dependency tree in this manner, finally finishing with an encoded hidden state value  $\mathbf{H}_e \in \mathbb{R}^{1 \times n}$  and a cell state value  $\mathbf{C}_e \in \mathbb{R}^{1 \times n}$  for the entire tree.

For the structured attention module, we use  $\mathbf{W}$  as input and apply a BLSTM on it to get an output vector,  $\mathbf{O}$ , which contains the LSTM output for each time step. Next, we pass this to the

Corpus	#Positive	#Negative	#Sentences
AIMed	1,000	4,834	1,955
BioInfer	2,534	7,132	1,100
IEPA	335	482	486
HPRD50	163	270	145
LLL	164	166	77

Table 5.1: Basic statistics of the corpora

structured attention (sAttn) module which gives an attention matrix  $\gamma \in \mathbb{R}^{n \times n}$  as output.

$$\gamma = \text{sAttn}(\mathbf{O}) \quad (5.20)$$

Next we use this  $\gamma$  with value matrix  $\mathbf{S}_v$  to calculate the syntactic context  $\mathbf{C}_s \in \mathbb{R}^{n \times d}$  as follows.

$$\mathbf{C}_s = \gamma \mathbf{S}_v \quad (5.21)$$

Following this, we take the vectors only at the first and last index of  $\mathbf{C}_s$  which has the entire left context as well as right context information respectively and concatenate them. We term this as  $\tilde{\mathbf{C}}_s$ . Then we concatenate this  $\tilde{\mathbf{C}}_s$  with  $\mathbf{H}_e$  to get the final context  $\mathbf{M}$ . Next, we use an MLP followed by sigmoid over this  $\mathbf{M}$  to generate a non-linear version  $\tilde{\mathbf{M}}$ . Finally, our model predicts a corresponding label  $y$  from this  $\tilde{\mathbf{M}}$  as follows,

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}, \theta) &= \text{sigmoid}(\text{MLP}(\mathbf{M})) \\ y_i &= \arg \max_y p(\mathbf{y}|\mathbf{x}, \theta) \end{aligned} \quad (5.22)$$

## 5.4 Experimental Analysis and Results

In this section, we describe the results obtained with our proposed architecture. We use precision, recall and F-score as our evaluation metrics. This section also contains the detailed statistics of all of the five PPI corpora, the preprocessing steps applied to convert the problem into classification domain as well as the hyper-parameter settings of our models. In addition to that, it contains the results of the top performing models for all the corpora and extensive comparative analysis with our models. Finally, we conclude this section by giving cross corpus evaluation statistics of our architecture where we train our model on one corpus and test on another.

We evaluate our tLSTM model on five publicly available PPI corpora: AIMed [42], BioIn-

fer [186], IEPA [68], HPRD50 [79] and LLL [167]. In our experiments, we use the converted version of these corpora<sup>1</sup> and details about these along with the conversion characteristics can be found in [185]. The statistics of the five PPI corpora are given in Table 5.1.

In order to generalize the learned model, we have modified the corpora slightly. Protein names are replaced with special symbols in each sentence, i.e., PROTX0, PROTX1 and PROTX2. Here, PROTX1 and PROTX2 are the proteins of interest and all other non-participating proteins are marked as PROTX0. For example, the following sentence “PROTX1 induces chemotaxis and adhesion by interacting with PROTX2 and PROTX0” indicates that PROTX1 and PROTX2 have a positive interaction. Similarly, the sentence “PROTX0 induces chemotaxis and adhesion by interacting with PROTX1 and PROTX2” indicates that PROTX1 and PROTX2 have a negative interaction. In this example there are three possible pairs of proteins and hence three variants of the sentence is possible. Two of them have positive interaction and one has negative interaction. In general, if a sentence has  $n$  protein references, there are  $\binom{n}{2}$  protein pairs and hence  $\binom{n}{2}$  variants of the sentence.

We evaluated our model with 10-fold cross validation on each corpus allowing us to compare our results with relevant earlier works. In  $k$ -fold cross validation, the corpus is divided into  $k$  parts,  $(k - 1)$  parts are training data and the other part is testing data, and is repeated  $k$  times. We used StratifiedKFold from Python’s Scikit-learn package which preserves the percentage of samples for each class in each fold [179].

Table 5.2 shows the detailed hyper-parameter settings used for our model. We trained our model on a GeForce GTX 1080 GPU with the ‘Adam’ and ‘SGD’ optimizers. All the results in the next section are reported using ‘SGD’ as it was giving the best results. The ‘Learning rate decay’ parameter was only used with the ‘SGD’ optimizer. We used PyTorch 0.4 to implement our model under the Linux environment.

Table 5.3 shows the overall evaluation of our model in terms of precision, recall and F-score for the five PPI corpora and compares these results with the currently available state of the art models. Among these five corpora, AIMed is the most difficult as it has more noise, the sentences have nested named entities and there are many inaccurate annotations. With the AIMed corpus, we achieved a highest F-score of **81.6%** with a significant **4.7** percentage points improvement over the previous best model. The tLSTM + tAttn model also achieved the best precision and recall scores of **81.4%** and **81.9%**, respectively. Our tLSTM model without attention also surpassed all of the existing models with a significant improvement in precision, recall and F-score achieving 80.5%, 80.8% and 80.6%, respectively. The previous best model [93] uses just the raw words and a BLSTM to capture the word context from the forward and backward directions. The second corpus that we evaluated our model on is BioInfer, the

<sup>1</sup><http://mars.cs.utu.fi/PPICorpora/>

Hyper-parameter	Values
Number of layers	1/2
Embedding dimensions	200
Hidden dimensions	<b>300</b> /400/500
Batch size	<b>10</b> /16/20
Number of epochs	30/ <b>40</b> /50
Dropout rate	<b>0.5</b> /0.1
Learning rate	0.001/ <b>0.015</b>
Learning rate decay	0.05

Table 5.2: Hyper-parameters used for the experiments (in boldface) and the ranges that were searched during tuning.

Methods	<i>AIMed</i>			<i>BioInfer</i>			<i>IEPA</i>			<i>HPRD50</i>			<i>LLL</i>		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
GK [15]	52.9	61.8	56.4	56.7	67.2	61.3	69.6	82.7	75.1	64.3	65.8	63.4	72.5	87.2	76.5
CK [159]	55.0	68.8	60.8	65.7	71.1	68.1	67.5	78.6	71.7	68.5	76.1	70.9	77.6	86.0	80.1
WWSK [108]	61.4	53.3	56.6	61.8	54.2	57.6	66.7	69.2	67.8	73.7	71.8	72.9	76.9	91.2	82.4
NHGK [260]	54.9	68.5	60.2	59.3	68.1	63.4	72.4	79.8	75.3	67.8	<b>85.3</b>	74.6	86.2	<b>92.1</b>	<b>89.1</b>
EDG [180]	57.3	65.3	61.1	57.6	59.9	58.7	69.9	76.2	72.9	76.7	83.3	79.9	<b>92.1</b>	78.2	84.6
PIPE [48]	57.2	64.5	60.6	68.6	70.3	69.4	62.5	<b>83.3</b>	71.4	63.8	81.2	71.5	73.2	89.6	80.6
Bi-LSTM [93]	78.8	75.2	76.9	87.0	87.4	87.2	–	–	–	–	–	–	–	–	–
RNN+CNN [261]	52.9	61.8	56.4	56.7	67.2	61.3	69.6	82.7	75.1	64.3	65.8	63.4	72.5	87.2	76.5
tLSTM	80.5	80.8	80.6	88.3	87.9	88.1	77.0	76.7	76.4	<b>82.4</b>	82.8	<b>82.0</b>	85.3	84.9	84.8
tLSTM + tAttn	<b>81.4</b>	<b>81.9</b>	<b>81.6</b>	<b>88.9</b>	<b>89.3</b>	<b>89.1</b>	<b>78.6</b>	78.7	<b>78.5</b>	81.7	82.3	81.3	84.8	84.3	84.2

Table 5.3: Results (in %) of our model (tLSTM) from 10-fold cross-validation against other methods. Bold text indicates the best performance in a column. **GK**: Graph Kernel (Airola et al., 2008). **CK**: Composite Kernel (Miwa et al., 2009). **WWSK**: Walk-weighted Subsequence Kernel (Kim et al., 2010). **NHGK**: Neighborhood Hash Graph Kernel (Zhang et al., 2011). **EDG**: Extended Dependency Graph (Peng et al., 2015). **PIPE**: Protein-protein Interaction Passage Extraction (Chang et al., 2016). **Bi-LSTM**: Bidirectional Long-Short Term Memory (Hsieh et al., 2017). **RNN + CNN**: Combination of Recurrent and Convolutional Neural Network (Zhang et al., 2018).

corpus with the most (9666) annotated interactions among the five. It has fewer sentences but more annotated examples than AIMed indicating that the sentences are significantly longer and contain a large number of proteins in a single sentence. With BioInfer, our tLSTM + tAttn model achieves a highest precision, recall and F-score of **88.1%**, **89.3%** and **89.1%**, respectively. Our tLSTM model without attention is the second best with precision – 88.3%, recall – 87.9% and F-score – 88.1%. It is to be noted that we achieved state of the art results

	<i>AIMed</i>	<i>BioInfer</i>	<i>IEPA</i>	<i>HPRD50</i>	<i>LLL</i>
AIMed †	–	47.0	38.6	41.5	34.6
AIMed ‡	–	45.0	37.9	39.1	33.5
BioInfer †	50.8	–	40.8	43.7	35.0
BioInfer ‡	50.0	–	40.0	45.5	33.5

Table 5.4: Cross-corpus results (F-score in %). Rows correspond to training corpora and columns to testing. Models marked with † represent *tLSTM* and ‡ represent *tLSTM + tAttn*

with all evaluation metrics on these two large and complex corpora without any manual feature engineering. With the IEPA corpus, our tLSTM + tAttn model achieves a highest precision and F-score of **78.6%** and **78.5%**, respectively. Our model’s recall score is 78.7% which is behind only the 83.3% of [48] which combines PPI with a convolution tree kernel. However, their precision and F-score is low compared to both of our tLSTM and tLSTM + tAttn models. Regarding the HPRD50 corpus, our tLSTM model without attention achieves the best precision and F-score of **82.4%** and **82.0%** respectively. Our tLSTM + tAttn model is the second best in terms of precision and F-score. However, none of our models reached the best recall score of 85.3% by [260] which is based on extracting the higher order substructure of the dependency graph by bit label operations on dependency graph nodes. Again, their precision and F-score is low compared to both of our models. With the LLL corpus, none of our models achieve best scores. Instead the tLSTM model without attention achieves the second best F-score of 84.8% and the third best precision score of 85.3%. The best recall and F-score of 92.1% and 89.1% is achieved by [260] which uses a neighbourhood hash graph kernel whereas [180] achieves the best precision score of 92.1% using an extended dependency graph. However, [260] and [180] both perform poorly for the other four datasets in comparison with our models. An interesting aspect of our evaluation is that whenever the number of training data samples is large, no matter how complex the samples are, deep learning based methods perform very well compared to the feature based methods. With a small number of training data samples, the performance can fall short of other methods. This is what happens with LLL, the smallest corpus. Also a large number of training data samples allows the structured attention mechanism to extract the dependency information very well. That is why for comparatively large corpora, AIMed, BioInfer and IEPA, our model with attention performs best and achieves state of the art results, whereas for the two small corpora, tLSTM without attention performs better.

The cross corpus evaluation is inspired by the work [239] to answer the fundamental question of practical PPI extraction – “which corpus to be trained on in real life?”. Table 5.4 shows this cross corpus evaluation. Rows correspond to the training corpora and columns correspond to the test corpora. We only used AIMed and BioInfer as the training corpora and ignored



the small ones because there is no point in training on small simple corpora and test on large complex corpora as suggested in [181]. It is clearly visible that the performance degrades on all of the corpora as the training and testing sets are not from the same distribution which goes against the fundamental machine learning theory about training and test sets being identically distributed. Being larger in size, the models that are trained on BioInfer perform better than the models trained on AIMed. One more interesting aspect of our evaluation is that the models without attention perform better than the models with attention. The main reason is that our structured attention captures the syntactic dependencies in the sentences and because of the two different distributions between the training and testing sets, the attention mechanism fails to capture these dependencies. Overall our cross corpus evaluation is close to the one from [181] with a slight improvement when training on BioInfer and testing on AIMed.

## 5.5 Conclusions

In this chapter, we propose a tree recurrent neural network architecture with structured attention mechanism for the supervised PPI extraction problem. Our model gets significant improvement on two largest public PPI corpora, AIMed and BioInfer. Addition to that, our model gets state of the art result for several other small corpora too. Our experimental result shows that our tree LSTM model with structured attention is more suitable compared to traditional recurrent neural network based approaches for extracting useful features from dependency tree information of a given bio-medical text. Moreover, we believe that other linguistics features that are already proven to be useful for PPI can be included to improve the model. In future, we would like to explore the idea of leveraging other features to make our model more accurate.

## Chapter 6

# Efficient Transformer-based Sentence Encoding for Sentence Pair Modelling

This chapter is based on the paper titled “Efficient Transformer-based Sentence Encoding for Sentence Pair Modelling” co-authored with Robert E. Mercer that appeared in The 32nd Canadian Conference on Artificial Intelligence (CAI 2019) [7]. The paper was awarded “Best Student Paper Award”.

Modelling a pair of sentences is important for many NLP tasks such as textual entailment (TE), paraphrase identification (PI), semantic relatedness (SR), and question answer pairing (QAP). Most sentence pair modelling work has looked only at the local context to generate a distributed sentence representation without considering the mutual information found in the other sentence. The proposed attentive encoder uses the representation of one sentence generated by a multi-head transformer encoder to guide the focussing on the most semantically relevant words from the other sentence using multi-branch attention. Evaluating this novel sentence encoder on the TE, PI, SR and QAP tasks shows notable improvements over the standard Transformer encoder as well as other current state-of-the-art models.

### 6.1 Introduction

Modelling the relationship of sentence pairs typically involves some comparison of the sentences: semantic relatedness [151], textual entailment between premise and hypothesis [39], true-false question-answer selection [251] and paraphrase identification [89]. Previous work tends to use the representation of each sentence separately overlooking the impact of combining the information from the two sentences. It is contrary to what humans do, we usually look at the keywords of both sentences when doing a comparison. For example, when comparing the sentence “*There is no biker jumping in the air*” with the sentence “*A lone biker is jumping in the air*” for the semantic relatedness task, attention should be given on the word “no” if we

consider both sentences at the same time, whereas if we look at them independently, then the attention shifts to the most important portion of the sentences, “*biker is jumping in the air*”, which may not have any contribution to the semantics. This example shows the need for an architecture that builds a representation of one sentence by looking at both.

Recurrent neural networks (RNNs) [194] are the most widely used neural network model for sequence data. To overcome issues such as the “vanishing gradient problem”, variants have been proposed: LSTM [92] and GRU [55]. They have been successful in a variety of applications such as machine translation, sequence labelling, speech recognition, and question answering. These models are difficult to parallelize because of a built-in state dependency and they are also inclined to overfit.

A popular LSTM framework is the sequence-to-sequence model using an encoder-decoder architecture where the encoder encodes the entire source sequence and the decoder generates the target sequence. For longer sequences, this model tends to generate asymmetrical sequences. To solve this, “Attention” was proposed, especially for the machine translation task: the decoder looks at each encoder word before generating a single output at each time step [24, 144]. However this kind of Attention has a high computational overhead and it is affected by the state dependency problem. As a variant to this, Parikh et al. [176] proposed a decomposable attention mechanism which first decomposes the entire problem into sub-problems and then calculates attention via a soft alignment. Utilizing this, Vaswani et al. [240] proposed the “Transformer” model, which doesn’t have any state dependencies, for the machine translation task. The main building blocks of their model are positional encoding [80], multi-head attention, and layer normalization [21].

In this chapter, we propose an improved “Transformer Encoder” model to encode a pair of sentences for the sentence pair modelling task. To model a sentence pair, each sentence is viewed individually via multi-head attention. The semantic keywords in both sentences are looked at using a multi-branch cross attention which takes the representation of one sentence to put attention on the words of the other sentence. Evaluation done on the four aforementioned tasks shows notable improvements over the standard Transformer encoder on all of the tasks and best results on the textual inference and question-answer pair tasks when compared against all state-of-the-art encoder models including the best models especially designed for the specific tasks.

## 6.2 Related Work

Cer et al. [45] propose a sentence encoder model based on the encoder portion of Transformer [240] and perform an element-wise sum of the encoded representations at each word

position to get a fixed length sentence representation. They evaluate their model mostly on sentence classification tasks and proved the effectiveness of transfer learning at the sentence level compared to the word level. Conneau et al. [58] also propose a universal sentence representation model based on LSTMs where it is first trained on the Stanford Natural Language Inference task and then uses transfer learning. Evaluation is on a range of tasks including QAP, PI, and SR. Zhou et al. [270] propose a sentence pair ranking model where they encode attention in the tree structure of the hypothesis based on the sequential representation of the premise. Lin et al. [135] propose a self-attentive sentence encoding model where they replace the pooling block with a self attention block on top of an LSTM encoder. Instead of extracting a vector representation, the authors use a matrix as the sentence representation where each row attends to different portions of the sentence. Zhao et al. [263] propose a self-adaptive hierarchical model which first extracts an intermediate representation of all possible phrases and finally takes the convex combination of them through gating. Yang et al. [247] use a hierarchical attention network for document classification where the first BiLSTM gets the word level attention and the second BiLSTM extracts the sentence level attention. Socher et al. [208] propose a recursive auto-encoder-based paraphrase identification model that first reconstructs each of the phrases from the tree representation of both sentences. Finally, this model extracts the sentence representation by doing a min-pooling operation over the transformed differences between the sentences. Yin et al. [251] propose an attention based convolutional sentence encoding model where they infer the attention matrix over a sentence pair through a simple matching function and then apply weighted pooling followed by another set of convolutions to get the final sentence representation. Mueller et al. [163] first encode the sentence pair to be compared using a siamese BiLSTM encoder and then use a simple Manhattan distance based matching function to infer the similarity score. Finally, they use an additional non-parametric regression block to further calibrate their model's prediction.

## 6.3 The Model

The standard pipeline architecture for sentence pair modelling starts with encoding both the hypothesis and premise sentences as vectors using a neural sentence encoder followed by a matching layer where the corresponding vector representations are compared with a similarity metric. Our sentence encoder architecture for modelling pairs of sentences is based on the “Transformer” [240] and the “Universal Sentence Encoder” [45]. Unlike LSTMs where the information at the current time step is dependent on the previous one, Transformer can encode all the words of a sentence using only linear tensor multiplications making it easily parallelizable. Figure 6.1 shows a sketch of our Transformer based sentence encoder which is built using

stacked self-attention, point-wise fully connected layers, and branch attention for both of the encoders, respectively.

First our encoder inputs both the hypothesis and premise as sequences,  $H = [h_1, h_2, \dots, h_a]$  and  $P = [p_1, p_2, \dots, p_b]$  respectively where  $a$  denotes number of words in the hypothesis and  $b$  denotes number of words in the premise. We then initialize a pre-trained word embedding layer which transforms each of the words in those sequences into vectors and gives a new input representation in terms of tensors  $H_{a \times d}$  and  $P_{b \times d}$ . However, these tensors only have bag of words information without any explicit knowledge of the word positions. In order to encode the positional information with the input, we initialize a positional encoding layer, where the positions are assigned by a unique 300 dimensional vector with each dimension being a sinusoid. Out of many possible initialization, we choose to initialize the odd dimensions by a sine function and the even dimensions by a cosine function as follows,

$$\begin{aligned} PE(pos, 2i) &= \sin(pos/10000^{(2i/d_{model})}) \\ PE(pos, 2i+1) &= \cos(pos/10000^{(2i/d_{model})}) \end{aligned} \quad (6.1)$$

where  $pos$  denotes the position and  $i$  denotes the dimension. These wavelengths form a geometric progression from  $2\pi$  to  $10000 \cdot 2\pi$  having an interesting property that for any fixed offset  $k$ ,  $\sin(k/z)$  and  $\cos(k/z)$  are constant and  $PE_{pos+k}$  can be represented by some linear function of  $PE_{pos}$ . Following this, we add a deep “Multi-Head Self Attention” layer which allows each position of our encoder to attend to all relative positions in the previous layer enabling it to easily learn long-range dependencies. In order to avoid repetition, we will explain this layer only for the hypothesis as for the premise it is exactly same. We start by initializing three learned parameters  $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{n \times d \times l}$  where  $n$  denotes the number of heads and  $d$  and  $l$  represent the hidden dimensions having the constraint  $d = n \times l$ . Next, we copy the hypothesis tensor,  $H$ ,  $n$  times creating a new tensor  $\tilde{H}$  of size  $n \times p \times d$  and multiply it with the three parameter tensors  $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$  independently as follows,

$$Q = \tilde{H}^T \mathbf{W}_q \quad K = \tilde{H}^T \mathbf{W}_k \quad V = \tilde{H}^T \mathbf{W}_v \quad (6.2)$$

This produces three new tensors called query ( $Q$ ), key ( $K$ ), and value ( $V$ ) with each of them having dimension  $n \times p \times l$ . Next we apply batch-wise tensor multiplication on query and key, normalize it and pass the result through Softmax which gives us the attention probabilities,  $A_{n \times p \times p}$  as follows,

$$A = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right) \quad (6.3)$$

Following this, we multiply this attention  $A$  with value  $V$  which gives us the transformed

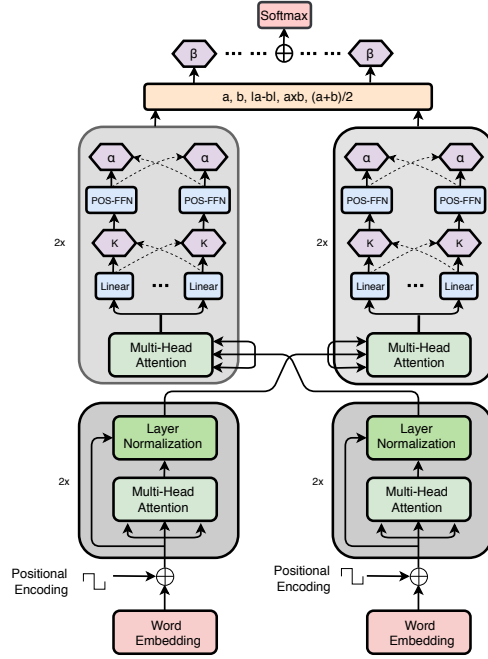


Figure 6.1: Model architecture

hypothesis tensor  $\hat{H}_{n \times p \times l}$  ( $\hat{H} = AV$ ) where each of the  $n$  heads captures how much attention should be given on different phrases within the sentence and finally, we combine all the decisions by doing a concatenation over the cardinal dimension of the resulting tensor. Next we add a residual block followed by a layer normalization block where we do the normalization of inputs across the features. This permits each input to have a different normalization operation thereby allowing arbitrary mini-batch sizes to be used easily. This gives the final self attentive representation of the hypothesis. We follow the same procedures as above and derive the self attentive representation of the premise  $\hat{P}_{q \times d}$ . Instead of passing these representations to a matching layer we utilize the mutual information between the sentences through another layer of attention. Our idea is inspired by Rocktaschel et al. [193], Hermann et al. [91] and Zhou et al. [270]. Contrary to choosing “Multi-Head Attention”, we choose “Multi-Branch Attention” as the composition function of this layer. According to Ahmed et al. [3], these branches mimic the same principles of multiple heads with an additional advantage of inferring a different representation at a time and the model automatically learns to combine these representations during training. Next, we explain this layer in terms of the hypothesis only in order to avoid needless repetition. Initially, we utilize the same equations as above from Multi-Head attention with the exception that the query is calculated on the hypothesis whereas the key and value are calculated on the premise as follows,

$$\hat{Q} = \hat{H}^T \hat{W}_q \quad \hat{K} = \hat{P}^T \hat{W}_k \quad \hat{V} = \hat{P}^T \hat{W}_v \quad (6.4)$$

Here  $\hat{\mathbf{W}}_q$ ,  $\hat{\mathbf{W}}_k$ ,  $\hat{\mathbf{W}}_v$  are new model parameters defined especially for this layer. Next we apply Eqn. 6.3 with these new  $\hat{Q}$  and  $\hat{K}$  to get a cross attention probability matrix  $\hat{A}_{n \times p \times q}$  and following this, we extract an intermediate representation through  $\hat{H} = \hat{A}\hat{V}$ . We then initialize a learned parameter  $\mathbf{W}_z \in \mathbb{R}^{n \times d \times d}$  and perform a batch-wise tensor multiplication with  $\hat{H}$  to get a cross attentive hypothesis representation  $\bar{H}_{n \times p \times d}$  as follows,

$$\bar{H} = \hat{H}\mathbf{W}_z \quad (6.5)$$

Here  $n$  depicts the branch id. Intuitively, this  $\bar{H}$  contains  $n$  different transformed representations of its input where each of them resides in its individual branch. Following this, we scale each of these branch representations with a new learned parameter  $\kappa \in \mathbb{R}^n$  and add a layer normalization block on top of it,

$$\bar{H}_i = \text{layerNorm}(\bar{H}_i \kappa_i) \quad (6.6)$$

However, we require  $\sum \kappa_i = 1$ .

Next we add a separate and identical position-wise feed forward module on each position. Vaswani et al. [240] use two convolutions with kernel size 1 and a RELU activation function. However, in this work we use two feed forward layers with feature expansion dimension of size 2048. Eqn. 6.7 summarizes these as follows,

$$FFN(x) = \max(0, x\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \quad \text{where } x \in \{\bar{H}_1, \bar{H}_2, \dots, \bar{H}_n\} \quad (6.7)$$

This gives a new set of transformed hypothesis tensors  $\bar{\bar{H}}_1, \bar{\bar{H}}_2, \dots, \bar{\bar{H}}_n \in \mathbb{R}^{p \times d}$ . Finally we perform another scaling operation on this set of tensors with a new learned parameter  $\alpha \in \mathbb{R}^n$  and then add all these resulting tensors to get the final attentive hypothesis representation as follows,

$$H^f = \sum_{i=1}^n \bar{\bar{H}}_i \alpha_i \quad (6.8)$$

Again, we require  $\sum \alpha_i = 1$ . We use the same procedures as above with slightly changed notation to get the final attentive premise representation  $P^f$ . Specifically, to achieve this, in Eqn. 6.4, query is calculated on the premise whereas key and value are calculated on the hypothesis and following this all instances of  $H$  are replaced with  $P$ . Next, we add a matching layer where both of these representations are compared and a class decision is made according to the underlying task. In this layer, both the hypothesis ( $H^f$ ) and premise ( $P^f$ ) representations are passed through a position-wise feed forward block as depicted in Eqn. 6.7. Next, these self- and cross-context-aware word representations are converted to a fixed length sentence encoding vector by computing the element-wise sum of the representations at each word position. A

nonlinearity,  $\tanh$ , is introduced in order to make this vector compatible with the underlying objective functions.

$$\begin{aligned} a &= \tanh(\text{elementwiseSum}(H^f)) \\ b &= \tanh(\text{elementwiseSum}(P^f)) \end{aligned} \quad (6.9)$$

Next, we compute a range of features such as difference, element-wise product and average for the tuple  $\langle a, b \rangle$ . We anticipate that such an operation could help to capture the inference between components in the tuples and catch induction relationships such as logical inconsistency. These features are then concatenated with the original representations,  $a$  and  $b$ , and we create a new tensor representation from this as follows,

$$F = \text{makeTensor}([a, b, |a - b|, a * b, \frac{(a + b)}{2}]) \quad (6.10)$$

Next, we perform an element-wise multiplication of this feature tensor with a learned parameter vector  $\beta \in \mathbb{R}^n$  having the constraint  $\sum \beta_i = 1$ . The final inference relationship vector results from an element-wise weighted addition of each feature position is as follows,

$$\bar{F} = \sum_{i=1}^4 F_i \beta_i \quad (6.11)$$

Next, an MLP maps this  $\bar{F}$  to the required number of classes  $Y_{1 \times c} = \text{MLP}(\bar{F})$ , where  $c$  represents the number of classes. Finally we use  $\text{Softmax}$  to turn this into class probabilities and our model predicts a corresponding label  $y^*$  as follows,

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}, \theta) &= \text{Softmax}(\text{MLP}(\bar{F})) \\ y^* &= \arg \max_y p(\mathbf{y}|\mathbf{x}, \theta) \end{aligned} \quad (6.12)$$

## 6.4 Experimental Setup

In this section, we detail the experimental setup used for evaluation. We first describe our training corpora as well as all of the benchmarks used in other standard sentence pair modelling studies. Following this, we explain the technical details of our proposed architectures along with their hyper-parameter settings.

First, we conduct our experiment on the Sentences Involving Compositional Knowledge (SICK) dataset [151]. The sentences are derived from video and image annotations. The first task is to classify a given sentence pair into three classes: Entailment, Neutral and Contradiction. The standard evaluation metric used for this inference task is accuracy. In addition to this,



	<b>SICK</b>	<b>MSRP</b>	<b>AI2-8grade</b>
Train	4500	4076	12689
Valid	500	N/A	2483
Test	4927	1725	11359

Table 6.1: Dataset description. Number of sentences in each split.

<b>Config</b>	<b>Value</b>	<b>Config</b>	<b>Value</b>
Learning rate	<b>0.1</b> / 0.05 / 0.001	Max Norm	<b>5</b>
Batch size	10 / 15 / <b>25</b>	Learning rate decay	<b>0.99</b>
No. of layers	1/ <b>2</b> / 3	Dropout FC	<b>0.1</b>
Hidden dimension	<b>300</b>	No. of Head / Branches	<b>8</b>
Word embedding	Glove 300D	$W_q, W_k, W_v$ dimension	<b>64</b>

Table 6.2: Hyper-parameters used for the experiments (in boldface) and the ranges that were searched during tuning.

each of the SICK sentence pairs is also annotated with a relatedness label  $\in [1, 5]$  corresponding to the average relatedness as judged by different individuals. The standard evaluation metrics used for this relatedness measurement task are Pearson’s  $r$  and mean squared error (MSE). We experiment with two different forms of this problem. While treating it as a regression problem, we use Manhattan distance similarity metric [163], and when converting it into a distribution mapping task, we measure the continuous distance between the predicted and ground truth distribution [227, 270]. We use weighted cross entropy as the loss function for the inference task and for the relatedness task, we use MSE when it’s a regression problem and KL divergence loss when it’s a distribution mapping problem. The next task used for evaluation is paraphrase identification using the Microsoft Research Paraphrase Corpus (MSRP) [69]. Given a pair of sentences, the task is to identify whether or not they are paraphrases of each other. As it is a classification task, we use accuracy as the evaluation metric. Since the sentences in this dataset are mostly stock market related and contain many numbers, during preprocessing we replace all these numbers with a  $\langle \text{number} \rangle$  tag. The final task on which we evaluate our model is the true-false question selection task from the AI2-8grade dataset [30]. Each data sample consists of a pair of sentences with one being the question and the other being the evidence formed by replacing the  $wh$  in the question by the answer. These evidence sentences in the sentence pairs are mainly collected from CK12 textbooks. As it is a binary classification task, we use accuracy as the evaluation metric. Detailed statistics for each of these datasets are summarized in Table 6.1

Table 6.2 shows the hyper-parameter settings used during the experiments. We train our models on a GeForce GTX 1080Ti GPU with ‘Adam’, ‘AdaDelta’, ‘AdaGrad’ and ‘SGD’ optimizers. Results in the next section are reported using ‘SGD’ as it was giving comparatively

better results. We use PyTorch 0.4.1 for implementing the models under the Linux environment.

## 6.5 Experimental Results and Analysis

In this section, we present the detailed results obtained with our Transformer encoder and compare with some of the top performing models for the four sentence pair modelling tasks. Additionally, we give a qualitative analysis by showing the predictions of our models on some random test samples for all of the tasks. Finally, we conclude this section by giving an insight about the performance of our model by analyzing the attention weights through heatmap visualization.

Table 6.3 displays our model’s overall evaluation on the four corpora in terms of task specific evaluation metrics. The first group contains the results of our model in two different configurations, each one having two semi versions: **1.** Head attention for both self and cross, **2.** Head attention for self and Branch attention for cross. In configuration 1, we keep the self and cross attention blocks separate for the hypothesis and premise whereas in the siamese version, we have only one copy of self and cross attention block for both hypothesis and premise as we keep the parameters shared. In order to analyze the potency of our model, we implement some currently available top performing sentence encoders with our fixed hyper-parameter settings and report their results in the second group of this table. The last group contains the results of some preeminent models on each task as well as the ones that report their evaluation on one or more corpora. For the paraphrase identification task on the MSRP dataset, our model achieves the second best accuracy falling just short of RAE [208] which is specifically designed for this task. However, our model with head attention outperforms all the SOTA sentence encoders by quite a good margin of 1.94%. We also get better accuracy than all of the tree structured models from [263] which have an implicit advantage of having access to the sentence structure as parse trees. For the true-false question-answer pairing task, our model with head attention achieves the best results. It clearly exceeds all of the tree structured models [263] by a very good margin of approximately 5-7 percentage points, but it ties with InferSent [58] on accuracy: 77.29%. However, it achieves better results on two other tasks. It is to be noted that the standard RNN and CNN based methods [30] perform very poorly on this dataset. One reason could be there is a lot of repetitiveness in the questions which makes the number of unique samples to be around 1/7th of the real data. For the SICK entailment task (SICK-E), the siamese version of our model with branch attention achieves the best result among all of the reported results as well as the ones that we re-implemented. Among the reported ones, Illinois-LH performs the best, but it employs many task specific features such as the count of words like ‘no’, and ‘not’ and

Model		MSRP Acc.	AI2-8grade Acc.	SICK-E Acc.	SICK-R r/MSE
TRANSFORMER NON SIAMESE	HEAD - HEAD	75.90	75.29	83.17	72.75/0.5129
	HEAD - BRANCH	75.80	74.78	82.63	73.04/0.5042
TRANSFORMER SIAMESE	HEAD - HEAD	76.40	<b>77.29</b>	84.80	72.78/0.5127
	HEAD - BRANCH	75.29	74.20	<b>85.22</b>	73.18/0.4969
INFERSENT [58] †		74.46	77.29	84.62	85.63/0.2732
LSTM [58] †		70.74	76.97	76.80	82.91/0.3244
BiLSTM PROJECTION LAYER [58] †		74.24	74.88	85.20	80.37/0.3667
BiGRU LAST ENCODER [58] †		70.46	74.76	81.47	83.17/0.3147
INNER ATTENTION [135] †		69.74	74.77	72.01	78.63/0.3944
CONVNET ENCODER [263] †		73.96	75.43	83.82	85.20/0.2806
SEQ-LSTMS [270]		71.70	63.30	-	0.8528/0.2831
SEQ-GRUs [270]		71.80	62.40	-	0.8595/0.2689
TREE LSTM [270]		73.50	69.10	-	0.8664/0.2610
TREE LSTM + ATTN. [270]		75.80	72.50	-	<b>0.8730</b> /0.2426
TREE GRU [270]		73.96	70.60	-	0.8672/0.2573
TREE GRU + ATTN. [270]		74.80	72.10	-	0.8701/0.2524
RAE [208]		<b>76.80</b>	-	-	-
COMBINE-SKIP + FEATS [111]		75.80	-	-	-
RNN [30]		-	36.1	-	-
CNN [30]		-	38.4	-	-
RNN-CNN [30]		-	37.6	-	-
ATTN1511 [30]		-	35.8	-	-
UBU.RNN [30]		-	44.1	-	-
ILLINOIS-LH [122]		-	-	84.60	-
UNAL NLP [101]		-	-	83.10	-
SNLI-TRANSFER 3-CLASS LSTM [39]		-	-	80.80	-
MALSTM FEATURES + LSTM [163]		-	-	84.20	-
ECNU [264]		-	-	83.60	0.8414/-
CHILD SUM TREE LSTM [227]		-	-	-	0.8676/0.2532
COMBINE SKIP + COCO [111]		-	-	-	0.8655/0.2561
CONVNET [89]		-	-	-	0.8686/0.2606

Table 6.3: Performance comparison of our model on different tasks against some existing top performing models. We mark models that we implemented as †.

also hypernyms. Among the models that we re-implemented, BiLSTM with a projection layer comes very close to our results but could not surpass. For SICK relatedness task (SICK-R), our model performs very badly compared to all others. To overcome this, we replace our matching layer with a Manhattan distance similarity function [163] but got almost the same results. One reason could be that the vectors produced by Transformer based models have to go through a linear summation followed by a normalization at the end which shrinks the norms of the vector drastically. This transformation makes it unsuitable for regression as well as the distribution

Dataset	Sentence 1	Sentence 2	GT	Pr
SICK-E	A biker is riding away from a fence	A man is dancing on the road	N	N
	Two white dogs are quickly running together	Two white dogs are running together	E	E
	The man is going into the water	The person is going into the sea	E	N
SICK-R	A few men in a competition are running outside	A few men are running competitions outside	3.9	3.7
	A girl in white is dancing	A girl is wearing white clothes and is dancing	4.9	4.5
	A man is talking on the radio	A man is spitting	1.7	1.4
MSRP	I'm never going to forget this day	I am never going to forget this throughout my life	1	1
	You can reach George Hunter at (313) 222-2027 or ghunter@detnews.com	Reach her at (248) 647-7221 or email lberman@detnews.com	0	0
	Orange shares jumped as much as 15 percent	France Telecom shares dropped 3.6 percent while Orange surged 13 percent	0	1
AI2-8grade	Venus has a warmer average surface temperature than Earth?	Even though it is farther from the Sun , Venus is even much hotter than Mercury	1	1
		As a result , Venus is the hottest planet	1	1
		Venus's clouds are a lot less pleasant	1	0
		Clouds on Earth are made of water vapor	1	0

Table 6.4: Example predictions from the test set. **GT**: ground truth, **Pr**: predicted.

mapping problem. However, for classification problems, another linear transformation with a  $\tanh$  activation overcomes this.

Table 6.4 displays some example predictions produced by our Transformer encoder model. The SICK-E examples show that our model is very good at predicting the textual entailment class given a sentence pair. However on the same dataset but with a different task (SICK-R), we noticed some deviation between our model's predicted score and the actual score. We argue that this behavior is incongruous because the sentences in the SICK dataset are fairly simple and there are fewer named entities and uncommon words in the vocabulary. We observe that our model is able to grasp the word ordering deviation correctly as in the first example, but fails in cases where both sentences do not share the same words as happened in the second and third examples. The third group displays the example predictions on the MSRP dataset. We find that our model is able to predict the correct paraphrases in most of the cases even though

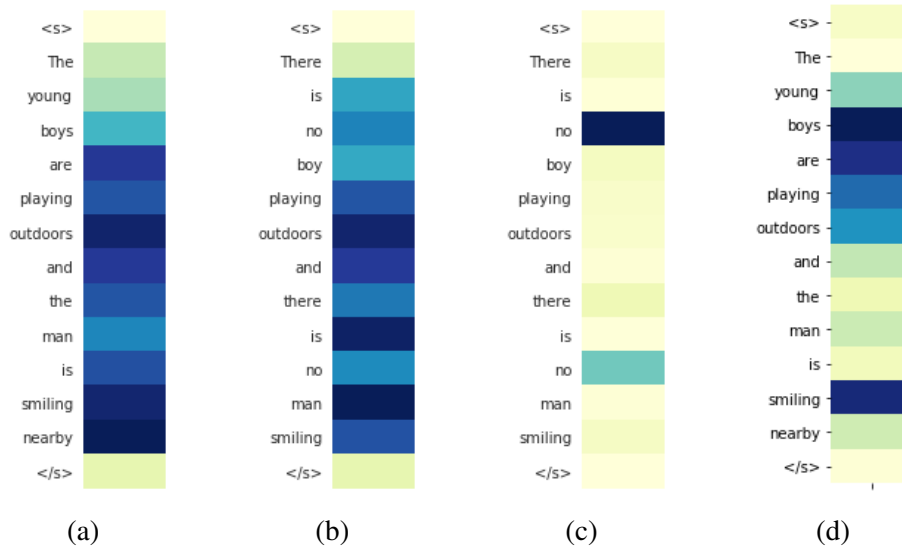


Figure 6.2: Visualization of attention weights. (a)  $Q=H$ ,  $K=V=H$  (b)  $Q=P$ ,  $K=V=P$  (c)  $Q=H$ ,  $K=V=P$  (d)  $Q=P$ ,  $K=V=H$  (**H** = Hypothesis and **P** = Premise)

the sentence pairs have different numbers but sometimes gives false positive predictions as happens in the third example where both sentences are talking about share prices. The AI2-8grade group shows that our model is very good at selecting false questions in almost every case and true questions in most of the cases but sometimes it makes false negative predictions if the evidence is quite different.

Figure 6.2 visualizes the amount of attention our model gives on different segments of the sentence pair when doing the comparison. As our model involves two self attentions and two cross attentions it outputs four attention vectors with different combinations of key, query, and value matrices. In the self attention case where the key, query, and value matrices are the same, our model puts more attention on keywords such as activities like ‘playing’ and ‘smiling’ and places like ‘outdoors’ and ‘nearby’. However the effectiveness of our model is more discernible with the results of cross attention. When we have the hypothesis as query, the premise as key, and the relationship being Contradiction, our model gives all of its attention to the word ‘no’ from the premise. On the other hand, when we have the premise as query and the hypothesis as key, our model puts all the attention on “young boys are playing outdoors” and ‘smiling’ which contain all of the information from the hypothesis, showing that adding multi-branch cross attention makes our model think more like a human when comparing two sentences thus making it very effective for any sentence pair modelling task.

## 6.6 Conclusion

Being able to independently model the context aware representations of words in a sentence with correct orderings, Transformer-based encoders are perfect candidates for parallelization. We have introduced a way to incorporate cross attention in the Transformer-based sentence encoders capturing the shared mutual information between sentences much like humans do. The attention heatmap clearly shows where attention is put when comparing two sentences. We evaluate our proposed models on four sentence pair modelling tasks, achieving state-of-the-art performance on two of them. Our sentence encoders do not take any task specific features into account making it more generalized. A thorough comparison against the available SOTA sentence encoders shows that, for classification tasks, Transformer-based encoders are superior to the LSTM and CNN-based ones. Adding cross attention on top of self attention usually makes it more robust in identifying the underlying relationship between the sentence pairs.

## Chapter 7

# Investigating Relational Recurrent Neural Networks with Variable Length Memory Pointer in Sentence Pair Modelling Tasks

This chapter is based on the paper titled “Investigating Relational Recurrent Neural Networks with Variable Length Memory Pointer in Sentence Pair Modelling Tasks” co-authored with Robert E. Mercer that appeared in The 33rd Canadian Conference on Artificial Intelligence (CAI 2020) [8].

Memory based neural networks have the ability to remember information for a longer period of time while modelling temporal data. The interaction of memories in these networks allows them to capture fairly complex linkages between data entities. To further improve LSTM’s information storage capability to extract good sentence representations, we encode a novel Relational Memory Core (RMC) inside it using the standard multi-head self attention mechanism with variable length memory pointer. The RMC replaces the cell states of the LSTM, which we now call  $\text{LSTM}_{RMC}$ , by creating new memories. Two claims are made about our improvement: It is important to expand the area on which the RMC operates to create the new memory as more data is seen with each time step, and the expanded area can be treated as a fixed size kernel having shared weights in the form of query, key, and value projection matrices. We design a novel sentence encoder using this  $\text{LSTM}_{RMC}$  and test our hypotheses on four NLP tasks: textual entailment, semantic relatedness, paraphrase identification, and question answer pairing. Our evaluation shows improvements over the standard LSTM and the Transformer encoder as well as state-of-the-art general sentence encoders.

## 7.1 Introduction

Humans use their memory system to access and retrieve important information irrespective of when they are actually perceived [113, 196, 198]. This memory is often understood as an informational processing system with explicit and implicit functioning that is made up of a sensory processor, short-term (or working) memory, and long-term memory [23]. In recent neural network based research, many successful attempts have been made to encode memory inside the LSTM cell [92, 196] and even designing core memory augmented neural networks [85, 195]. These memory based networks provide efficient information storing and retrieval capabilities reinforced by bounded computational cost, augmented memory capacities and able to surpass the vanishing gradient problem.

Recurrent neural networks (RNNs) [194] are the most widely used neural network for modelling sequence data having successors in the form of LSTM [92] and GRU [55] which have been successfully used in a variety of applications such as machine translation, sequence labelling, speech recognition, and question answering. However, these models have a locality bias present inside it because of its built-in state dependency nature. Also, a single cell state in LSTM should never be enough to crawl and store the information flowing through a sequence [196]. Convolutional neural networks (CNNs) [78] somehow mitigate this drawback using its kernels by computing a relation of the entities in a local receptive field. This is good for computing spatial correlations but to deal with temporal data one needs to built up memory which the CNNs cannot provide.

In terms of NLP temporal domain, it is necessary to analyze and compare phrases as well as sentence constituents seen at different time steps to extract the meaning. Attention mechanism [240] implicitly provides this by computing a weighted combination of sentence constituents which also helps in extracting good relational reasoning. It has been proved that, allowing vanilla LSTM's common hidden memory to access more of the previous representations using a relational memory core (RMC) gives good performance boost in tasks demanding particular types of temporal relational reasoning [196]. This RMC is designed using the standard multi-head self attention framework. Furthermore, this type of encoding of attention cell inside the tree version of LSTM also proved to be beneficial in building up semantically focussed phrase representations [13].

In this chapter, we design an improved relational memory core (RMC) having access to previously seen representations through a variable length memory pointer. Our idea is that the memory created at each time step should reflect the previously created representations whereas the LSTM gates should be updated only with the last one. We encode this new RMC as the cell state inside an LSTM cell and design a sentence encoder model to encode a pair of



sentences for the sentence pair modelling task. Evaluation done on textual entailment (TE), semantic relatedness (SR), paraphrase identification (PI), and question answer pairing (QAP) shows improvements over standard LSTM encoder on all of the tasks and best results on the textual inference and second best on question-answer pair tasks when compared against all state-of-the-art encoder models including the best models especially designed for the specific tasks.

## 7.2 Related Work

Memory based models are good at creating new memories based on attention. These have been used to modify standard and tree LSTMs. Sukhbaatar et al. [220] train a memory based neural network in an end-to-end fashion to solve a compartmentalization problem with a slot-based memory matrix. Linear addition of input and output at layer  $k$  as the input to layer  $k + 1$  allows the memories to interact or relate with one another. Santoro et al. [197] propose a plug-and-play neural network to perform relational reasoning task. It involves two MLPs as the composition functions. The first one computes a representation of each of the existing relations and the second operates on the linear addition of these prior representations. Ahmed et al. [13] encode a single head dot product attention block inside a tree-LSTM cell where a weighted combination of child hidden states is assigned as the parent hidden state in a bottom up recursive traversal of natural language grammar trees to obtain the sentence representation.

General purpose sentence encoders have been state-of-the-art on the four tasks of interest in this chapter. Cer et al. [46] propose a sentence encoder model based on the encoder portion of Transformer [240] and perform an element-wise sum of the encoded representations at each word position to get a fixed length sentence representation. They evaluate their model mostly on sentence classification tasks and prove the effectiveness of transfer learning at the sentence level compared to the word level. Conneau et al. [58] also propose a universal sentence representation model based on LSTMs where it is first trained on the Stanford Natural Language Inference task and then uses transfer learning. Evaluation is on a range of tasks including QAP, PI, and SR. Zhou et al. [270] propose a sentence pair ranking model where they encode attention in the tree structure of the hypothesis based on the sequential representation of the premise and vice versa. Lin et al. [135] propose a self-attentive sentence encoding model where they replace the pooling block with a self attention block on top of an LSTM encoder. Instead of extracting a vector representation, the authors use a matrix as the sentence representation where each row attends to different portions of the sentence. Zhao et al. [263] propose a self-adaptive hierarchical model which first extracts an intermediate representation of all possible phrases and finally takes the convex combination of them through gating. Socher et al. [208] propose

a recursive auto-encoder-based paraphrase identification model that first reconstructs each of the phrases from the tree representation of both sentences. Finally, they extract the sentence representation by doing a min-pooling operation over the transformed differences between the sentences.

## 7.3 The Model

We are motivated to improve the design principle of the current RMC [196] so that the model can effectively learn to categorize information and to perform better interactions among them. To achieve this, we extend the scope of the memory pointer in RMC by giving the self attention module more to explore. In this section we first explain how this modified RMC can be used as a module inside the standard LSTM cell. Following this, we explain the general working principle of an RMC and present the variation that we propose in terms of a variable length memory pointer. Finally, we conclude this section by designing a sentence pair modelling architecture using an RMC encoded LSTM ( $\text{LSTM}_{\text{RMC}}$ ).

Generally, in LSTM the gating mechanisms handle the storage of information from the input sequence into the cell state. All the gates inside an LSTM cell operate on this cell state to add, delete, or modify the already stored information. It starts with an input gate which selects what to update using a sigmoid layer using Eqn. 7.1. The forget gate stores how much information the network is allowed to forget from the previous cell state using Eqn. 7.2. Next, a tanh layer creates new candidate values to be added to the existing cell state using Eqn. 7.3. Finally, the new candidate values are scaled by the input gate values and summed with a filtered version of the candidate cell state values where the forget gate is working as the filter as shown in Eqn. 7.4. Finally, the next hidden state is just a tanh squashing of the cell state as shown in Eqn. 7.5.

$$\mathbf{i}^t = \sigma(\mathbf{W}^{(i)}\mathbf{x}^t + \mathbf{U}^{(i)}\mathbf{h}^{t-1} + \mathbf{b}^{(i)}) \quad (7.1)$$

$$\mathbf{f}^t = \sigma(\mathbf{W}^{(f)}\mathbf{x}^t + \mathbf{U}^{(f)}\mathbf{h}^{t-1} + \mathbf{b}^{(f)}) \quad (7.2)$$

$$\tilde{\mathbf{c}}^t = \tanh(\mathbf{W}^{(c)}\mathbf{x}^t + \mathbf{U}^{(c)}\mathbf{h}^{t-1} + \mathbf{b}^{(c)}) \quad (7.3)$$

$$\mathbf{c}^t = \mathbf{i}^t \cdot \tilde{\mathbf{c}}^t + \mathbf{f}_t \cdot \mathbf{c}^{t-1} \quad (7.4)$$

$$\mathbf{h}^t = \tanh(\mathbf{c}^t) \quad (7.5)$$

It is evident that the cell state is the one on which all the gates operate whenever a new input comes into play at each time step. Relational recurrent neural networks encode this cell state value as a memory designed using the Relational Memory Core (RMC). The RMC uses

`multi-head self attention` to allow memories to interact with each other. Below, we first explain the standard RMC pipeline which uses a fixed length memory pointer. Following this, we explain how we extend this memory interaction with a variable length memory pointer.

### 7.3.1 Using a Fixed Length Memory Pointer

In the classic setting [196], the network maintains a fixed length memory pointer which begins with a random memory  $M_{1 \times b \times d}$  which is then iteratively updated at each time step. The starting memory is initialized as either an identity matrix or a trivial random seed. It is to be noted that this initial memory size is dependent on the batch size  $b$ . At each time step  $t$ , it creates a new tensor  $\tilde{M}$  by concatenating the previous memory and a linear projection of the current input as follows,

$$\tilde{M}_{2 \times b \times d}^t = [M_{1 \times b \times d}^{t-1}; \mathbf{W}x_{1 \times b \times d}^t] \quad (7.6)$$

Following this, the `multi-head self attention` block is applied on this  $\tilde{M}$  to produce a new scaled tensor according to the self attention weights. These attention weights indicate how much information to take from the current input and the previous memory to create the next memory. This `multi-head self attention` block consists of three projection matrices  $\mathbf{W}^q$ ,  $\mathbf{W}^k$  and  $\mathbf{W}^v$  as network parameters. It first creates query,  $Q(= M\mathbf{W}^q)$ , and key,  $K(= M\mathbf{W}^k)$ , tensors by projecting  $\tilde{M}$  using the corresponding weight matrices. Following this, it applies the standard dot product attention  $A(= \text{softmax}(\frac{QK^T}{\sqrt{d_k}}))$  on this  $Q$  and  $K$  to get the scalar weights. Finally, the value,  $V(= M\mathbf{W}^v)$ , tensor gets scaled using these weights and the resultant tensor represents the weighted average of the original memory tensor  $\tilde{M}$ .

A residual connection is then added to the resultant tensor  $\tilde{M}$  with its prior version  $M(= \tilde{M} + M)$ . This residual connection allows the network to decide whether to carry forward the previous memory as the next memory at time step  $t + 1$  or to continue with the weighted version of the memory created using the `multi-head self attention` block. This resultant tensor is then passed through a `Layer-Normalization` block where the activations of neurons in the current layer are normalized to get a comparatively faster training. This normalization process starts by first calculating the mean and the standard deviation of the input at the current time step as follows,

$$\mu^t = \frac{1}{H} \sum_{i=1}^H x_i^t \quad \sigma^t = \sqrt{\frac{1}{H} \sum_{i=1}^H (x_i^t - \mu^t)^2} \quad (7.7)$$

Following this, we use this mean  $\mu$  and standard deviation  $\sigma$  to normalize the given input  $x$  and further scale each dimension of the resultant tensor with two new parameters  $\gamma$  and  $\beta$ . However, unlike the original implementation, we do not use separate versions of these new parameters according to the cardinal dimension of  $\tilde{M}$ . Instead we maintain the same parameter

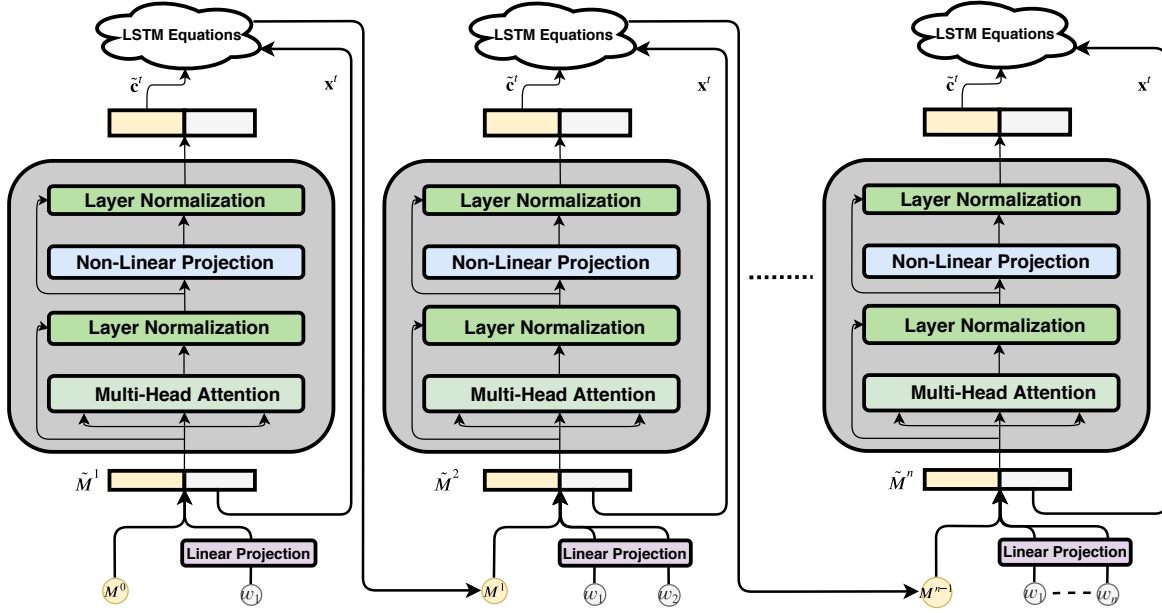


Figure 7.1: Sample encoding of a sentence using RMC with variable length memory pointer (from left to right direction). Rightmost  $M$  represents the sentence representation. In order to be aligned with BLSTM working principle and get a right to left representation, the word order can be flipped ( $w_n, w_{n-1}, \dots, w_1$ ).

over all the dimensions. The justification for doing this is provided in Section 7.3.2. We summarize these steps using Eqn. 7.8 as follows,

$$h^t = \gamma \odot \left( \frac{x^t - \mu^t}{\sigma^t + \epsilon} \right) + \beta \quad (7.8)$$

Here,  $\epsilon$  is a hyperparameter. Next,  $n$  non-linear projections of  $h^t$  are taken followed by a residual connection. Eqn. 7.9 summarizes this step for  $n = 2$ .

$$X = \mathbf{f}(\mathbf{W}^{(1)} \mathbf{f}(h^t \mathbf{W}^{(2)})) + h^t \quad (7.9)$$

Here,  $\mathbf{f}$  is an activation function and we use ReLU as  $\mathbf{f}$  for our experiments. The resultant tensor  $X$  from Eqn. 7.9 has shape  $2 \times b \times d$ . We then extract the new memory by splitting out this tensor at the cardinal dimension and take the first item as follows,

$$M_{1 \times b \times d}^t = X_{1 \times b \times d}^1 \quad (7.10)$$

This new memory can act as the candidate cell state as in Eqn. 7.3. This allows us to discard Eqn. 7.3 and modify Eqn. 7.4 as follows,

$$\mathbf{c}^t = i^t \cdot \mathbf{M}^t + f^t \cdot \mathbf{M}^{t-1} \quad (7.11)$$

Returning to the LSTM equations [7.1](#)-[7.3](#),  $x^t$  is replaced with the projected input ( $= \mathbf{W}x^t$ ) from Eqn. [7.6](#).

### 7.3.2 Using a Variable Length Memory Pointer

Unlike Eqn. [7.6](#) from Section [7.3.1](#) where the previous memory is always concatenated with the projected input at time step  $t$ , in this setting, we continually expand the area on which the multi-head self attention operates as shown in Figure [7.1](#). At time step  $t$ , we perform this memory expansion by concatenating all the projected inputs from time step  $t$  to 1 with the previous memory as follows,

$$\tilde{\mathbf{M}}_{(t+1) \times b \times d}^t = [\mathbf{M}_{1 \times b \times d}^{t-1}; \mathbf{W}x_{1 \times b \times d}^t; \mathbf{W}x_{1 \times b \times d}^{t-1}; \dots; \mathbf{W}x_{1 \times b \times d}^1] \quad (7.12)$$

It is to be noted that the weight ( $\mathbf{W}$ ) is shared for projecting all the words that have been seen until the current time step  $t$ . The resulting tensor  $\tilde{\mathbf{M}}^t$  has shape  $(t+1) \times b \times d$ . Following this, we apply multi-head self attention, Layer-Normalization and the necessary projections to get a new transformed representation  $X$  as in Eqn. [7.9](#). As we keep concatenating the inputs all the way to time step  $t = 1$  at the end of the tensor and keep the memory always at the first index, we can easily use Eqn. [7.10](#) and extract the new memory as before. Finally, as done in Section [7.3.1](#), we use  $\mathbf{W}x^t$  as  $x^t$  in the LSTM equations [7.1](#)-[7.3](#).

Although it is possible to give the attention block access to everything from time step  $t$  to 1 during each input, access is limited to a maximum window size in the experiments. As the attention block sees a variable length tensor at each time step, a fixed set of  $\gamma$  and  $\beta$  in the LayerNormalization block cannot be maintained as this makes the gradient accumulation of few parameters unstable.

### 7.3.3 Modelling Sentence Pairs using RMC

The standard pipeline for sentence pair modelling starts with encoding both the hypothesis and premise sentences as vectors using a neural sentence encoder followed by a matching layer where the corresponding vector representations are compared with a similarity metric. Proceeding in this way, our model first traverses each sentence as a sequence of  $T$  words  $\{x^t\}_{t=1, \dots, T}$  from both left to right and right to left and generates two memory representations at the last time step,  $\vec{h}_T$  and  $\overleftarrow{h}_T$ . During input, it considers the vector representation of each word,

$x^t$ , in the sentence from a pre-trained word embedding model and these representations are not further trained with the network parameters.

$$\begin{aligned}\vec{h}_t &= \overrightarrow{\text{LSTM}_{RMC}}(x^1, \dots, x^T) \\ \overleftarrow{h}_t &= \overleftarrow{\text{LSTM}_{RMC}}(x^1, \dots, x^T)\end{aligned}\tag{7.13}$$

Next we create two vectors, one represents the encoded hypothesis sentence  $a$  and one represents the encoded premise sentence  $b$ . In each case, the vector is a concatenation of  $\vec{h}_T$  and  $\overleftarrow{h}_T$  of the respective encoded sentence. We compute a range of features such as difference, element-wise product, and average for the tuple  $\langle a, b \rangle$ . These features are intended to capture the inference between components in the tuples and catch induction relationships such as logical inconsistency. These features are then concatenated with the original representations,  $a$  and  $b$ , and we create a new tensor representation from this as follows,

$$F = \text{makeTensor}([a; b; |a - b|; a * b; (a + b)/2])\tag{7.14}$$

Next, we perform an element-wise multiplication of this feature tensor with a learned parameter vector  $\beta \in \mathbb{R}^n$  having the constraint  $\sum \beta_i = 1$ . The final inference relationship vector results from an element-wise weighted addition of each feature position as follows,

$$x = \sum_{i=1}^5 F_i \beta_i\tag{7.15}$$

The resultant vector is then projected into the space of classes  $y$ , through a series of fully connected layers as follows,

$$P(y|\mathbf{X}) = \sigma(\mathbf{W}_1 \sigma(\mathbf{W}_2 x + \mathbf{b}_2) + \mathbf{b}_1)\tag{7.16}$$

$P(\cdot)$  represents the model's predicted scores over classes using the feature vector  $x$ .  $\mathbf{W}$ 's and  $\mathbf{b}$ 's are the weights and biases of the classifier network. The model is trained by optimizing a task specific loss function as follows,

$$H(p, q) = - \sum_{i=1}^N Q(y_i) \log(P(y_i))\tag{7.17}$$

$Q(\cdot)$  is a one hot vector representation of the true probability distribution of the actual label. The multiplication of  $P(\cdot)$  and  $Q(\cdot)$  indicates how close the model's decision over the actual class is with respect to the true distribution. Finally, this objective is negated and minimized rather than maximized.

For the classification tasks, the aforementioned loss function works fine but for the regression task, to predict a score, we first compute a target distribution  $\tilde{Q}$  as a function of the actual score  $y$  as follows,

$$\tilde{Q}_i(y) = \begin{cases} y - \lfloor y \rfloor, & \text{if } i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & \text{if } i = \lfloor y \rfloor \\ 0, & \text{otherwise} \end{cases} \quad (7.18)$$

The distance between the predicted distribution  $P(\cdot)$  and the distribution of the ground truth  $\tilde{Q}(\cdot)$  is minimized using the KL-divergence loss function as follows,

$$H(p, \tilde{q}) = \frac{1}{N} \sum_{k=1}^N \text{KL}(\tilde{Q}(y_k) \| P(y_k)) + \frac{\lambda}{2} \|\theta\|^2 \quad (7.19)$$

## 7.4 Experimental Setup

In this section, we detail the experimental setup used for evaluation. We first describe our training corpora as well as all of the benchmarks used in other standard sentence pair modelling studies. Following this, we explain the technical details of our proposed architectures along with their hyper-parameter settings.

**Datasets:** Model evaluation uses the following datasets:

- **MSRP:** Paraphrase identification: given a pair of sentences, the task is to identify whether or not they are paraphrases of each other [69]. Train: 4076, Valid: N/A, Test: 1725.
- **SICK:** Natural language inference and semantic relatedness: the dataset contains sentences derived from video and image annotations and it comes with two tasks. The first task is to classify a given sentence pair into three classes: Entailment, Neutral and Contradiction. The second task is to assign a score between 1 and 5 for a sentence pair based on their semantic relatedness [151]. Train: 4500, Valid: 500, Test: 4927.
- **AI2-8grade:** Question-answer pair modelling: this true-false question selection task consists of sentence pairs with one being the question and the other being the evidence formed by replacing the *wh* in the question by the answer [30]. Train: 12689, Valid: 2483, Test: 11359.

**Experimental Setup:** Word vectors are initialized with the 300 dimension GloVe embeddings [183] and are not updated during training. To smooth the update, the gradients are divided by  $B^2$

Config	Value	Config	Value
Initial learning rate	<b>0.1</b> / 0.05 / 0.001	maxNorm	<b>5</b>
Batch size	10 / <b>16</b> / 25	Learning rate decay	<b>0.99</b>
No. of Attention layers	<b>1</b> / 2 / 3	Dropout FC	<b>0.0375 - 0.5</b>
Hidden dimension	256, 512, <b>1024</b>	No. of Heads	<b>8</b>
Word embedding	Glove 300D	$W^q, W^k, W^v$ dimension	<b>128</b>

Table 7.1: Hyper-parameters used for the experiments (in boldface) and the ranges that were searched during tuning.

where  $B$  is the batch size. The learning rate decays proportional to  $\frac{maxNorm}{\sqrt{\sum_{i=1}^k \|\nabla \theta_i^2\|}}$  if  $\sqrt{\sum_{i=1}^k \|\nabla \theta_i^2\|}$  is more than  $maxNorm$ .

Table 7.1 shows the hyper-parameter settings used during the experiments. We train our models on a GeForce GTX 1080Ti GPU with ‘Adam’, ‘AdaDelta’, ‘AdaGrad’ and ‘SGD’ optimizers. Results in the next section are reported using ‘SGD’. Models are implemented in PyTorch 0.4.1 under the Linux environment.

## 7.5 Experimental Results and Analysis

In this section, we present the detailed results obtained with our RMC based sentence encoder and compare with some of the top performing models for the four sentence pair modelling tasks. Finally, we conclude this section by giving some insight into the performance of our model by analyzing the attention weights.

Table 7.2 displays our model’s overall performance on the four corpora in terms of task specific evaluation metrics. The first group contains the results of  $LSTM_{RMC}$  in the classic fixed pointer (FLMP) and our variable pointer (VLMP) configurations. In order to analyze the strength of our model, the second group of this table reports the results of some currently available top performing sentence encoders with their best hyper-parameters. The last group contains the results of some preeminent but less general models. Notably, on the MSRP task, the VLMP model is only behind RAE [208]. It is specifically designed, using additional grammatical information, for this task. Our VLMP model achieves better accuracy (75.89%) than the FLMP model (74.67%). This indicates that for paraphrase identification, it is more important to see the word overlap in a local context than looking at everything right from the beginning. On AI2-8grade dataset, it is standard to report both the development and test set accuracy as we also do so. On development set, we are only behind InferSent which uses an additional pooling block and BiLSTM with additional projection Layer [58] by a margin of 1.45% and 0.28% respectively. However, on test set, our model is only behind BiLSTM projec-



Model	MSRP	AI2-8grade		SICK-E	SICK-R
	Acc.	Dev Acc.	Test Acc.	Acc.	$r$ /MSE
LSTM <sub>RMC</sub> + FLMP	74.67	76.65	74.72	85.38	0.8107/0.3452
LSTM <sub>RMC</sub> + VLMP (WINDOW SIZE=5)	75.89	76.21	74.72	84.28	0.8440/0.2925
INFERSENT [58] †	74.46	78.10	74.10	84.62	0.8563/0.2732
LSTM [58] †	70.74	76.33	74.24	76.80	0.8291/0.3244
BiLSTM PROJECTION LAYER [58] †	74.24	76.93	75.15	85.20	0.8037/0.3667
BiGRU LAST ENCODER [58] †	70.46	76.25	74.46	81.47	0.8317/0.3147
INNER ATTENTION [135] †	69.74	76.61	74.32	72.01	0.7863/0.3944
CONVNET ENCODER [263] †	73.96	76.29	75.15	83.82	0.8520/0.2806
TRANSFORMER ENCODER [46]	74.96	-	-	81.15	-/0.5241
SEQ-LSTMS [270]	71.70	71.80	63.30	-	0.8528/0.2831
SEQ-GRUs [270]	71.80	72.10	62.40	-	0.8595/0.2689
TREE LSTM [270]	73.50	74.60	69.10	-	0.8664/0.2610
TREE LSTM + ATTN. [270]	75.80	76.20	72.50	-	0.8730/0.2426
TREE GRU [270]	73.96	75.20	70.60	-	0.8672/0.2573
TREE GRU + ATTN. [270]	74.80	76.40	72.10	-	0.8701/0.2524
RAE [208]	76.80	-	-	-	-
COMBINE-SKIP + FEATS [111]	75.80	-	-	-	-

Table 7.2: Performance of our model on different tasks compared to some top performing models. We mark models that we implemented as †. **FLMP** = Fixed length memory pointer, **VLMP** = Variable length memory pointer

tion Layer [58] and ConvNet Encoder [263] by a margin of 0.43%. As we can see, we manage to close the performance gap on test set which is more aligned with the real world samples. Our application of classic FLMP on the SICK-E task achieves state of the art accuracy of 85.38% going past the better models on AI2-8grade dataset by quite a good margin (Infersent - 0.76%, BiLSTM projection Layer - 0.18%, and ConvNet Encoder - 1.56%). On SICK-R task, we are getting, 0.8440 Pearson’s  $r$  and 0.2925 MSE, which puts us on the top three sequential models on this task. It is to be noted that, although the tree based models are doing much better than the sequential models on this dataset, both versions of our proposed model are superior to these tree based models on the other tasks. Finally, these scores clearly tell that, our model is always at the top three general purpose sentence encoders on all of the four different corpora.

Figure 7.2 visualizes the amount of attention our model gives on different segments of the sentence pair when doing the comparison. As we keep the window size to 5, at each time step, our model puts attention on a 5-gram and a memory. It is clear that, every time, the model reads a stop-word, it puts less attention on it and the probability gets distributed over the other content words inside the window. It is to be noted that, in the left sentence, the phrase “Virginia attorney general’s office” is the content phrase and because of this at the last time step, the attention on the memory is less significant compared to the attention on the 5-grams.

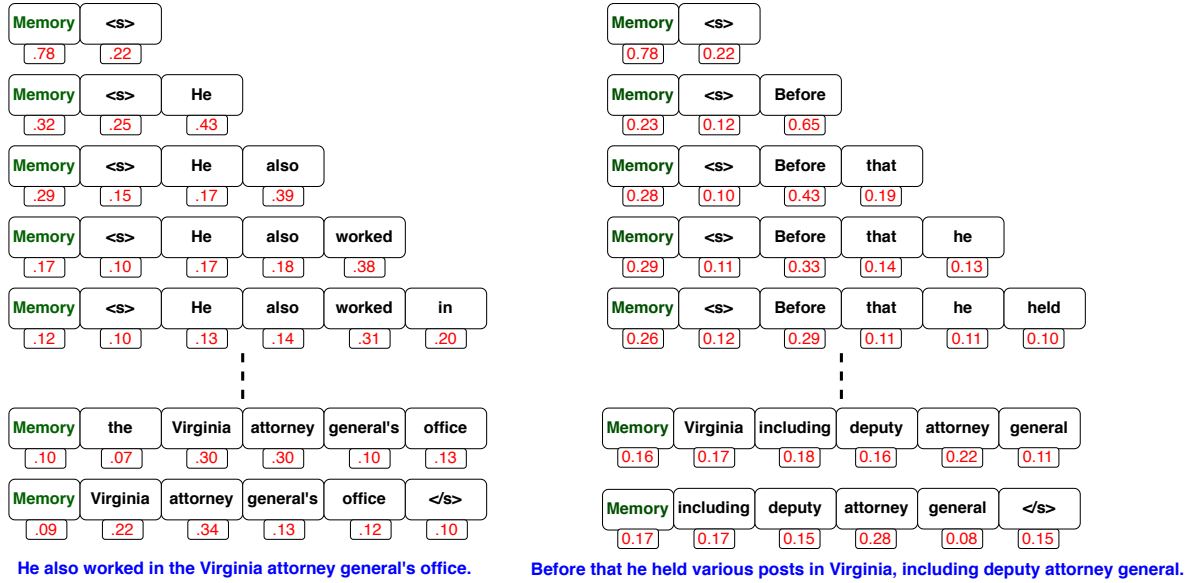


Figure 7.2: Change in attention weights as the attention window shifts over time (memory pointer length / window size = 5).

On the other hand, in the right sentence, the reading of the content phrase gets done at the second last time step. This is why the memory gets comparatively higher weight at the last time step. However, the content words in the window still gets quite a good amount of weights.

## 7.6 Conclusion and Future Work

In this chapter, we have modified the classical RMC with the concept of a variable length memory pointer so it uses a non-local context to compute an enhanced memory during input of each word in a sentence. We encode this improved RMC inside an LSTM cell and design a sentence pair modelling architecture where we evaluate our proposed model on four different tasks. We show that our model achieves on par performance on most of the tasks and state-of-the-art performance on one of them. Our sentence encoder does not take any task specific features into account making it more general. Being equipped with this new memory, our model interprets very well how the attention shifting is done as we gradually read and compare two sentences side by side. Our limited experiments show that the memory pointer length does not follow a uniform pattern across all datasets, making it an interesting area to investigate for our future studies.

# Chapter 8

## You Only Need Attention to Traverse Trees

This chapter is based on the paper titled “You Only Need Attention to Traverse Trees” co-authored with Muhammad Rifayat Samee, and Robert E. Mercer that appeared in the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019) [14].

In recent NLP research, a topic of interest is universal sentence encoding, sentence representations that can be used in any supervised task. At the word sequence level, fully attention-based models suffer from two problems: a quadratic increase in memory consumption with respect to the sentence length and an inability to capture and use syntactic information. Recursive neural nets can extract very good syntactic information by traversing a tree structure. To this end, we propose *Tree Transformer*, a model that captures phrase level syntax for constituency trees as well as word-level dependencies for dependency trees by doing recursive traversal only with attention. Evaluation of this model on four tasks gets noteworthy results compared to the standard transformer and LSTM-based models as well as tree-structured LSTMs. Ablation studies to find whether positional information is inherently encoded in the trees and which type of attention is suitable for doing the recursive traversal are provided.

### 8.1 Introduction

Following the breakthrough in NLP research with word embeddings by [156], recent research has focused on sentence representations. Having good sentence representations can help accomplish many NLP tasks because we eventually deal with sentences, e.g., question answering, sentiment analysis, semantic similarity, and natural language inference.

Most of the existing task specific sequential sentence encoders are based on recurrent neural

nets such as LSTMs or GRUs [59, 135, 141]. All of these works follow a common paradigm: use an LSTM/GRU over the word sequence, extract contextual features at each time step, and apply some kind of pooling on top of that. However, a few works adopt some different methods. [111] propose a skip-gram-like objective function at the sentence level to obtain the sentence embeddings. [143] reformulate the task of predicting the next sentence given the current one into a classification problem where instead of a decoder they use a classifier to predict the next sentence from a set of candidates.

The attention mechanism adopted by most of the RNN based models require access to the hidden states at every time step [120, 248]. These models are inefficient and at the same time very hard to parallelize. To overcome this, [174] propose a fully attention-based neural network which can adequately model the word dependencies and at the same time is parallelizable. [240] adopt the multi-head version in both the encoder and decoder of their Transformer model along with positional encoding. [3] propose a multi-branch attention framework where each branch captures a different semantic subspace and the model learns to combine them during training. [46] propose an unsupervised sentence encoder by leveraging only the encoder part of the Transformer where they train on the large Stanford Natural Language Inference (SNLI) corpus and then use transfer learning on smaller task specific corpora.

Apart from these sequential models, there has been extensive work done on the tree structure of natural language sentences. [212, 214, 216] propose a family of recursive neural net (RvNN) based models where a composition function is applied recursively bottom-up on children nodes to compute the parent node representation until the root is reached. [228] propose two variants of sequential LSTM, child sum tree LSTM and N-ary tree LSTM. The same gating structures as in standard LSTM are used except the hidden and cell states of a parent are dependent only on the hidden and cell states of its children.

Recently, [204] propose a Parsing-Reading-Predict Network (PRPN) which can induce syntactic structure automatically from an unannotated corpus and can learn a better language model with that induced structure. Later, [94] test this PRPN under various configurations and datasets and further verified its empirical success for neural network latent tree learning. [242] also validate the effectiveness of two latent tree based models but found some issues such as being biased towards producing shallow trees, inconsistencies during negation handling, and a tendency to consider the last two words of a sentence as constituents.

In this chapter, we propose a novel recursive neural network architecture consisting of a decomposable attention framework in every branch. We call this model *Tree Transformer* as it is solely dependent on attention. In a subtree, the use of a composition function is justified by a claim of [212, 214]. In this work, we replace this composition function with an attention module. While [212, 214] consider only the child representations for both dependency and con-

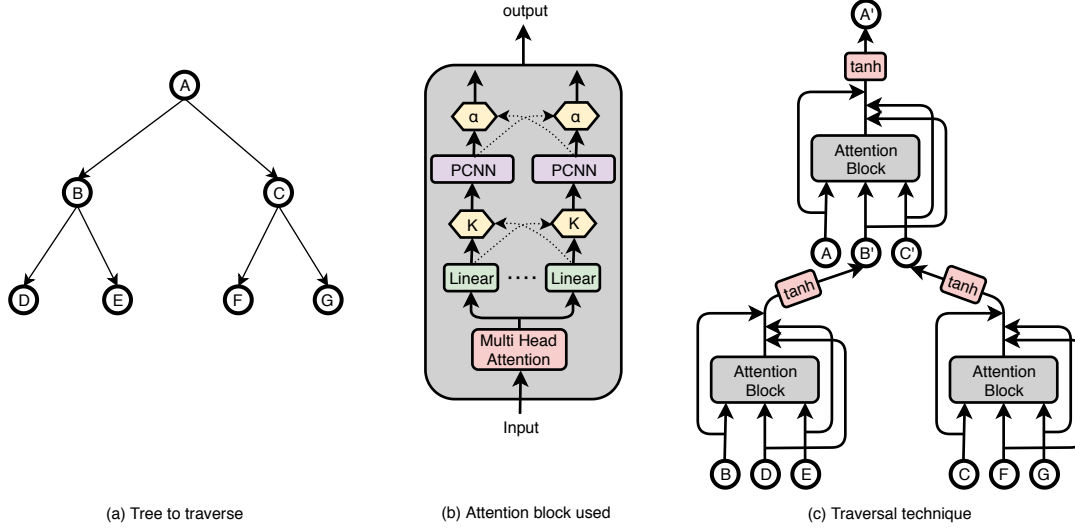


Figure 8.1: Attention over the tree structure

stitency syntax trees, in this work, for dependency trees, the attention module takes both the child and parent representations as input and produces weighted attentive copies of them. For constituency trees, as the parent vector is entirely dependent on the upward propagation, the attention module works only with the child representations. Our extensive evaluation proves that our model is better or at least on par with the existing sequential (i.e., LSTM and Transformer) and tree structured (i.e., Tree LSTM and RvNN) models.

## 8.2 Proposed Model

Our model is designed to address the following general problem. Given a dependency or constituency tree structure, the task is to traverse every subtree within it attentively and infer the root representation as a vector. Our idea is inspired by the RvNN models from [212][214][216] where a composition function is used to transform a set of child representations into one single parent representation. In this section, we describe how we use the attention module as a composition function to build our *Tree Transformer*. Figure 8.1 gives a sketch of our model.

A dependency tree contains a word at every node. To traverse a subtree in a dependency tree, we look at both the parent and child representations ( $\mathbf{X}_d$  in Eqn. 8.1). In contrast, in a constituency tree, only leaf nodes contain words. The non-terminal vectors are calculated only after traversing each subtree. Consequently, only the child representations ( $\mathbf{X}_c$  in Eqn. 8.1) are

considered.

$$\mathbf{X}_d = \begin{bmatrix} \mathbf{p}_v \\ \mathbf{c}_{1_v} \\ \vdots \\ \mathbf{c}_{n_v} \end{bmatrix} \quad \mathbf{X}_c = \begin{bmatrix} \mathbf{c}_{1_v} \\ \mathbf{c}_{2_v} \\ \vdots \\ \mathbf{c}_{n_v} \end{bmatrix} \quad (8.1)$$

Here,  $\mathbf{p}_v$  is the parent representation and the  $\mathbf{c}_{i_v}$ 's are the child representations. For both of these trees, Eqn. 8.2 computes the attentive transformed representation.

$$\tilde{\mathbf{P}} = \mathbf{f}(\mathbf{x}), \quad \text{where } \mathbf{x} \in \{\mathbf{X}_d, \mathbf{X}_c\} \quad (8.2)$$

Here,  $\mathbf{f}$  is the composition function using the multi-branch attention framework [3]. This multi-branch attention is built upon the multi-head attention framework [240] which further uses scaled dot-product attention [174] as the building block. It operates on a query  $\mathbf{Q}$ , key  $\mathbf{K}$  and value  $\mathbf{V}$  as follows

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{\mathbf{d}_k}}\right)\mathbf{V} \quad (8.3)$$

where  $\mathbf{d}_k$  is the dimension of the key. As we are interested in  $n$  branches,  $n$  copies are created for each  $(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ , converted to a 3D tensor, and then a scaled dot-product attention is applied using

$$\mathbf{B}_i = \text{Attention}(\mathbf{Q}_i \mathbf{W}_i^Q, \mathbf{K}_i \mathbf{W}_i^K, \mathbf{V}_i \mathbf{W}_i^V) \quad (8.4)$$

where  $i \in [1, n]$  and the  $\mathbf{W}_i$ 's are the parameters that are learned. Note that  $\mathbf{W}_i^Q, \mathbf{W}_i^K$  and  $\mathbf{W}_i^V \in \mathbb{R}^{d_m \times d_k}$ . Instead of having separate parameters for the transformation of leaves, internal nodes and parents [212], we keep  $\mathbf{W}_i^Q, \mathbf{W}_i^K$  and  $\mathbf{W}_i^V$  the same for all these components. We then project each of the resultant tensors into different semantic sub-spaces and employ a residual connection [90, 218] around them. Lastly, we normalize the resultant outputs using a layer normalization block [22] and apply a scaling factor  $\kappa$  to get the branch representation. All of these are summarized in Eqn. 8.5

$$\bar{\mathbf{B}}_i = \text{LayerNorm}(\mathbf{B}_i \mathbf{W}_i^b + \mathbf{B}_i) \times \kappa_i \quad (8.5)$$

Here,  $\mathbf{W}_i^b \in \mathbb{R}^{n \times d_v \times d_m}$  and  $\kappa \in \mathbb{R}^n$  are the parameters to be learned. Note that we choose  $\mathbf{d}_k = \mathbf{d}_q = \mathbf{d}_v = \mathbf{d}_m/n$ . Following this, we take each of these  $\bar{\mathbf{B}}$ 's and apply a convolutional neural network (see Eqn. 8.6) consisting of two transformations on each position separately

and identically with a *ReLU* activation ( $R$ ) in between.

$$\text{PCNN}(x) = \text{Conv}(R(\text{Conv}(\mathbf{x}) + \mathbf{b}_1)) + \mathbf{b}_2 \quad (8.6)$$

We compute the final attentive representation of these subspace semantics by doing a linearly weighted summation (see Eqn. 8.7) where  $\alpha \in \mathbb{R}^n$  is learned as a model parameter.

$$\text{BranchAttn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \sum_{i=1}^n \alpha_i \text{PCNN}(\bar{\mathbf{B}}_i) \quad (8.7)$$

Lastly, we employ another residual connection with the output of Eqn. 8.7, transform it non-linearly and perform an element-wise summation (EwS) to get the final parent representation as in Eqn. 8.8

$$\tilde{\mathbf{P}} = \text{EwS}(\tanh((\tilde{\mathbf{x}} + \mathbf{x})\mathbf{W} + b)) \quad (8.8)$$

Here,  $\mathbf{x}$  and  $\tilde{\mathbf{x}}$  depict the input and output of the attention module.

### 8.3 Experiments

In this section, we present the effectiveness of our *Tree Transformer* model by reporting its evaluation on four NLP tasks. We present a detailed ablation study on whether positional encoding is important for trees and also demonstrate which attention module is most suitable as a composition function for the recursive architectures.

**Experimental Setup:** We initialize the word embedding layer weights with GloVe 300-dimensional word vectors [183]. These embedding weights are not updated during training. In the multi-head attention block, the dimension of the query, key and value matrices are set to 50 and we use 6 parallel heads on each input. The multi-branch attention block is composed of 6 position-wise convolutional layers. The number of branches is also set to 6. We use two layers of convolutional neural network as the composition function for the PCNN layer. The first layer uses 341  $1d$  kernels with no dropout and the second layer uses 300  $1d$  kernels with dropout 0.1.

During training, the model parameters are updated using the Adagrad algorithm [72] with a fixed learning rate of 0.0002. We trained our model on an Nvidia GeForce GTX 1080 GPU and used PyTorch 0.4 for the implementation under the Linux environment.

**Datasets:** Evaluation is done on four tasks: the Stanford Sentiment Treebank (SST) [214] for sentiment analysis, Sentences Involving Compositional Knowledge (SICK) [150] for semantic relatedness (-R) and natural language inference (-E), and the Microsoft Research Paraphrase (MSRP) corpus [70] for paraphrase identification.

The samples in the SST dataset are labelled for both the binary and the 5-class classification task. In this work we are using only the binary classification labels. The MSRP dataset is labelled with two classes. The samples in the SICK dataset are labelled for both the 3-class SICK-E classification task and the SICK-R regression task which uses real-valued labels between 1 and 5. Instead of doing a regression on SICK-R to predict the score, we are using the same setup as [228] who compute a target distribution  $p$  as a function of the predicted score  $y$  given by Eqn. 8.9

$$\tilde{p}_i = \begin{cases} y - \lfloor y \rfloor, & \text{if } i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & \text{if } i = \lfloor y \rfloor \\ 0, & \text{otherwise} \end{cases} \quad (8.9)$$

The SST dataset includes already generated dependency and constituency trees. As the other two datasets do not provide tree structures, we parsed each sentence using the Stanford dependency and constituency parser [148].

For the sentiment classification (SST), natural language inference (SICK-E), and paraphrase identification (MSRP) tasks, accuracy, the standard evaluation metric, is used. For the semantic relatedness task (SICK-R), we are using mean squared error (MSE) as the evaluation metric.

We use KL-divergence as the loss function for SICK-R to measure the distance between the predicted and target distribution. For the other three tasks, we use cross entropy as the loss function.

Table 8.1 shows the results of the evaluation of the model on the four tasks in terms of task specific evaluation metrics. We compare our *Tree Transformer* against tree structured RvNNs, LSTM based, and Transformer based architectures.

To do a fair comparison, we implemented both variants of Tree LSTM and Transformer based architectures and some of the RvNN and LSTM based models which do not have reported results for every task. Instead of assessing on transfer performance, the evaluation is performed on each corpus separately following the standard train/test/valid split.

For SICK-E, our model achieved 82.95% and 82.72% accuracy with dependency and constituency tree, respectively, which is on par with DT-LSTM (83.11%) as well as CT-LSTM (82.00%) and somewhat better than the standard Transformer (81.15%). As can be seen, all of the previous recursive architectures are somewhat inferior to the *Tree Transformer* results.

For SICK-R, we are getting .2774 and .3012 MSE whereas the reported MSE for DT-LSTM and CT-LSTM are .2532 and .2734, respectively. However, in our implementation of those models with the same hyperparameters, we haven't been able to reproduce the reported results.



Types of Models	Model	SICK-E (Acc.)	SICK-R (MSE)	SST (Acc.)	MSRP (Acc.)
Tree Structured	SDT-RNN [212]	-	.3848	-	-
	RAE [209]	-	-	82.40	76.80
	MV-RNN [211]	58.14 †	-	82.90	66.91 †
	RNTN [216]	59.42 †	-	85.40	66.91 †
	DT-RNN [212]	63.38 †	.3822	86.60	67.51 †
	DT-LSTM [228]	83.11 †	.2532/.2625 †	85.70/85.10 †	72.07 †
	CT-LSTM [228]	82.00 †	.2734/.2891 †	88.00/87.27 †	70.07 †
LSTM	LSTM [228]	76.80	.2831	84.90	71.70
	Bi-LSTM [228]	82.11 †	.2736	87.50	72.70
	2-layer LSTM [228]	78.54 †	.2838	86.30	69.35 †
	2-layer Bi-LSTM [228]	79.66 †	.2762	87.20	70.40 †
	Infersent [59]	84.62	.2732	86.00	74.46
Transformer	USE_T [46]	81.15	.5241 †	85.38	74.96 †
	USE_T+DAN [46]	-	-	86.62	-
	USE_T+CNN [46]	-	-	86.69	-
Tree Transformer	DTT	82.95	.2774	83.12	70.34
	CTT	82.72	.3012	86.66	71.73

Table 8.1: Performance comparison of the *Tree Transformer* against some state-of-the-art sentence encoders. Models that we implemented are marked with †.

Instead we ended up getting .2625 and .2891 MSE for DT-LSTM and CT-LSTM, respectively. On this task, our model is doing significantly better than the standard Transformer (.5241 MSE).

On the SST dataset, our model (86.66% Acc.) is again on par with tree LSTM (87.27% Acc.) and better than Transformer (85.38% Acc.) as well as Infersent (86.00% Acc.)<sup>1</sup>

On the MSRP dataset, our dependency tree version (70.34% Acc.) is below DT-LSTM (72.07% Acc.). However, for the constituency tree version, we are getting better accuracy (71.73%) than CT-LSTM (70.07%). It is to be noted that all of the sequential models, i.e., Transformer, Infersent and LSTMs, are doing better compared to the tree structured models on this paraphrase identification task.

Since positional encoding is a crucial part of the standard Transformer, Table 8.2 presents its effect on trees. In constituency trees, positional information is inherently encoded in the tree structure. However, this is not the case with dependency trees. Nonetheless, our experiments suggest that for trees, positional encoding is irrelevant information as the performance drops in all but one case. We also did an experiment to see which attention module is best suited as a composition function and report the results in Table 8.3. As can be seen, in almost all the

<sup>1</sup>The official implementation available at <https://github.com/facebookresearch/InferSent> is used. Reported hyperparameters are used except LSTM hidden state, 1024d is chosen due to hardware limitations.

Model	PE	SICK-E	SICK-R	SST	MSRP
DTT	On	78.58	.3383	83.03	69.01
	Off	82.28	.2774	83.12	70.34
CTT	On	81.83	.3088	83.96	71.73
	Off	82.72	.3012	86.66	68.62

Table 8.2: Effect of positional encoding (PE).

Model	S/M/B	SICK-E	SICK-R	SST	MSRP
DTT	S	82.95	.3004	81.71	68.62
	M	82.86	.2955	82.97	69.07
	B	82.28	.2774	83.12	70.34
CTT	S	80.17	.4657	84.58	69.35
	M	79.66	.4346	83.74	70.01
	B	82.72	.3012	86.32	71.73

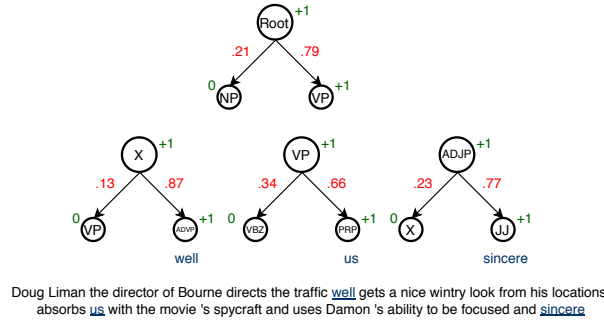
Table 8.3: Effect of different attention modules as a composition function. **S**: single-head attention, **M**: multi-head attention, **B**: multi-branch attention.

Figure 8.2: Attentive tree visualization (CTT)

cases, multi-branch attention has much better performance compared to the other two. This gain by multi-branch attention is much more significant for CTT than for DTT.

Figure 8.2 visualizes how our CTT model puts attention on different phrases in a tree to compute the correct sentiment. Space limitations allow only portions of the tree to be visualized. As can be seen, the sentiment is positive (+1) at the root and the model puts more attention on the right branch as it has all of the positive words, whereas the left branch (NP) is neutral (0). The bottom three trees are the phrases which contain the positive words. The model again puts more attention on the relevant branches. The words ‘well’ and ‘sincere’ are inherently positive. In the corpus the word ‘us’ is tagged as positive for this sentence.

## 8.4 Conclusion

In this chapter, we propose *Tree Transformer* which successfully encodes natural language grammar trees utilizing the modules designed for the standard Transformer. We show that we can effectively use the attention module as the composition function together with grammar information instead of just bag of words and can achieve performance on par with Tree LSTMs and even better performance than the standard Transformer.

## Chapter 9

# Encoding Dependency Information inside Tree Transformer

This chapter is based on the paper titled “Encoding Dependency Information inside Tree Transformer” co-authored with Robert E. Mercer that is going to appear in The 34th Canadian Conference on Artificial Intelligence (CAI 2021) [10].

Representing a sentence in a high dimensional space is fundamental for most natural language processing (NLP) tasks at present. These representations depend on the underlying structures upon which they are built. Two scenarios are possible: one is to view the sentence as a sequence of words and another is to consider its inherent grammatical structure. It is possible to equip the first way with some external grammatical knowledge, but to capture a proper syntax would be close to impossible. Therefore, we investigate the second one by extending the design of an existing dependency tree transformer (DT-Transformer). We propose adding a novel edge encoding mechanism to this prior architecture. Experiments show that in sentence encoding, having access to information about the relationships between “head” words and their “dependent” words and how the heads are influenced by the dependent words achieves better sentence representation. Evaluation on the four tasks shows noteworthy results compared to the existing DT-Transformer, standard Transformer, LSTM-based models, and tree-structured LSTMs. Extensive experimentation with representing the edge embeddings as different distributions (mean and standard deviation), encoding the edges in different ways, and an ablation study to find where to place each module in the architecture and which modules to use in the design is also provided.

## 9.1 Introduction

Following the breakthrough in natural language processing (NLP) research with word embeddings [156], recent research has had its focus shifted to developing effective sentence embeddings [46, 59]. Having good generalized sentence representations is fundamental for solving a range of NLP tasks because they eventually involve sentences, e.g., question answering, sentiment analysis, semantic similarity, and natural language inference. Encoding a sentence requires word level information either as a sequence or in a structured form. Large scale models can afford to overlook sentence structure as they deal with lots of data and have access to large computing power for training [65, 132, 246], whereas on a smaller scale, having access to structural information gives a great boost even when training with less data.

In traditional linguistics, dependency parses are used to understand the grammatical structure of a sentence as they are morphologically rich and free of word order [47, 63]. For example in the sentence “I prefer the morning flight through Denver”, the word “prefer” is the root word and its dependents are “I” and “flight”. In phrase-based parsing, the connection *prefer*  $\rightarrow$  *flight* is more distant. Similarly, “morning” and “Denver”, modifiers of “flight”, are linked to it directly in the dependency structure, which is not the case in the phrase structure tree. Therefore there is more flexibility in using a dependency tree for getting the semantic representation of a sentence as the grammatical information will come with it as well.

In this chapter, we propose a novel edge encoding mechanism of a dependency parse tree and extend the design of one of the existing dependency tree transformer models from Chapter 8 using very few extra parameters. To the best of our knowledge, no work has been done on encoding these head-dependent relations into a dependency tree edge. We share the same composition function across word level encoding and edge level encoding, allowing the attention module to transfer knowledge across these two levels. Unlike existing tree structured models [212, 214], the number of parameters in our model is not dependent on the number of dependents under a head node. We present visualizations showing how our model makes a classification decision by observing the attention probabilities in the tree. Our extensive evaluation shows that the edge label encoding information certainly helps to improve our model or to be at least on par compared to the existing sequential (i.e., LSTM and Transformer) and tree-structured (i.e., Tree LSTM, RvNN and Tree Transformer) models.

## 9.2 Related Work

Most recent sentence encoders utilize deep learning-based encoding to obtain a dense semantic representation. These encoders treat the sentence either in the raw form just as a sequence of

words or look at its tree representation generated from a dependency or constituency parser.

The attention mechanism adopted by most of the RNN based models requires access to the hidden states at every time step [120, 248]. This introduces a wait time as all hidden states have to be generated prior, making it hard to parallelize. To overcome this, [174] propose a query and key-based fully attentive neural network which depends on matrix multiplication to calculate the attention probability and is easily parallelizable. Later, [240] adopt this and build a state of the art machine translation model called Transformer. The encoder portion of transformer operates using the multi-head version of the attention module. Following this, [3] propose a multi-branch version of transformer where they claim that it is important to allow different heads to operate to their required extent rather than to make all of them contribute uniformly. [46] propose an unsupervised sentence encoder by leveraging only the encoder part of Transformer. They train on the large Stanford Natural Language Inference (SNLI) corpus and then use transfer learning on smaller task-specific corpora.

Large scale language models have proven to be very effective in providing good sentence representations. [65] propose a general-purpose NLP architecture by stacking several transformer encoders where a special token padded at the beginning of the sentence contains the sentence representation. [188] uses multilayer transformer decoders where multi-head self-attention is applied over the input followed by a *softmax* to get the output distribution over the target tokens. In this series, [132] combines the prior two methods and proposes a machine translation model where the former one reconstructs the [MASK] tokens at the encoder and the latter one performs an auto-regressive prediction at the decoder.

[212, 214, 216] propose a family of recursive neural net (RvNN) based models, where a composition function is applied recursively bottom-up on child nodes to compute the parent node representation until the root is reached. [228] propose two tree-structured LSTMs, where the hidden and cell state of a parent node depends only on its children, allowing the connection between a head and its dependent (dependency Tree LSTM) or on the words within a phrase (constituency tree LSTM) to remain intact. [13] further encodes attention inside these tree version LSTMs showing the degree to which different children contribute to generate the parent node representation. [14] propose a tree version of transformer where they replace the composition function with an attention module and test it on four different NLP tasks.

Unlike the models which require precomputed structures, some recent works automatically extract structure from a sentence as part of an end-to-end design. [258] propose an actor-critic based reinforcement learning framework where the critic evaluates an actor-generated structure and provides feedback as a reward or penalty to train the actor. [9] utilize a similar framework for sentence pair modelling tasks and at the same time encodes a cross lookup mechanism inside the actor. Recently, [204] propose a Parsing-Reading-Predict Network (PRPN) which

can induce syntactic structure automatically from an unannotated corpus and can learn a better language model to that induced structure. Later, [94] verify the empirical success of neural network latent tree learning by testing this PRPN under various configurations and datasets. However, [242] find some issues with this latent tree learning, such as inconsistencies during negation handling, being biased towards producing shallow trees, and a tendency to consider the last two words of a sentence as constituents. Unlike all these models which can only induce syntactic trees, [235] generate task-specific latent dependency trees without having any knowledge from a gold standard parser utilizing Kirchhof’s matrix tree theorem [41] to mimic the latent grammatical association between words as  $N \times N$  matrix.

## 9.3 Model

In this section, we describe the model in detail. We are motivated to traverse a dependency tree using the attention module designed for a standard Transformer. Unlike the tree transformer [14] that we are extending, we leverage both word and edge information. First, we describe the design components which include a novel edge encoding mechanism followed by an in-depth exposure to the general architecture of our edge encoded Tree Transformer model. Finally, we conclude this section by suggesting some task-specific modifications.

### 9.3.1 Design Components

We start by first extracting the dependency tree of a given sentence using the Stanford CoreNLP 3.9.1 parser using Universal Dependencies version 1 [148] and traversing it as shown in Figure 9.1(a). A dependency tree has word information ( $\mathbf{R} \rightarrow \{B, C\}; \mathbf{B} \rightarrow \{D, E\}; \mathbf{C} \rightarrow \{F, G\}$ ) on every node and we represent this information with word vectors  $\mathbf{X} \in \mathbb{R}^{1 \times d}$  initialized by some standard word embeddings. In addition to this, each edge is marked with dependency information of which no standard representation is available. We initialize an edge embedding layer with a normal distribution having a mean ( $\mu$ ) and a standard deviation ( $\sigma$ ) to represent the  $N$  unique dependency labels from the Stanford parser [63].

$$\mathbf{a}_i \in \mathcal{N}(\mu, \sigma); \forall i \in [1 \dots N] \quad (9.1)$$

We propose a novel encoding mechanism of the dependency tree edges as shown in Figure 9.1(b). It is to be noted that there exist other encoding mechanisms but this one gives comparatively better results. We hypothesize that representing an edge in both the forward and backward directions conveys more information than representing in just one direction. Each edge has three participants: parent ( $\mathbf{q}$ ), child ( $\mathbf{x}$ ) and edge label ( $\mathbf{y}$ ). Our extensive experiments

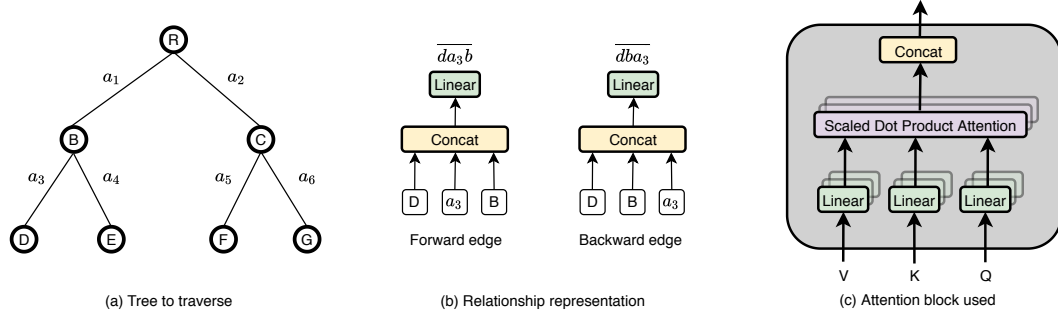


Figure 9.1: Design components.

indicate that  $\mathbf{q}$ ,  $\mathbf{x}$ , and  $\mathbf{y}$  can be in any permutation as long as the orders for the forward ( $\mathbf{f}$ ) and backward ( $\mathbf{b}$ ) directions are unique. We create an edge representation by concatenating the participants and create two large vectors as follows,

$$\mathbf{f}^{1 \times 3d} = [\mathbf{q}; \mathbf{y}; \mathbf{x}] \quad \mathbf{b}^{1 \times 3d} = [\mathbf{q}; \mathbf{x}; \mathbf{y}] \quad (9.2)$$

Next, we project these edge representations into a lower dimension  $d$  just to be consistent with the dimension of the word embedding as follows,

$$\bar{\mathbf{f}}^{1 \times d} = \mathbf{W}_1 \mathbf{f} \quad \bar{\mathbf{b}}^{1 \times d} = \mathbf{W}_2 \mathbf{b} \quad (9.3)$$

We use multi-head self-attention [240] as the composition function of a sub-tree as shown in Figure 9.1(c). It starts by projecting the input into a lower dimension  $n$  times ( $n$  being the number of heads) and applies a self-attention (SA) block on top of them. This SA block is equipped with three projection matrices:  $\mathbf{W}_Q$ ,  $\mathbf{W}_K$  and  $\mathbf{W}_V$ . It considers the input tensor as query, key, and value matrices. First, it transforms the query and key matrices as  $\mathbf{W}_Q \mathbf{Q}$  and  $\mathbf{W}_K \mathbf{K}$ . Next, it uses a dot-product for the alignment followed by *softmax* to get the attention probabilities. Finally, it multiplies these probabilities with the transformed value matrix  $\mathbf{W}_V \mathbf{V}$  to get a scaled representation. All of these computations are summarized in Eqn. 9.4. The  $\mathbf{p}_i$ 's are concatenated to get the final attentive representation.

$$\mathbf{p}_i = \text{softmax} \left( \frac{(\mathbf{W}_Q \mathbf{Q})(\mathbf{W}_K \mathbf{K})^T}{\sqrt{d_k}} \right) (\mathbf{W}_V \mathbf{V}) \quad (9.4)$$



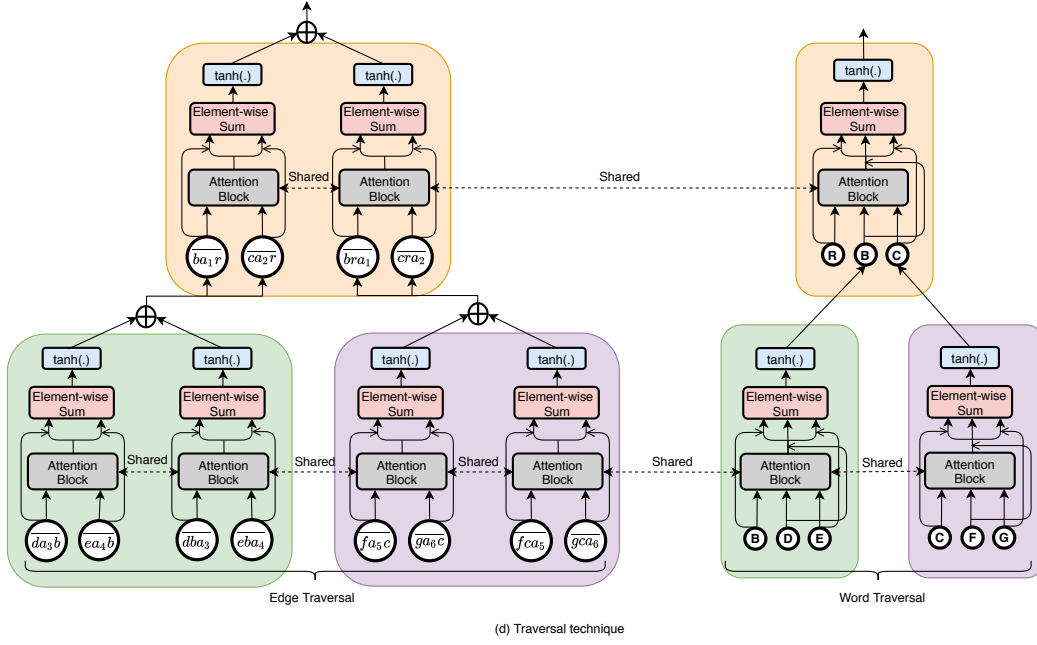


Figure 9.2: Encoding edge information inside dependency tree transformer.

### 9.3.2 General Architecture

As shown in Figure 9.2, our general architecture involves traversing the tree twice: once using the word information and another using the edge label information. However, the composition function (**F**) remains the same for both cases. We now explain what is used as input to this composition function for both cases. Later in this subsection we explain how we can extract the final parent representation utilizing the transformed input.

**Word Traversal:** Dependency trees contain a word at every node. To traverse a subtree in a dependency tree, we look at both the parent and child representations by concatenating them as a tensor **u** as follows,

$$\mathbf{u} = [\mathbf{q}; \mathbf{x}_1; \dots; \mathbf{x}_n] \quad (9.5)$$

Here, **q** is the parent representation and the  $\mathbf{x}_i$ 's are the child representations. Finally, we apply **F** over **u** to get the attentive transformed representation of a subtree as  $\mathbf{p}_u = \mathbf{F}(\mathbf{u})$ .

**Edge Traversal:** Given a subtree, we traverse it in two different ways: one with forward edges and one with backward edges. We create two new tensors  $\mathbf{v}^f$  and  $\mathbf{v}^b$  representing the forward and backward edge versions, respectively,

$$\mathbf{v}^f = [\bar{\mathbf{f}}_1; \bar{\mathbf{f}}_2; \dots; \bar{\mathbf{f}}_n] \quad (9.6)$$

$$\mathbf{v}^b = [\bar{\mathbf{b}}_1; \bar{\mathbf{b}}_2; \dots; \bar{\mathbf{b}}_n] \quad (9.7)$$

Here, the  $\mathbf{f}$ 's and  $\mathbf{b}$ 's are the forward and backward edge representations as defined above. Finally, we apply the composition function to them as follows,

$$\mathbf{p}_{v^f} = \mathbf{F}(\mathbf{v}^f) \quad \mathbf{p}_{v^b} = \mathbf{F}(\mathbf{v}^b) \quad (9.8)$$

**Parent Representation:** As mentioned before, we use a multi-head self-attention block as the composition function ( $\mathbf{F}$ ) [240]. For each of the representations,  $\mathbf{u}$ ,  $\mathbf{v}^f$  and  $\mathbf{v}^b$ , after transforming the inputs using this  $\mathbf{F}$ , we apply a residual connection, transform it non-linearly and perform an element-wise summation (EwS) to get the three hidden representations of the subtree. The computations for word traversal and edge traversal are shown in Eqn. 9.9

$$\begin{aligned} \tilde{\mathbf{p}}_{\mathbf{u}} &= \tanh(\text{EwS}((\mathbf{p}_{\mathbf{u}} + \mathbf{u})\mathbf{W} + b)) \\ \tilde{\mathbf{p}}_{v^f} &= \tanh(\text{EwS}((\mathbf{p}_{v^f} + \mathbf{v}^f)\mathbf{W} + b)) \\ \tilde{\mathbf{p}}_{v^b} &= \tanh(\text{EwS}((\mathbf{p}_{v^b} + \mathbf{v}^b)\mathbf{W} + b)) \\ \tilde{\mathbf{p}}_{\mathbf{v}} &= \tilde{\mathbf{p}}_{v^f} + \tilde{\mathbf{p}}_{v^b} \end{aligned} \quad (9.9)$$

Finally, at the root, we concatenate  $\tilde{\mathbf{p}}_{\mathbf{u}}$  and  $\tilde{\mathbf{p}}_{\mathbf{v}}$  to get the final tree representation ( $R = [\tilde{\mathbf{p}}_{\mathbf{u}}; \tilde{\mathbf{p}}_{\mathbf{v}}]$ ).

### 9.3.3 Dataset Specific Design Details

For most tasks, both the word and edge traversals are done independently and the resultant representations are concatenated as shown above. We tried performing both of these traversals concurrently but it did not give good performance. Some tasks require predicted labels for each node. For these tasks we perform the two traversals in the same way as before but in parallel, and perform classification at each node. Also needed is the computation of an additional representation ( $r$ ) at each node which is used for the classification with the edge label  $\mathbf{y}$  (from Section 9.3.1) as follows,

$$r = \begin{cases} \tanh(\mathbf{W}_I(\tilde{\mathbf{p}}_{\mathbf{u}} + \tilde{\mathbf{p}}_{\mathbf{v}})), & \text{if current node is an intermediate node} \\ \tanh(\mathbf{W}_L(\tilde{\mathbf{p}}_{\mathbf{u}} + \mathbf{y})), & \text{if current node is a leaf node} \end{cases} \quad (9.10)$$

## 9.4 Datasets and Experimental Setup

In this section, we present the dataset details along with the experimental setup. We first describe our task corpora as well as their statistics in terms of the number of samples: train, test, and validation. Following this, we explain the technical details of our proposed architecture along with its hyper-parameter settings.

**Datasets:** We wanted to test our model on data from different domains as well as on different tasks. The tasks are: the Stanford Sentiment Treebank (SST) [214] for sentiment analysis, Sentences Involving Compositional Knowledge (SICK) [150] for semantic relatedness (-R) and natural language inference (-E), and the Microsoft Research Paraphrase (MSRP) corpus [70] for paraphrase identification.

- **MSRP:** Given a pair of sentences, the task is to identify whether or not they are paraphrases of each other [70]. Train: 4076; Test: 1725; Validation: N/A. We randomly sample and exclude 10% of the training data and use that as the validation set.
- **SICK:** The dataset contains sentences derived from video and image annotations. The samples are labelled for both the 3-class SICK-E classification task and the SICK-R regression task which uses real-valued labels between 1 and 5 [150]. We use the same setup as [228] who compute a target distribution  $p$  as a function of the predicted score  $y$  given by Eqn. 9.11. Train: 4500; Test: 4927; Validation: 500.

$$\tilde{p}_i = \begin{cases} y - \lfloor y \rfloor, & \text{if } i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & \text{if } i = \lfloor y \rfloor \\ 0, & \text{otherwise} \end{cases} \quad (9.11)$$

- **SST:** The samples in this dataset are labelled for both the binary and the 5-class classification task. In this work, we are using only the binary classification labels. Train: 6920; Test: 1821; Validation: 872.

We use accuracy as the standard evaluation metric for SICK-E, MSRP and SST and mean squared error (MSE) along with Pearson’s  $r$  as the evaluation metric for SICK-R. We use KL-divergence as the loss function for SICK-R and cross entropy as the loss function for the other three tasks.

**Experimental Setup:** We use GloVe 300-dimensional word vectors [183] to initialize the word embedding layer weights. We use a normal distribution with a cross-validated mean and standard deviation to initialize the edge embedding layer weights. We keep these layers frozen during training. In the multi-head self-attention block, the dimensions of the query, key and value matrices are set to 50 and we use 6 parallel heads on each input. We use a dropout probability of 0.1. During training, the model parameters are updated using the Adagrad algorithm [72] with a fixed learning rate of 0.0002. We train our model on an Nvidia GeForce GTX 1080Ti GPU and use PyTorch 1.3 for the implementation under the Linux environment.

Model Types	Model	SICK-E (Acc.%)	SICK-R (MSE)	SST (Acc.%)	MSRP (Acc.%)
LSTM	LSTM [228]	76.80	.2831	84.90	71.70
	Bi-LSTM [228]	82.11 †	.2736	87.50	72.70
	2-layer LSTM [228]	78.54 †	.2838	86.30	69.35 †
	2-layer Bi-LSTM [228]	79.66 †	.2762	87.20	70.40 †
	InferSent [59]	84.62	.2732	86.00	74.46
Transformer	USE_T [46]	81.15	.5241 †	85.38	74.96 †
	USE_T+DAN [46]	-	-	86.62	-
	USE_T+CNN [46]	-	-	86.69	-
Tree Structured	SDT-RNN [212]	-	.3848	-	-
	RAE [209]	-	-	82.40	76.80
	MV-RNN [211]	58.14 †	-	82.90	66.91 †
	RNTN [216]	59.42 †	-	85.40	66.91 †
	DT-RNN [212]	63.38 †	.3822	86.60	67.51 †
	DT-LSTM [228]	83.11 †	.2532/.2625 †	85.70/85.10 †	72.07 †
	CT-LSTM [228]	82.00 †	.2734/.2891 †	88.00/87.27 †	70.07 †
	CTT [14]	82.72 †	.3012 †	86.66 †	71.73 †
	DTT [14]	82.95 †	.2774 †	83.12 †	70.34 †
Ours	DTT + edge label	83.32	.2627	83.75	71.96

Table 9.1: Performance comparison of the *DT-Transformer* + *edge label* against some state-of-the-art LSTM, transformer and tree-structured models. Models that we implemented are marked with †.

## 9.5 Results and Discussion

In this section, we present the effectiveness of equipping a dependency tree transformer with edge label information by evaluating on four NLP tasks. We show that initializing the edge embedding layer weights with a normal distribution having various means and standard deviations impacts the model’s performance. We also present how different permutations of the dependency tree edge encoding affect the model’s effectiveness. A detailed ablation study reveals the design that led to the best performance. Finally, we conclude this section by visualizing the dependency trees of two test set samples from the SST dataset, one is predicted correctly and the model mistakenly predicts the other.

Table 9.1 compares our model’s performance on four tasks using task-specific evaluation metrics. To do the comparison, we implement both variants of Tree LSTM and Transformer-based architectures and some of the RvNN and LSTM-based models which do not have reported results for all tasks. We only include models that have a comparable number of parameters to ours and do not compare with large language models like BERT [65], BART [132], XLNet [246], and OpenAI-GPT [188]. The evaluation is performed on each corpus separately following the standard train/test/validation split instead of assessing on any transfer perfor-

Mean	STD	SICK-E (Acc.%)	SICK-R (Pearson's r/MSE)	SST (Acc.%)	MSRP (Acc.%)
0	.025	81.65	85.89/.2763	81.22	71.57
0	.050	<b>82.97</b>	86.02/.2648	82.32	71.79
0	.075	82.18	85.90/.2684	82.54	71.07
0	.100	81.98	85.91/.2666	81.93	71.18
0	.125	82.24	85.95/.2656	81.37	71.18
0	.150	81.95	85.72/.2698	81.48	71.79
0	.175	82.30	85.84/.2679	81.55	<b>71.96</b>
0	.200	81.17	<b>86.11/.2627</b>	81.60	71.23
0	.225	81.79	85.81/.2682	<b>83.75</b>	<b>71.96</b>
0	.250	82.44	85.61/.2726	81.16	71.23
0	.275	81.79	85.94/.2663	81.49	71.35
0	.300	81.29	85.93/.2663	81.22	71.07
0	.325	81.81	85.92/.2669	81.93	70.91
0	.350	81.65	85.34/.2769	81.11	71.40
0	.375	81.61	85.85/.2693	82.10	71.35
0	.400	82.24	85.70/.2703	82.08	71.24
-.0014	.381	81.77	85.97/.2667	-	71.01
-.0066	.373	-	-	82.10	-

Table 9.2: Effect of initializing edge embeddings with a fixed mean (0) and changing standard deviation (STD). The last two rows have the respective word embedding mean and STD for SICK-E, SICK-R, MSRP, and SST together with the performance when used as the edge embedding mean and STD.

mance. On SICK-E, our model is achieving 83.32% accuracy, which is on par with DT-LSTM (83.11%) as well as CT-LSTM (82.00%) and much better than the DT-Transformer (82.95%) as well as the standard Transformer (81.15%). On SICK-R, our .2627 MSE is almost the same as the best performed DT-LSTM evaluated in the same settings as ours. On this task, our model is doing significantly better than the standard Transformer (.5241 MSE) and comparatively better than the DT-Transformer having no edge label information. On the SST dataset, our model is getting 83.75% accuracy which is not as good as most of the tree-structured and sequential models. However, it is doing much better than RAE (82.40% Acc.) which is the best performing model on the MSRP dataset. The performance gap between our model and the constituency tree-based models can at least be partially attributable to the fact that these latter ones are trained on more labeled data (150K vs. 319K). On the MSRP dataset, our model (71.96% Acc.) is much better than DT-Transformer (70.34% Acc.), CT-Transformer (71.73% Acc.) as well as CT-LSTM (70.07% Acc.) and almost similar to DT-LSTM (72.07% Acc.). It is to be noted that almost all the sequential models, i.e., Transformer, Infsent, and LSTMs, are doing better compared to the tree-structured models on this task.

Table 9.2 shows the effect of changing the edge embedding layer weights on the final model

Forward Edge	Backward Edge	SICK-E (Acc.%)	SICK-R (Pearson's r/MSE)	SST (Acc.%)	MSRP (Acc.%)
<i>xyq</i>	<i>xqy</i>	82.97	<b>86.11/.2627</b>	83.58	71.29
<i>xyq</i>	<i>qyx</i>	81.20	85.71/.2700	82.32	<b>71.96</b>
<i>yxq</i>	<i>yqx</i>	81.12	85.48/.2741	82.10	70.63
<i>qyx</i>	<i>yqx</i>	81.14	85.37/.2786	<b>83.75</b>	68.57
<i>qyx</i>	×	<b>83.32</b>	85.31/.2781	79.68	69.35
<i>qxy</i>	×	81.65	84.90/.2848	81.65	69.02
<i>xyq</i>	×	81.71	85.69/.2702	82.92	71.68
<i>xqy</i>	×	82.10	84.91/.2839	81.93	71.79
<i>yqx</i>	×	81.33	85.13/.2825	81.11	68.07
<i>yxq</i>	×	81.81	85.12/.2819	82.32	69.29

Table 9.3: Effect of different ways to form forward and backward edges. *q*: parent; *x*: child; *y*: edge label.

performance. We explore this phenomenon by initializing the edge embedding layer weights with a normal distribution having a fixed mean and varying standard deviation. We also experiment with initializing the edge embedding layer weights following the same distribution as the word embedding layer. As can be seen, the best performances across the datasets are achieved with mean 0 and standard deviation in the range of 0.175 – 0.225, except for SICK-E. The best performance is with standard deviation 0.05. Similar to the word embedding layer weights, we keep the edge embedding layer weights frozen by not including them in the training.

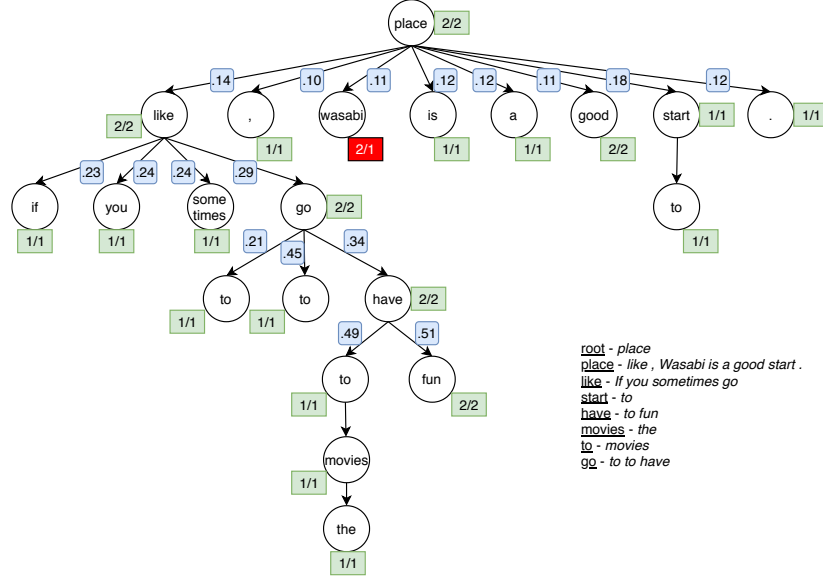
Table 9.3 shows the effect of different ways to encode forward and backward edges on the final model performance. Together, there are  $3! + (3! \times 3!)$  ways to encode forward and backward edges, but here we only report some of the noteworthy results as the others are giving similar or comparatively poorer performance. We do not experiment with only backward edges as it seemed similar to having forward edges only. Of our four tasks, having both forward and backward edges together gives better performance as opposed to having them alone. As can be seen, there is no fixed pattern for encoding the edges. Instead, it is dataset specific and can only be determined through a brute force experiment.

We also perform an ablation study on our model by plugging in and out different modules. We report the results in Table 9.4. We show only the 5 best performing settings on each dataset. For better readability, we provide default settings and then override them by turning on and off different modules and changing their values as well. As can be seen, the default settings are giving the best performance only for the SICK-R and MSRP datasets. For SICK-E, forward edges only (see Table 9.3), multi-branch attention as the composition function, and a linear classifier gives the best performance. For the SST dataset, we use the default settings for the tree traversal and the settings defined in Eqn. 9.10 for the node-wise classification.

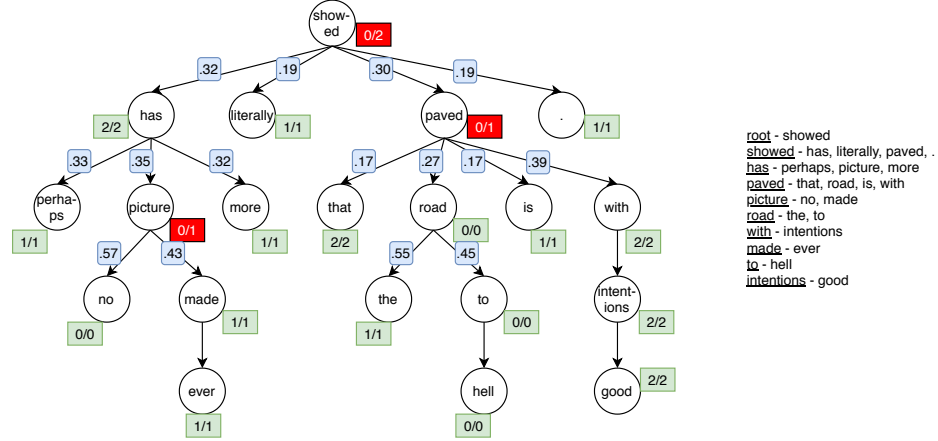
<b>SICK-E</b>	
<b>Model Settings</b>	<b>Accuracy (%)</b>
<i>fc_layer</i> = linear	82.97
<i>fc_layer</i> = linear, <i>final_activation</i> = softmax	80.55
<i>backward_edge</i> = False, <i>word_attention</i> = multi-head, <i>edge_attention</i> = multi-head	81.88
<i>backward_edge</i> = False, <i>word_attention</i> = multi-head, <i>edge_attention</i> = multi-head, <i>fc_layer</i> = linear	82.87
<i>backward_edge</i> = False, <i>word_attention</i> = multi-branch, <i>edge_attention</i> = multi-branch, <i>fc_layer</i> = linear	<b>83.32</b>
<b>SICK-R</b>	
<b>Model Settings</b>	<b>Pearson's r/MSE</b>
<i>Default</i>	<b>86.11/.2627</b>
<i>pooling_type</i> = mean	82.21/.3451
<i>word_attention</i> = multi-branch	82.21/.3451
<i>word_attention</i> = single-head, <i>edge_attention</i> = single-head	81.00/.3507
<i>word_attention</i> = multi-branch, <i>edge_attention</i> = multi-branch	84.83/.2872
<b>SST</b>	
<b>Model Settings</b>	<b>Accuracy (%)</b>
<i>Default</i> + word traversal to classify both leaf and intermediate nodes	80.89
<i>Default</i> + word and edge traversal to classify both leaf and intermediate nodes	81.60
<i>Default</i> + word traversal to classify leaf, word and edge traversal to classify intermediate nodes	81.99
<i>Default</i> + word and edge traversal to classify leaf, word and edge vectors to classify intermediate nodes	<b>83.75</b>
<i>Default</i> + word and edge vectors to classify leaf, word and edge traversal to classify intermediate nodes	82.70
<b>MSRP</b>	
<b>Model Settings</b>	<b>Accuracy (%)</b>
<i>Default</i>	<b>71.96</b>
<i>projection_residual</i> = True	71.07
<i>optimizer</i> = SGD, <i>backward_edge</i> = False	71.18
<i>word_attention</i> = single-head, <i>edge_attention</i> = single-head	70.74
<i>word_attention</i> = multi-branch, <i>edge_attention</i> = multi-branch	70.68

Table 9.4: Ablation study. (Default settings: *word\_embedding\_train* = False, *edge\_embedding\_train* = False, *pooling\_type* = maxpool, *word\_attention* = multi-head, *edge\_attention* = multi-head, *optimizer* = Adagrad, *fc\_layer* = non-linear, *word-edge\_projection* = not-shared, *forward\_edge* = True, *backward\_edge* = True, *projection\_residual* = False, *final\_activation* = log-softmax)

Figure 9.3 shows how the model puts attention on different parts of a dependency tree and computes the final sentiment. The visualization also shows whether the model is making a correct decision or not on each of the nodes. As can be seen in Figure 9.3(a), our model gets the



(a) If you sometimes like to go to the movies to have fun , Wasabi is a good place to start.



(b) Perhaps no picture ever made has more literally showed that the road to hell is paved with good intentions .

Figure 9.3: Attentive tree visualization (DT-Transformer + edge label). **0**-Negative; **1**-Neutral; **2**-Positive. **X/Y**: X-Predicted label; Y-True label. The blue squares indicate attention probabilities.

final sentiment of the sentence correctly and it also correctly classifies the sentiment of each of the nodes except the word “wasabi”. At the first level, the word “like” is getting 14% attention which gets its positive sentiment from the subtree *go* → *to*, *to*, *have* which itself is getting highest attention (29%) among its siblings. Also, in the subtree *have* → *to*, *fun*, the edge “fun” is getting a comparatively higher probability which is a positive sentiment word. In the second example shown in Figure 9.3(b), our model is making correct predictions for almost all nodes but getting some of the key ones wrong. The sentiment of the subtree *paved* → *that*, *road*, *is*, *with* is predicted as negative, which propagates and makes the final decision wrong. However, the



misprediction in subtree *picture*  $\rightarrow$  *no, made* does not have any effect because this entire subtree is getting almost equal attention as its siblings. Overall, having these attention probabilities provides transparency to the model and tells how it is thinking about a specific example and helps us to relate to how we as a human would treat it. We have not visualized the output for SICK and MSRP datasets since the attention weights were equal or almost equal, respectively. The reasons could be the very short sentences in the SICK dataset and the syntactic similarity between the two sentences in the SICK and MSRP datasets.

## 9.6 Conclusion

In this chapter, we propose a novel way to include edge label information inside a dependency tree transformer which encodes natural language grammar trees utilizing the modules designed for the standard Transformer. We show that having edge label information definitely helps and gives a significant boost compared to not having it. We conduct a series of in-depth experiments, which explore the distribution of the edge embeddings along with different ways of forming them. We also provide an ablation study to get the optimal performance with minimal settings. Our study shows that having edge label information complements a dependency tree transformer by putting it on par with Tree LSTMs and making it even better compared to the standard Transformer.

## Chapter 10

# Modelling Sentence Pairs via Reinforcement Learning: An Actor-Critic Approach to Learn the Irrelevant Words

This chapter is based on the paper titled “Modelling Sentence Pairs via Reinforcement Learning: An Actor-Critic Approach to Learn the Irrelevant Words” co-authored with Robert E. Mercer that appeared in the 34th AAAI Conference on Artificial Intelligence (AAAI 2020) [9].

Learning sentence representation is a fundamental task in Natural Language Processing. Most of the existing sentence pair modelling architectures focus only on extracting and using the rich sentence pair features. The drawback of utilizing all of these features makes the learning process much harder. In this chapter, we propose a reinforcement learning (RL) method to learn a sentence pair representation when performing tasks like semantic similarity, paraphrase identification, and question-answer pair modelling. We formulate this learning problem as a sequential decision making task where the decision made in the current state will have a strong impact on the following decisions. We address this decision making with a policy gradient RL method which chooses the irrelevant words to delete by looking at the sub-optimal representation of the sentences being compared. With this policy, extensive experiments show that our model achieves on par performance when learning task-specific representations of sentence pairs without needing any further knowledge like parse trees. We suggest that the simplicity of each task inference provided by our RL model makes it easier to explain.

## 10.1 Introduction

In natural language processing (NLP), most of the tasks require a transformation from an input space to some high dimensional vector space where each dimension captures some aspect of the underlying structure about the data. Learning this kind of representation is a fundamental challenge in NLP and is extensively explored by many recent works [32], [125]. In order to make correct decisions, the well-studied mainstream tasks, such as comparing a pair of sentences in terms of semantics or paraphrasing, depend heavily on the quality of the learned representation of each sentence since the comparison is made in this representation space [58], [45], [163] and [227].

Mainstream sentence pair comparing models look only at the sentences being compared individually while overlooking the information they share between them [58], [45]. This is contrary to what humans do, we usually look at the key words of both sentences while comparing them and ignore most of the irrelevant information. For example, while comparing the sentence “A boy is lying in the snow and is making snow angels” with the sentence “Two people wearing snowsuits are on the ground making snow angels” for the natural language inference task, we can just consider that whether the actors in both of these sentences are “making snow angels” or not and take a decision based on that.

Almost all the sentence pair modelling architectures follow a uniform framework: represent the sentences to compare in some high dimensional space using an encoder and compare these representations using a matching module. The encoding section of these models can be classified into four types: *Bag of words* based models which ignore the ordering of the words, *Recurrent and Convolutional neural network* based models which take into consideration the contexts surrounding each word, *Transformer* based models where the complex attention module does the summarizing and finally, the models that work on *predefined structures* like parse trees.

[98] and [104] propose *bag of words* type models where they ignore the word ordering and instead rely on taking the average of word vectors followed by some linear projection layers. [137] also ignore the structure as well as context but relies on an auto-encoder to generate the representation by adapting domain and sentiment supervision.

[163] use just one copy of an LSTM to encode the entailment task sentences followed by a Manhattan distance based similarity function for the inference. [247], [58], and [141] follow the same framework utilizing LSTM followed by a pooling block to summarize the representation even more. [227] and [215] utilize predefined dependency as well as constituency tree structures and use tree based recurrent networks as the composition function to extract the representation. Furthermore, to get more powerful representations, [270] encode attention in-

side the dependency tree variant of tree LSTM whereas [50] extract local and global inference composition by jointly utilizing both standard LSTM as well as tree LSTM.

Recently, Transformer [240] is getting the spotlight for doing machine translation where the recurrence is mimicked by positional encoding and a series of multi-head attention blocks [175]. [45] utilize the encoder portion of the Transformer to project the input into a representation space and then use a matching block as the inference layer. [65] create a generalized language representation model using a multi-layer bidirectional Transformer encoder which also provides very good sentence representation.

Unlike most of the existing works which looks at the predefined structures, there are some works which uses automatically optimized structures for learning a representation. [252] propose a very complex and overly deep model to compose binary tree structures from the supervised downstream task which sometimes gives tree structures that are very different from the standard English syntactic trees. [54] propose a hierarchical multi-scale recurrent neural network which can capture the latent hierarchical structure in the sequence by encoding the temporal dependencies with different timescales. [235] propose a model to compose the English syntactic tree structures without even looking at the gold standard label utilizing Kirchhof's matrix tree theorem [41].

In this work, we propose a sentence representation building model using reinforcement learning (RL) for doing the sentence pair modelling task. We devise a training strategy incorporating a policy based actor which takes decision based on the previous context, current input and structure of the counterpart sentence. We use a delayed reward to guide the structure discovery and the reward is computed based on the performance of a sub-optimal critic. [230]. We use Monte Carlo sampling for exploring the decision space and the final representation is available when all the sequential decisions are made [88]. We fuse the representation module with a policy and critic network where the policy network performs structure discovery based on the response of a sub-optimal critic and the critic gets further tuned on this structured pair representation to adapt itself more on the response of the actor. Even without any explicit structure annotation, our policy based actor builds pretty good sentence representations by filtering out some irrelevant words allowing the critic to get on par or even better performance on these possibly optimal structures.

## 10.2 Model

In this section, we describe our model in detail. We first explain how the critic is trained in a delayed manner with and without the actor response. Following this, we explain how we train the actor using the response of a trained critic. We conclude this section by giving a high level

view of the workflow of our model.

**Training the Critic:** As the critic, we have chosen to use InferSent [58], an LSTM based model, to compute the representations of a pair of sentences and then compare them for an underlying task. It first traverses each sentence as a sequence of  $T$  words  $\{x_t\}_{t=1,\dots,T}$  from left to right and generates a hidden representation at each time step  $\vec{h}_t$ ,  $\forall t \in [1, \dots, T]$ .

$$\vec{h}_t = \overrightarrow{\text{LSTM}}_t(x_1, \dots, x_T) \quad (10.1)$$

Following this, it employs a max (or mean) pooling block to summarize the hidden states in one dense representation.

$$\vec{h} = \text{maxpool}(\vec{h}_1, \dots, \vec{h}_T) \quad (10.2)$$

The next steps are to infer the similarity between the two representations  $(\vec{h}_a, \vec{h}_b)$  using standard matching methods and to project the resultant vector into the space of classes,  $y$ , through a series of fully connected layers as follows

$$x = (\vec{h}_a, \vec{h}_b, |\vec{h}_a - \vec{h}_b|, \vec{h}_a * \vec{h}_b) \quad (10.3)$$

$$P(y|\mathbf{X}) = \sigma(\mathbf{W}_1 \sigma(\mathbf{W}_2 x + \mathbf{b}_2) + \mathbf{b}_1) \quad (10.4)$$

Finally, it is trained by optimizing a task specific loss function as follows

$$H(p, q) = - \sum_{i=1}^n Q(y_i) \log(P(y_i)) \quad (10.5)$$

However, in order to have a sub-optimal critic, we do not train it in the standard fashion. Instead, we initialize two copies: *final* and *active*. We perform all of the steps above using the *final* version of the critic, compute the gradients with respect to its parameters  $(\theta_f = \{\theta_{f_1}, \dots, \theta_{f_k}\})$  and store them.

$$\frac{\partial H}{\partial \theta_f} = [\frac{\partial H}{\partial \theta_{f_1}}, \dots, \frac{\partial H}{\partial \theta_{f_k}}] \quad (10.6)$$

Generally, in batch-wise training, an average loss is calculated for all of the samples in the batch and the network is updated based on that loss. To mimic this behavior, first the *final* version of critic computes loss for each sample in the batch and stores gradients with respect to its parameters for that loss. Next, all of these gradients are accumulated<sup>1</sup> and we update the

---

<sup>1</sup>We also tried averaging the gradients but the addition works better.

*active* version ( $\theta_a = \{\theta_{a_1}, \dots, \theta_{a_k}\}$ ) of critic as follows.

$$\theta_{a_j} = \theta_{a_j} + \sum_{i \in \text{batch}} \frac{\partial H_i}{\partial \theta_{fj}} \quad (10.7)$$

This kind of updating allows us to get a sub-optimal critic and we can utilize it later when we train an optimal critic in a weighted fashion based on the actor response. This type of training paradigm is different from training with batch size 1, since here, for each sample in a batch, the loss gets calculated with respect to a fixed set of parameters and the corresponding gradients are applied once the traversal over the entire batch is done, whereas in training with batch size 1, for each batch, a loss is calculated with respect to a new set of parameters because of the continual updating. Finally, after going over an entire batch, the parameters of the *final* version of the critic gets updated by its *active* version parameters as  $\theta_f = \theta_a$ .

Apart from this, we adopt another way for training the critic as mentioned before which we use during the fine tuning phase based on the actor response. Instead of doing a straight assignment as above ( $\theta_f = \theta_a$ ), the *final* version gets updated in a weighted fashion as follows

$$\begin{aligned} \theta_f &= \theta_a \times \alpha + \theta_f \times (1 - \alpha), \\ \text{where } \theta_a &\in \{\theta_{a_1}, \dots, \theta_{a_k}\}, \theta_f \in \{\theta_{f_1}, \dots, \theta_{f_k}\} \end{aligned} \quad (10.8)$$

Here, the hyperparameter  $\alpha$  is set to 0.1 for all experiments.

**Training the Actor:** We adopt the policy gradient method [225] to update the actor. The policy network guides the policy learning using a stochastic policy  $\pi_\theta(a_t|\mathbf{s}_t; \boldsymbol{\theta})$  along with a delayed reward. At each time step  $t$ , an action  $a_t$  is sampled from the policy following a probability distribution as follows

$$\pi_\theta(a_t|\mathbf{s}_t; \boldsymbol{\theta}) = \sigma(\mathbf{s}_t \mathbf{W} + \mathbf{b}) \quad (10.9)$$

where  $\pi_\theta(a_t|\mathbf{s}_t; \boldsymbol{\theta})$  denotes the probability of choosing  $a_t$  and  $\{\mathbf{W}, \mathbf{b}\}$  is the set of parameters of the actor policy network. Since, we want our policy network to consider the representation of the counterpart sentence, we include it along with the current input and previous context as state. Formally, the state is defined as

$$\mathbf{s}_t = [\mathbf{c}_{t-1}; \mathbf{h}_{t-1}; \mathbf{x}_t; \tilde{\mathbf{h}}_T] \quad (10.10)$$

Here,  $\mathbf{c}_{t-1}$ ,  $\mathbf{h}_{t-1}$  denotes the cell state and hidden state of critic LSTM at time step  $t-1$ ,  $\mathbf{x}_t$  denotes the input at time step  $t$  and  $\tilde{\mathbf{h}}_T$  is the summary vector of the counterpart sentence generated by the critic LSTM. We use the same policy network to sample actions for both the sentences to

be compared. Using this state and policy, we perform action sampling for the whole sequence to obtain the delayed reward [259] as follows

$$action = \begin{cases} 1, & \text{if } P(\mathcal{Y}) > \epsilon \text{ and } P(\mathcal{Z}) > \pi_\theta \\ 0, & \text{if } P(\mathcal{Y}) > \epsilon \text{ and } P(\mathcal{Z}) < \pi_\theta \\ 1, & \text{if } P(\mathcal{Y}) < \epsilon \text{ and } P(\mathcal{Z}) < \pi_\theta \\ 0, & \text{if } P(\mathcal{Y}) < \epsilon \text{ and } P(\mathcal{Z}) > \pi_\theta \end{cases} \quad (10.11)$$

where  $\mathcal{Y}$  and  $\mathcal{Z}$  are uniform random variables, the hyperparameter  $\epsilon$  is set to 0.05 and  $\pi_\theta$  is the policy from Eqn. 10.9. Our action space is limited to produce binary actions which in a sense can be used as a representation selection model. Once all of the actions are sampled for both the sentences, we get a new representation for both of them which is further used by the critic to compute  $P(y|\mathbf{X})$  and estimate the reward to guide the policy learning. Formally, for a given sentence  $X = x_1, x_2, \dots, x_L$ , there is a corresponding sampled action sequence  $A = a_1, a_2, \dots, a_L$  obtained from the policy where  $a_i = 1$  means to keep the word and  $a_i = 0$  means to discard it as it has no contribution in the final representation when compared with its counterpart sentence. Using this hypothesis, the critic states are updated as follows

$$\mathbf{c}_t, \mathbf{h}_t = \begin{cases} \mathbf{c}_{t-1}, \mathbf{h}_{t-1}, & \text{action} = 0 \\ f(\mathbf{c}_{t-1}, \mathbf{h}_{t-1}, \mathbf{x}_t) & \text{action} = 1 \end{cases} \quad (10.12)$$

where  $f$  denotes all the gate and update functions from the critic, and  $\mathbf{c}_t$ ,  $\mathbf{h}_t$  and  $\mathbf{x}_t$  denote the cell state, hidden state and input at time step  $t$ , respectively. In summary, if a word gets deleted at time step  $t$ , the cell state and hidden state are just copied from time step  $t - 1$ , otherwise it is generated using the standard LSTM gates of critic.

The parameters of our policy net are optimized using the REINFORCE algorithm [243] with an objective being maximizing the expected delayed reward as follows

$$\begin{aligned} J(\theta) &= \mathbb{E}_{(\mathbf{s}_t, a_t) \sim P_\theta(\mathbf{s}_t, a_t)} r(\mathbf{s}_1 a_1 \dots \mathbf{s}_T a_T) \\ &= \sum_{\mathbf{s}_1 a_1 \dots \mathbf{s}_T a_T} P_\theta(\mathbf{s}_1 a_1 \dots \mathbf{s}_T a_T) R_T \\ &= \sum_{\mathbf{s}_1 a_1 \dots \mathbf{s}_T a_T} P_\theta(\mathbf{s}_1) \prod_t \pi_\theta(a_t | \mathbf{s}_t) P_\theta(\mathbf{s}_{t+1} | \mathbf{s}_t, a_t) R_T \end{aligned} \quad (10.13)$$

The delayed reward  $R_T$  is computed using the logarithm of the output probability distribution of the critic  $\log P(y|\mathbf{X})$  over just one sample. Since our state at time step  $t + 1$  is fully determined by the state and action at time step  $t$ , we can make  $P_\theta(\mathbf{s}_1) = P_\theta(\mathbf{s}_{t+1} | \mathbf{s}_t, a_t) = 1$  in Eqn. 10.13

Also as we perform the action sampling for both sentences to be compared, we change the objective function as follows

$$J(\theta) = \sum_{s_1 a_1 \dots s_T a_T} \prod_{t \in l} \pi_\theta(a_t | s_t) R_T + \sum_{s_1 a_1 \dots s_T a_T} \prod_{t \in r} \pi_\theta(a_t | s_t) R_T \quad (10.14)$$

where  $l$  denotes the left sentence and  $r$  denotes the right sentence. By applying the likelihood ratio trick, we update the policy network with the following gradient

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{t \in l} R_T \frac{\partial}{\partial \theta} \log \pi_\theta(a_t | s_t, \theta_t) + \sum_{t \in r} R_T \frac{\partial}{\partial \theta} \log \pi_\theta(a_t | s_t, \theta_t) \quad (10.15)$$

The policy network controls the number of deleted words by using a unimodal function  $f(x) = x + \beta/x$  with the reward and maximize them jointly. The reward  $R_T$  is defined as

$$R_T = \log P(y|\mathbf{X}) - \frac{\gamma Z}{2} \left( \text{abs}\left(\frac{L'_l}{L_l} + \frac{\beta L_l}{L'_l}\right) + \text{abs}\left(\frac{L'_r}{L_r} + \frac{\beta L_r}{L'_r}\right) \right) \quad (10.16)$$

where  $L'_l$  and  $L'_r$  are the number of words with corresponding action value 1, and  $L_l$  and  $L_r$  are the actual lengths for the left and right sentences respectively.  $\gamma$  is a hyper-parameter,  $Z$  is the number of classes and  $\beta$  is the proportion of words we want deleted.

Like the critic, we initialize two versions of the actor: *final* and *active*. For each sample in a batch, the *active* version of the actor policy network is updated by the gradient of the *final* version of the actor policy network parameters

$$\theta_{a_j} = \theta_{a_j} + \sum_{i \in \text{batch}} \frac{\partial J(\theta)_i}{\partial \theta_{f_j}} \quad (10.17)$$

Once a batch is finished, the *final* version of the actor policy is updated in a similar way as the critic using Eqn. 10.8.

**Workflow:** We now give a high level view of the work flow of our entire model. An algorithmic presentation detailing these steps is provided in Algorithms 10.1 and 10.2. We start by training a sub-optimal critic in a delayed manner by first initializing two versions of it (*final* and *active*). For each sample in a batch, we update the *active* version using the gradients of the *final* version. While doing this, we keep the parameters of the *final* version fixed. Once the entire batch is looked at, the *active* version is used to update the *final* version through a straightforward assignment.

After training this sub-optimal critic, we use its response to train an actor with a policy gradient method. To accomplish this, we again start by having two versions of actor (*final* and *active*). With each sample in the batch, Eqn. 10.10 gets an  $s_t$  for each word in the two sentences



**Algorithm 10.1** Overall Training

---

Initialize two copies of Critic network  $C_f$  and  $C_a$   
 Denote the LSTMs of Critic network as  $L_f$  and  $L_a$   
 Set of Critic Parameters  $\{\theta_{f_1} \dots \theta_{f_k}\} \cup \{\theta_{a_1} \dots \theta_{a_k}\}$   
 Discount factor  $\alpha \in [0, 1]$  and Learning rate  $\lambda \in [0, 1]$   
**for** batch  $\in$  Batches **do**  
   **for** sample  $\in$  batch **do**  
      $x, y \leftarrow \text{sample}$   
      $\nabla \theta_f \leftarrow \nabla P_{\theta_f}(\text{out}^* | x, y, \theta_f), \quad \forall \theta_f \in \{\theta_{f_1} \dots \theta_{f_k}\}$   
      $\theta_{a_j} = \theta_{a_j} + \lambda \times \nabla \theta_{f_j}, \quad j \in \{1 \dots k\}$   
   **end for**  
    $\theta_{f_j} = \theta_{a_j}, \quad j \in \{1 \dots k\}$   
**end for**  
 Train Actor on Critic ( $C_f$ ) response using Algorithm 10.2  
 Initialize  $\{\theta_{a_1} \dots \theta_{a_k}\}$  by  $\{\theta_{f_1} \dots \theta_{f_k}\}$   
**for** batch  $\in$  Batches **do**  
   **for** sample  $\in$  batch **do**  
      $x, y \leftarrow \text{sample}, \tilde{x} \leftarrow \emptyset, \tilde{y} \leftarrow \emptyset$   
      $\tilde{h}_{T_x} \leftarrow L_f(y), \tilde{h}_{T_y} \leftarrow L_f(x)$   
     Calculate  $s_{t_x}$  using word vector  $x_t$ , sentence vector  $\tilde{h}_{T_x}$ , cell and hidden states of  $L$   
     at previous time step with Eqn. 10  
     Calculate  $s_{t_y}$  using word vector  $y_t$ , sentence vector  $\tilde{h}_{T_y}$ , cell and hidden states of  $L$   
     at previous time step with Eqn. 10  
     Calculate  $\pi_{\theta_x}$  using  $s_{t_x}$ ,  $\pi_{\theta_y}$  using  $s_{t_y}$  with Eqn. 9  
     Sample actions  $a_x$  using policy  $\pi_{\theta_x}$  and  $a_y$  using policy  $\pi_{\theta_y}$  with Eqn. 11  
     **for** action in  $a_x$  **do**  
       **if** action = 1 at index  $t$  **then**  
         keep the word at index  $t$  and add it to  $\tilde{x}$   
       **end if**  
     **end for**  
     **for** action in  $a_y$  **do**  
       **if** action = 1 at index  $t$  **then**  
         keep the word at index  $t$  and add it to  $\tilde{y}$   
       **end if**  
     **end for**  
      $\nabla \theta_f \leftarrow \nabla P_{\theta_f}(\text{out}^* | \tilde{x}, \tilde{y}, \theta_f), \quad \forall \theta_f \in \{\theta_{f_1} \dots \theta_{f_k}\}$   
      $\theta_{a_j} = \theta_{a_j} + \lambda \times \nabla \theta_{f_j}, \quad j \in \{1 \dots k\}$   
   **end for**  
    $\theta_{f_j} = \theta_{f_j} \times (1 - \alpha) + \theta_{a_j} \times \alpha, \quad j \in \{1 \dots k\}$   
   Initialize  $\{\theta_{a_1} \dots \theta_{a_k}\}$  by  $\{\theta_{f_1} \dots \theta_{f_k}\}$   
**end for**

---

**Algorithm 10.2** Actor Training

---

Start with a pre-trained Critic  $C$  having an LSTM cell  $L$   
 Number of random sampling  $N \in [1, n]$   
 Initialize two copies of Actor policy networks  $\pi_{\theta_f}, \pi_{\theta_a}$   
 Set of Actor Parameters  $\{\theta_{f_1} \dots \theta_{f_k}\} \cup \{\theta_{a_1} \dots \theta_{a_k}\}$   
 Discount factor  $\alpha \in [0, 1]$ , Learning rate  $\lambda \in [0, 1]$   
 Function to calculate the length  $F(\cdot)$   
 No. of classes  $Z \in [1, n]$   
 Proportion of the words to be deleted  $\beta \in [1, n]$   
**for** batch  $\in$  Batches **do**  
   **for** sample  $\in$  batch **do**  
      $x, y \leftarrow \text{sample}$   
     Left Summary  $x' \leftarrow L(x)$ , Right Summary  $y' \leftarrow L(y)$   
     States  $s_X \leftarrow \emptyset$ ,  $s_Y \leftarrow \emptyset$   
     Actions  $a_X \leftarrow \emptyset$ ,  $a_Y \leftarrow \emptyset$   
     Loss  $l \leftarrow \emptyset$ ,  $j \leftarrow 0$   
     **while**  $j < N$  **do**  
        $\tilde{x} \leftarrow \emptyset$ ,  $\tilde{y} \leftarrow \emptyset$   
       Calculate  $s_{t_x}$  using word vector  $x_t$ , sentence vector  $\tilde{h}_{T_x}$ , cell and hidden states of  $L$   
       at previous time step with Eqn. 10  
       Calculate  $s_{t_y}$  using word vector  $y_t$ , sentence vector  $\tilde{h}_{T_y}$ , cell and hidden states of  $L$   
       at previous time step with Eqn. 10  
       Calculate  $\pi_{\theta_x}$  using  $s_{t_x}$ ,  $\pi_{\theta_y}$  using  $s_{t_y}$  with Eqn. 9  
       Sample actions  $a_x$  using policy  $\pi_{\theta_x}$  and  $a_y$  using policy  $\pi_{\theta_y}$  with Eqn. 11  
        $a_{X_j} \leftarrow a_{X_j} + a_{t_x}$ ,  $a_{Y_j} \leftarrow a_{Y_j} + a_{t_y}$   
        $s_{X_j} \leftarrow s_{X_j} + s_{t_x}$ ,  $s_{Y_j} \leftarrow s_{Y_j} + s_{t_y}$   
       **for** action in  $a_x$  **do**  
         **if** action = 1 at index  $t$  **then**  
           keep the word at index  $t$  and add it to  $\tilde{x}$   
         **end if**  
       **end for**  
       **for** action in  $a_y$  **do**  
         **if** action = 1 at index  $t$  **then**  
           keep the word at index  $t$  and add it to  $\tilde{y}$   
         **end if**  
       **end for**  
        $l \leftarrow l + C(\tilde{x}, \tilde{y}) + \frac{\gamma Z}{2} \left( \left| \frac{F(\tilde{x})}{F(x)} + \frac{\beta * F(x)}{F(\tilde{x})} \right| + \left| \frac{F(\tilde{y})}{F(y)} + \frac{\beta * F(y)}{F(\tilde{y})} \right| \right)$   
        $j \leftarrow j + 1$   
     **end while**  
      $l_a \leftarrow (\sum_{i=1}^N l_i) / N$   
     Initialize the gradients  $\nabla \theta_f \in \{\nabla \theta_{f_1} \dots \nabla \theta_{f_k}\}$  by 0  
      $j \leftarrow 0$

---

**Algorithm 10.2** Actor Training (continued)

---

```

while  $j < N$  do
   $i \leftarrow 0$ 
  while  $i < F(a_X)$  do
     $R_L \leftarrow (l_{j,i} - l_a) \times \alpha$ 
     $\nabla \theta_f \leftarrow \nabla \theta_f + R_L \nabla \log \pi_{\theta_f}(a_{X_{ji}}^* | s_{X_{ji}}, \theta_f), \quad \forall \theta_f \in \{\theta_{f_1} \dots \theta_{f_k}\}$ 
     $i \leftarrow i + 1$ 
  end while
   $i \leftarrow 0$ 
  while  $i < F(a_Y)$  do
     $R_L \leftarrow (l_{j,i} - l_a) \times \alpha$ 
     $\nabla \theta_f \leftarrow \nabla \theta_f + R_L \nabla \log \pi_{\theta_f}(a_{Y_{ji}}^* | s_{Y_{ji}}, \theta_f), \quad \forall \theta_f \in \{\theta_{f_1} \dots \theta_{f_k}\}$ 
     $i \leftarrow i + 1$ 
  end while
   $j \leftarrow j + 1$ 
end while
 $\theta_{a_j} = \theta_{a_j} + \lambda \times \nabla \theta_{f_j}, \quad j \in \{1 \dots k\}$ 
end for
 $\theta_{f_j} = \theta_{f_j} \times (1 - \alpha) + \theta_{a_j} \times \alpha, \quad j \in \{1 \dots k\}$ 
Initialize  $\{\theta_{a_1} \dots \theta_{a_k}\}$  by  $\{\theta_{f_1} \dots \theta_{f_k}\}$ 
end for

```

---

in the sample. Next, we use these two sets of  $s_t$ 's in Eqn. 10.9 to calculate the policies ( $\pi_\theta$ 's) for each word in the two sentences. We then use each policy with the sampling strategy as defined in Eqn. 10.11 to get the corresponding action ( $a_t$ ) associated with each word in the two sentences. Each action represents whether to delete or keep a word. Next, we modify both of the sentences according to these sampled actions. These modified sentences are used with the trained critic to calculate a reward. This reward is modified with a term that reflects the ratio of the number of words that we delete from both of the sentences (the subtrahend in Eqn. 10.16). We store all of the  $s_t$ 's,  $a_t$ 's as well as the associated reward and repeat this strategy  $N$  times. We then calculate an average reward. The objective function in Eqn. 10.14 and the gradients in Eqn. 10.15 are calculated  $N$  times using the  $N$  sets of the logarithm of policies and the  $N$  differences between the  $N$  rewards and the average reward. We again adopt the delaying strategy in updating the actor parameters.

For each sample in the batch, the just calculated gradients of the parameters of the *final* version of the actor are used to update the *active* version. We keep the *final* version parameters constant until all the samples in the batch are looked at. Eqn. 10.8 is used to update the parameters of the *final* version of actor by its *active* version in a weighted fashion. The *active* version is then set to this new *final* version. We continue this updating of actor until all the batches are looked at.

After the actor is trained, the previously trained sub-optimal critic is further tuned using

the response of this trained actor. Rather than looking at the real sentence pairs, now the critic looks at the actor-modified sentences. A loss is calculated based on the critic response and the *active* version is updated through the gradients of the parameters of the *final* version. Eqn. 10.8 updates the *final* version of the critic once an entire batch is looked at.

**Ensemble Method:** In order to verify the contribution and structure selection ability of actor, we also perform an ensemble decision check by combining the responses of two critics (i.e., InferSent), one trained just on the raw words without any actor and one trained on the response of an actor. We hypothesize that the raw InferSent model should get more importance than the one which needs an actor. Both model weights are initialized to 0.5 weight. The weight of the one without any actor goes from 0.5 to 1.0 and the one with an actor goes from 0.5 to 0. The final decision  $F_d$  is taken by doing a weighted average on the response of both of the participating entities:  $C_d$  (critic decision) and  $AC_d$  (actor-critic decision).

$$F_d = C_d \times w + AC_d \times (1 - w) \quad (10.18)$$

Here,  $w$  and  $(1 - w)$  are the weights on the critic and actor-critic decisions and they are selected through a grid search over the validation set.

### 10.3 Datasets, Experimental Setup, Results and Analysis

In this section, we explain the experimental setup along with the results obtained and a thorough analysis. We first describe our training corpora as well as all of the benchmarks used in other standard sentence pair modelling studies. Following this, we explain the technical details of our proposed architecture along with its hyper-parameter settings. We also present the detailed results obtained with our RL model and compare with some of the top performing models on their selected datasets. Additionally, we give a qualitative analysis by showing the predictions of our models on some random test samples for all of the tasks. Finally, we conclude this section by giving some insight into the performance of our model by analyzing the generated structures.

**Datasets:** Model evaluation uses three datasets: paraphrase identification, natural language inference, and question-answer pair modelling.

- **MSRP:** Given a pair of sentences, the task is to identify whether or not they are paraphrases of each other [69].
- **SICK:** The dataset contains sentences derived from video and image annotations and the task is to classify a given sentence pair into three classes: Entailment, Neutral and

Contradiction [151].

- **AI2-8grade:** The task is to do a true-false question selection where each data sample consists of a pair of sentences with one being the question and the other being the evidence formed by replacing the *wh* in the question by the answer [30].

**Experimental Setup:** The LSTM hidden state dimension is set to 1024. Word vectors are initialized with the 300 dimension GloVe embeddings [183] and are not updated during training. To smooth the update during critic training, the gradients are divided by  $B^2$  where  $B$  is the batch size which is 5. The learning rate is reduced by a factor of 2 if  $\sqrt{\sum_{i=1}^k \|\nabla \theta_i^2\|}$  is more than a threshold, which is 5 for our experiments. To smooth the policy gradient update, we add a  $\gamma$ ,  $\beta$ , and  $Z$  scaled regularized reward as shown in Eqn. 10.16. The values of  $\gamma$  and  $\beta$  are 0.1 and 0.15, respectively, and  $Z$  is the number of classes.

During training, the critic parameters are updated using stochastic gradient descent [38] with an initial learning rate of 0.1, whereas for training actor, we use Adam [109] to update the parameters with a fixed learning rate of 0.01.

**Results and Analysis:** Table 10.1 compares the performance of our two models on the three tasks to some top performing generalized sentence encoders and some designed for a specific task using accuracy as the evaluation metric. To do a fair comparison, we used the official implementation of InferSent, LSTM, BiGRU Last encoder, Inner attention, ConvNet encoder with the same settings as ours<sup>2</sup>. On the MSRP task, our ensemble model achieves 76.12% accuracy which is better than all of the sequential and tree based sentence encoders. It is noteworthy that these tree based models have access to parse trees which are expensive to compute. We get better performance without this information. On the AI2-8grade dataset, performance (73.84% for InferSent + RL and 74.91% for InferSent + RL (Ensemble)) is below the existing models; however, it is still on par. Later, we show that our model (InferSent + RL) is removing around 90% of the content from this dataset yet is still able to achieve this comparable performance. Finally, state of the art performance (86.12% accuracy) is achieved on the natural language inference task on the SICK dataset. Even though our critic model is much simpler having just a unidirectional LSTM, it is doing much better than [39] with 80.80% accuracy who use transfer learning and [135] with 72.01% accuracy who use an attention block on top of a bidirectional LSTM. Lastly, our version of InferSent with an actor is better than the standard InferSent suggesting that our actor has been able to correctly remove the irrelevant words.

Table 10.2 gives the ensemble method's final  $w$  values and final results. Adding RL alone improves InferSent slightly on two of the three datasets. Carefully re-introducing the effect

<sup>2</sup>Available at <https://github.com/facebookresearch/InferSent>

Model	MSRP Acc.	AI2-8grade Acc.	SICK Acc.
InferSent [58] †	74.46	74.71	84.07
LSTM [58] †	70.74	74.93	76.80
BiGRU Last Encoder [58] †	70.46	74.61	81.47
Inner Attention [135] †	69.74	74.73	72.01
ConvNet Encoder [263] †	73.96	<b>75.26</b>	83.82
InferSent + RL	74.74	73.84	84.57
InferSent + RL (Ensemble)	<b>76.12</b>	74.91	<b>86.12</b>
Seq-LSTMs [270]	71.70	63.30	-
Seq-GRUs [270]	71.80	62.40	-
Tree LSTM [270]	73.50	69.10	-
Tree LSTM + Attn. [270]	75.80	72.50	-
Tree GRU [270]	73.96	70.60	-
Tree GRU + Attn. [270]	74.80	72.10	-
RNN [30]	-	36.10	-
CNN [30]	-	38.40	-
RNN-CNN [30]	-	37.60	-
Attn1511 [30]	-	35.80	-
Ubu.RNN [30]	-	44.10	-
Illinois-LH [122]	-	-	84.60
UNAL NLP [101]	-	-	83.10
SNLI-Transfer 3-class LSTM [39]	-	-	80.80
MaLSTM features + LSTM [163]	-	-	84.20
ECNU [264]	-	-	83.60

Table 10.1: Performance comparison of our model on different tasks against some existing top performing models. We mark models that we implemented as †.

Dataset	w	1 - w	Acc.
MSRP	0.53	0.47	76.12
AI2-8grade	0.76	0.24	74.91
SICK	0.65	0.35	86.12

Table 10.2: The  $w$  values that give the best results when combining the critic ( $w$ ) with the trained actor-critic ( $1 - w$ ).

of an actor-free InferSent with an ensemble technique further improves the InferSent + RL to being state of the art on the MSRP and SICK datasets. On the AI2-8grade dataset, only one model is doing better than our ensemble model.

Table 10.3 shows the performance of our InferSent + RL model on some examples from the test sets of the three corpora. For the SICK dataset, our RL model removes prepositions, articles, and adjectives like colours, which seem not to have any impact on the semantics. It also removes the common phrases like “a piece of”, “There is” and “of the”. On the MSRP

Dataset	Sentence 1	Sentence 2	GT	Pr
SICK	A brown dog is attacking another animal in front of the man in pants	There is no dog wrestling and hugging	N	N
	A cat is crawling under a piece of furniture	An animal is crawling under a piece of furniture	E	E
	A blonde boy in green is sitting on a swing	A blonde boy in green is standing on a swing	C	E
MSRP	They did not read footnotes in a document said the official referring to the section that contained the State Departments dissent	They did not read footnotes in a document he said referring to the annex	1	1
	The company didn't detail the costs of the replacement and repairs	But company officials expect the costs of the replacement work to run into the millions of dollars	0	0
	We are piloting it there to see whether we roll it out to other products	Macromedia is piloting this product activation system in Contribute to test whether to roll it out to other products	1	0
AI2-8grade	cell wall structure is found in a plant cell but not in an animal cell	The cell wall provides structural support and protection	1	1
		Pores in the cell wall allow water and nutrients to move into and out of the cell	1	1
		The cell wall also prevents the plant cell from bursting when water enters the cell	1	1
	positive effect of recycling aluminum cans to manufacture new beverage containers is warming	also maintains the ozone layer that helps protect life from damaging UV	0	0
	The layered mixture of gases surrounding Earth is called the atmosphere	Without it Earth would be a harsh barren world	1	0

Table 10.3: Example predictions from the test set. **GT**: ground truth, **Pr**: predicted.

dataset, our model again removes articles and prepositions which seem not to be of concern when checking for paraphrasing. We notice that our model tries to keep the same subset of words from both sentences in most of the cases and removes the words interspersed among those shared words. For example, in the first example in this group, our model deletes the phrase “to the section that” from the left sentence and some smaller phrases like “in a” and “to the” from both of the sentences. This reduction makes the sentences quite similar and eventually the predicted decision is 1. On the other hand, our model makes a wrong prediction

Dataset	Left		Right	
	Before	After	Before	After
MSRP	18.55	12.11	18.51	12.10
AI2-8grade	21.52	2.37	15.34	2.02
SICK	9.68	5.02	9.52	4.93

Table 10.4: The original average length and the average length after filtering through RL.

on the third example. It tries to do the same thing by removing phrases, but as it does not have any world knowledge about “Macromedia Contribute” being a product, it fails to map it to the word “product” in the left sentence. Finally, on the AI2-8grade dataset, our model removes most of the words from the answer sentence and ends up keeping just the key words to match with the corresponding question. In the first example, our model keeps the word “wall” in all of the answer sentences making the mapping easier with the question. In the second example, our model keeps the key words like “protect”, “damaging” and “UV” in the answer sentence making it clear that the question and the answer topics are completely different and as a result the prediction is 0. Finally, in the third example, our model incorrectly removes the word “Earth” from the answer sentence which makes it difficult for the critic to correctly map it to the question.

Table 10.4 exhibits how much information is removed and how much is kept for each task by our InferSent + RL model. For the paraphrase identification task, about 33% of the original content is removed giving better results than the sequence based models. For the question-answer selection task on the AI2-8grade dataset, our model removes around 90% of the original content and still achieves on par performance. In this dataset, the question-answer pairs are selected from 2nd to 8th grade books. To increase their readability [131], some easy to read words are added around the key words of those sentences. Our model suggests that the easy to read words are not important for the inference. Finally, to do natural language inference (NLI) on the SICK dataset, our model removes about 45% of the content and gets better performance than all of the existing models. These results indicate that the three tasks can be done with more condensed and purified information without losing too much generalizability.

We also analyze the type of words deleted from each corpus with the InferSent + RL model and report the results in Table 10.5. We use the harmonic mean of the ratio of the number of times a word is deleted to its frequency in the corpus (as a percentage) and the number of occurrences of that word to sort the list. It is clear that in two corpora, non-content words (i.e., prepositions, articles) are deleted most of the time. However, in the SICK dataset, there are words like “black”, “white”, “blue” and “red” which are deleted most often because for doing the NLI task, these words contribute very little and can be ignored. For the other two corpora, our model also deletes words like “this”, “have”, “likely”, “has”, “than” and “they” quite often



MSRP				AI2-8grade				SICK			
Word	HM	Word	HM	Word	HM	Word	HM	Word	HM	Word	HM
a	184.38	at	105.56	a	197.30	this	105.56	a	197.14	another	108.09
<num>	184.03	would	102.58	the	196.92	two	102.58	is	194.75	two	107.42
and	181.79	not	97.923	of	196.12	likely	97.923	the	194.07	for	100.49
the	179.23	this	95.864	to	195.07	used	95.864	of	184.80	are	93.080
in	162.63	one	92.516	that	194.90	cells	92.516	white	167.67	has	86.587
for	160.63	out	86.473	in	193.43	energy	86.473	black	167.37	near	84.682
to	160.30	about	85.469	are	192.99	will	85.469	in	159.06	next	84.642
is	156.13	which	84.511	<num>	192.76	example	84.511	and	153.53	that	83.274
with	151.28	into	78.881	by	192.48	best	78.881	blue	147.53	other	81.547
on	147.49	over	73.781	most	191.74	and	73.781	red	146.91	purple	77.008
it	138.55	had	73.705	for	191.57	different	73.705	brown	146.32	top	69.342
of	134.14	her	72.263	is	190.94	than	72.263	an	144.27	orange	69.281
was	133.29	who	70.566	an	190.83	they	70.566	on	137.38	some	66.847
be	126.91	could	69.592	can	190.53	other	69.592	at	135.94	oil	66.524
an	126.19	she	68.012	on	190.42	has	68.012	green	131.74	each	63.945
as	124.75	down	67.791	water	189.17	cell	67.791	with	129.84	big	60.107
he	120.40	all	67.752	as	188.95	student	67.752	yellow	128.57	colored	59.712
will	118.72	than	66.384	be	188.66	not	66.384	group	126.00	to	56.275
or	114.89	when	65.907	it	187.76	organism	65.907	small	125.67	colorful	52.941
up	113.90	some	65.753	with	187.60	do	65.753	large	122.48	field	51.923
that	113.77	I	65.262	Earth	186.39	sound	65.262	its	120.63	three	51.851
its	110.12	first	63.965	The	186.03	This	63.965	one	118.38	golden	49.624
his	108.11	they	63.897	have	185.78	would	63.897	pink	116.86	from	48.457
but	107.83	have	63.358	one	184.53	more	63.358	little	111.07	family	48.422
also	106.21	can	59.299	force	184.49	body	59.299	which	109.16	grey	47.614

Table 10.5: Top 50 deleted words from the test set of all three corpora. **HM**: Harmonic Mean of percentage of a word gets deleted and number of times that word appears.

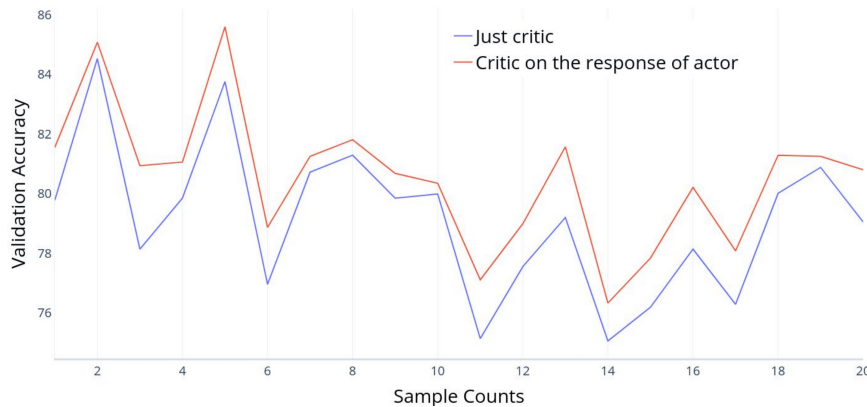


Figure 10.1: Effect of RL sample counts on validation accuracy.

but the rate of deleting them is not as high as the ones reported in Table 10.5

Table 10.5 might suggest that stop words be removed in a preprocessing step. For tasks like semantic relatedness our research suggests not. Stop words with semantic polarity, like “no” and “not”, are very important as they control the overall sentiment of the sentence [7]. We have trained InferSent on the SICK dataset with stop words removed. The evaluation indicates a poorer performance (79.95%) than InferSent with stop words (84.07%). Similar results are obtained for the other two datasets.

In RL, the search space is huge making it hard to find the best action with the maximum

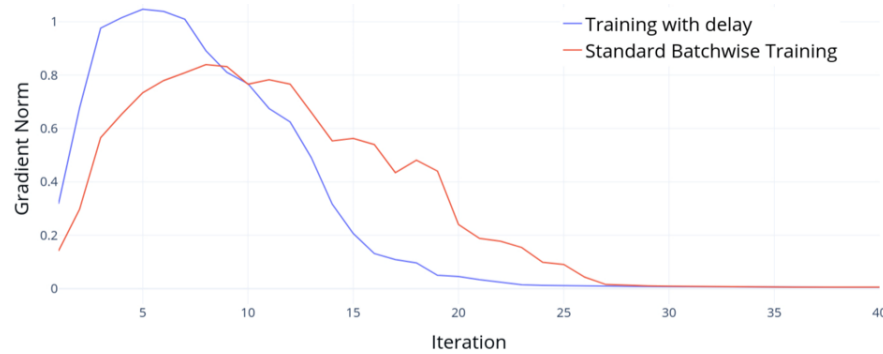


Figure 10.2: Change in gradient norm with training iterations while training with and without delay.

reward. To explore a larger region of this space, we choose to experiment with changing the number of sample counts and storing the average reward based on each possible action. Figure 10.1 depicts the effect of different sample counts on validation accuracy. We choose the range of sample counts to be from 1 to 20, train the actor based on the number of sample counts and record the validation accuracies. We further train as many critics as the number of actors and also record the validation performances. Finally, sample count 5 was selected, as our model was obtaining the best validation performance with this number.

In this study, we adopt a delayed update mechanism in training both actor and critic. This framework allows us to fine tune critic partially without relying on all that actor suggests. To check the effectiveness of this type of training compared to the standard batch gradient descent, we have plotted the gradient norm changes at each epoch in Figure 10.2. As evidenced, change is initially quite drastic but eventually becomes much smoother with the number of iterations and convergence is much quicker compared to standard training.

## 10.4 Conclusion

We have proposed a reinforcement learning method to train a sentence pair modelling architecture using an actor-critic framework. With the help of an actor, our critic model learns to compare two sentences by looking more at content words rather than all words in the sentence, similar to how humans do it. To take an action, our actor model looks not only at the content of just one sentence, it also considers the mutual information shared between the two sentences. Results show that our model gets on par or better performance compared to the existing models by overlooking the irrelevant content.

## Chapter 11

# Learning to Compare Sentence Pairs at the Phrase Level: An Actor-Critic Approach

This chapter is based on the paper titled “Learning to Compare Sentence Pairs at Phrase Level: An Actor-Critic Approach” co-authored with Robert E. Mercer. We are preparing the paper for a journal submission.

Learning sentence representation is a fundamental task in Natural Language Processing. One application that uses sentence representations is sentence pair modelling. Most of the existing sentence pair modelling architectures focus only on word relationships and extract rich features conditioning on that from the sentences to be compared. Apart from just looking at the plain word level information, sometimes it is easy to inspect a phrase chunk and make decisions based on that. In this study, we propose a reinforcement learning (RL) method to automatically generate phrase level structure from a sentence and then compare a pair of sentences based on that. We formulate this automatic phrase generation task using the actor-critic paradigm, where the job of the actor is to come up with an optimal phrasal structure and the critic decides the quality of that structure through a discrete regularized reward. We address this decision making with a policy gradient RL which decides whether to consider a word either as “inside a phrase” or as the “end of a phrase” by looking at the sub-optimal representations of the sentences being compared. We evaluate our learning framework by experimenting on three tasks: semantic similarity, paraphrase identification, and question-answer pair modelling. Our extensive experiments show that our model achieves on par or better performance when learning task-specific representations of sentence pairs.

## 11.1 Introduction

In natural language processing (NLP), a fundamental task is to transform the input to some high dimensional space where each dimension tells something significant about the underlying structure of data. A number of research works have been conducted to solve this challenging problem [32, 125]. Mainstream tasks like paraphrase identification, natural language inference, and question-answer pair modelling depend heavily on the quality of these learned representations as the inference is done in the representation space [45, 58, 163, 227, 247]. To enhance these pre-learned representations, we can build a hierarchically higher level representation with or without the help of an external knowledge base [235, 253]. This method can be either task specific or generalized depending on the problem being solved.

Mainstream sentence pair comparison models check only for word level associations [45, 58] and ignore higher level information like phrases. This seems to be contrary to what humans do: we tend to look first for the higher level structures and then investigate whether those structures talk about the same thing or not in the two sentences. For example, when comparing the sentence “*A boy is lying in the snow and is making snow angels*” with the sentence “*Two people wearing snowsuits are on the ground making snow angels*” for the natural language inference task, we can just look at the content portion “*making snow angels*” as a phrase chunk and check if it exists in both the sentences or not to come to a decision.

In this work, we propose an actor-critic-based reinforcement learning (RL) framework to generate phrase level sentence structures whilst solving the sentence pair modelling task. Our RL framework involves three stages of training. In the first stage, we train a hierarchical attention based critic where the hierarchy looks at the word level association within a phrase and the phrase level association within a sentence. In the second stage, we train a policy-based actor where the policy is to decide whether a word represents “inside a phrase” or “end of a phrase”. This policy update [225] is dependent on the performance of a sub-optimal critic and is influenced by a delayed reward to guide the structure discovery [230]. We limit the decision space to be either 0 or 1 and use a simple sampling strategy to explore it. In the third stage, the pre-trained critic is fine-tuned further in order to adapt itself with the actor-generated task specific structures. Our RL model learns to generate reasonably good task specific phrase structures for the underlying sentence pair comparison tasks. The involvement of task specific loss in the reward portion of actor allows us to achieve on par or better performance and the possibly optimal structures at the same time.

## 11.2 Related Work

Generally, sentence pair modelling architectures follow a uniform framework. The sentences to be compared are first projected into a high dimension vector space. A classification module then generates a decision by comparing the vector representations. This projection is done with various types of encoders. 1. *Bag of words models*: represent a text as a multiset of its words, disregarding grammar and word order but keeping multiplicity intact, 2. *Recurrent and Convolutional neural network models*: which take into consideration the word ordering and word contexts, 3. *Transformer models*: where each word gets affected by all the other words through a complex attention module and 4. *Tree Structured models*: that use parse trees as their knowledge base.

[98] and [104] propose *bag of words* type models where they take the average of word vectors in a sentence followed by a series of linear projection layers. Their intuition was that the depth of the series of projection layers can capture the subtle variations in the input order. [137] suggest the use of an auto-encoder with sentiment and domain supervision where they combine sentiment label loss and domain loss in their objective. Their model ignores sentence structure and local context. [163] encode both the hypothesis and premise using one copy of an LSTM and then apply a Manhattan distance based similarity function on top of that for the inference. [141], [247], and [58] follow almost the same framework but after the encoder they have a matching layer followed by a pooling block at the inference stage. [215] and [227] apply tree-based recurrent neural networks as the composition function on the dependency and constituency trees to extract the representation. [270] encode attention inside the dependency tree LSTM using a cross attention mechanism whereas [13] encode attention inside both dependency and constituency tree-LSTMs using a dot-product attention mechanism. [50] extract local and global inference compositions by jointly utilizing both standard and tree-LSTMs.

Recently, fully attention-based models are achieving state of the art performance for many of the complex NLP problems, such as Transformer for the machine translation task [240]. Inside Transformer, the encoder blocks are composed of a stack of multi-head attention modules and the recurrence behavior of an LSTM is replaced by a positional encoding module. These multi-head attention modules use a number of single head dot-product attention modules [175] as the composition function. Most recently, [65] created BERT, a generalized language representation model, using a multi-layer bidirectional Transformer encoder which also provides very good sentence representations. The advantage is that any task specific layer can be plugged in as the final layer without designing any substantial architecture modifications.

Apart from looking at just the raw word level association, some research takes the underlying grammatical structure of sentences into account as well. It is possible for these structures

to be pre-defined or task specific. [227], [270], and [13] work with the tree structures generated by a standard parser whereas [54], [252], and [235] generate task specific latent trees without having any knowledge from a gold standard parser. Most of the works in this category use Kirchhof's matrix tree theorem [41] to mimic the latent grammatical association between words as an  $N \times N$  matrix. The idea is to learn a square attention matrix where each row represents the probability distribution of one word being the head word of other words.

Unlike [146], [221], [249], and [170] our motivation is not doing any phrasal alignment. Instead our goal in this work is to generate meaningful sentence constituents that can aid with solving a downstream task. Of these research works, only [170] is neural network based, where the authors first compute preliminary scores of a source chunk vs. all possible target chunks. These chunks are formed from sentence constituents and siblings extracted from a constituency tree. Finally, a voting mechanism is employed to calculate the aligned source/target token pair scores. This is done by summing the preliminary alignment values for all source/target chunk pairs.

## 11.3 Model

In this section, we describe our model in detail. We first explain how the critic is trained in a delayed manner with and without the actor response. Following this, we explain how we train the actor using the response of a trained critic. We conclude this section by giving a high level view of the workflow of our model.

**Training the Critic:** As critic, we design a two level hierarchical structured model with each level containing a gated recurrent unit (GRU). The word level GRU handles the phrase structures as word sequences whereas the phrase level GRU amalgamates the phrase level information and turns that into a final sentence representation. Using this hierarchically structured model we compute the representations of a pair of sentences and then compare them for an underlying task. As our model is capable of handling phrases, apart from just taking raw word information, it also takes a sequence of words representing a phrase of a certain length existing in a sentence.

In order to extract the phrase level structure we use the Stanford constituency [148] to first come up with a constituency tree of each sentence. Following this, we perform a depth-first traversal of this tree (summarized in Algorithm 11.1) in such a way that if the number of children in a subtree ( $F(\cdot)$  gives this count) is less than a threshold  $U$ , we record a certain number of 0's ending with a single 1 for that subtree. As the threshold, we use  $\sqrt{L} + 0.5$ , where  $L$  is the length of the sentence. At the start, an empty list  $A$  is passed to the algorithm. This list will store the stream of 0's and 1's, called the action sequences ( $\mathbf{a}_t$ 's).

**Algorithm 11.1** Phrase Sequence Generation: PSG(tree, A, U)

---

```

if  $F(\text{tree}) \leq U$  then:
     $A \leftarrow A + (F(\text{tree}) - 1) \times 0$ 
     $A \leftarrow A + 1$ 
else
    PSG(Left Child, A, U)
    PSG(Right Child, A, U)
end if

```

---

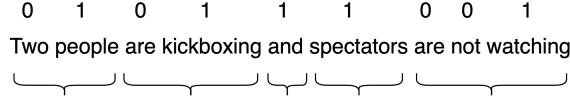


Figure 11.1: Sample example of extracting phrases from a sentence given a stream of 0's and 1's.

$\mathbf{a}_{t-1}$	$\mathbf{a}_t$	Structure type
0	0	Phrase continues at $x_t$
0	1	Phrase ends at $x_t$
1	0	Phrase starts at $x_t$
1	1	$x_t$ is a single-word phrase

Table 11.1: Type of structure according to  $a_{t-1}$  and  $a_t$ .

Next, from this sequence of 0's and 1's, gold label phrase structures are extracted as illustrated in Figure 11.1. During this step, we keep track of the beginning and end of a phrase using the constraints designed in Table 11.1. Once the sentence is divided into phrase chunks, the word-level  $\text{GRU}^w$  traverses each phrase as a sequence of  $T$  words  $\{x_t\}_{t=1,\dots,T}$  from left to right and generates a hidden representation at each time step  $h_t^w$ ,  $\forall t \in [1, \dots, T]$ .

$$\mathbf{h}_t^w = \text{GRU}_t^w(\mathbf{x}_t, \dots, \mathbf{x}_T) \quad (11.1)$$

The transition of the word-level  $\text{GRU}^w$  depends upon action  $a_{t-1}$ . If action  $a_{t-1}$  is End, the word at position  $t$  is the start of a phrase and the word-level  $\text{GRU}^w$  starts with a zero-initialized state. Otherwise, the action is Inside and the word-level  $\text{GRU}^w$  continues from its previous state. We summarize this entire step as follows,

$$\mathbf{h}_t^w = \begin{cases} f(\mathbf{0}, \mathbf{x}_t), & \mathbf{a}_{t-1} = 1(\text{End}) \\ f(\mathbf{h}_{t-1}^w, \mathbf{x}_t) & \mathbf{a}_{t-1} = 0(\text{Inside}) \end{cases} \quad (11.2)$$

where  $f$  denotes all of the gate and update functions from critic  $\text{GRU}^w$ . After this, it applies an attention layer which scales each of the hidden states  $\mathbf{h}_t^w$  according to a scalar weight  $\alpha_j^w$ . This

weight resembles the importance of words within a phrase. It is to be noted that the attention spans over the phrases are independent with respect to each other,

$$\alpha_j^w = \frac{e^{\mathbf{W}_w \tanh(\mathbf{h}_j^w)}}{\sum_{j=1}^k e^{\mathbf{W}_w \tanh(\mathbf{h}_j^w)} + 0.0001} \quad (11.3)$$

$$\tilde{\mathbf{h}}_t^w = \alpha_t^w \mathbf{h}_t^w \quad (11.4)$$

where  $k$  represents the phrase length and  $\mathbf{W}_w$  is a parameter of this attention layer. Following this, it employs a max (or mean) pooling block within a phrase chunk to summarize the scaled hidden states in one phrase-level dense representation as follows,

$$\mathbf{z}_t^p = \text{maxpool}(\tilde{\mathbf{h}}_1^w, \dots, \tilde{\mathbf{h}}_k^w) \quad (11.5)$$

Next, the phrase-level GRU<sup>*P*</sup> traverses each  $z$  as a sequence of  $P$  phrases  $\{z_t\}_{t=1, \dots, P}$  from left to right and generates a hidden representation at each time step  $h_t^p$ ,  $\forall t \in [1, \dots, P]$ .

$$\mathbf{h}_t^p = \text{GRU}_t^p(\mathbf{z}_t, \dots, \mathbf{z}_P) \quad (11.6)$$

Following this, it scales these phrase-level hidden representations  $\mathbf{h}_t^p$  using a set of attention weights  $\alpha_j^p$ . These attention weights are coming from an attention layer having its own set of parameters ( $\mathbf{W}_p$ ). Finally it employs a max (or mean) pooling block to summarize these scaled hidden states in one dense sentence representation. The entire process is summarized as follows,

$$\alpha_j^p = \frac{e^{\mathbf{W}_p \tanh(\mathbf{h}_j^p)}}{\sum_{j=1}^P e^{\mathbf{W}_p \tanh(\mathbf{h}_j^p)} + 0.0001} \quad (11.7)$$

$$\tilde{\mathbf{h}}_t^p = \alpha_t^p \mathbf{h}_t^p \quad (11.8)$$

$$\vec{h} = \text{maxpool}(\tilde{\mathbf{h}}_1^p, \dots, \tilde{\mathbf{h}}_P^p) \quad (11.9)$$

The next steps are to infer the similarity between the two representations  $(\vec{h}_a, \vec{h}_b)$  using standard matching methods and to project the resultant vector into the space of classes,  $y$ , through a series of fully connected layers as follows,

$$x = (\vec{h}_a, \vec{h}_b, |\vec{h}_a - \vec{h}_b|, \vec{h}_a * \vec{h}_b) \quad (11.10)$$

$$P(y|\mathbf{X}) = \sigma(\mathbf{W}_1 \sigma(\mathbf{W}_2 x + \mathbf{b}_2) + \mathbf{b}_1) \quad (11.11)$$



Finally, it is trained by optimizing a task specific loss function as follows,

$$H(p, q) = - \sum_{i=1}^n Q(y_i) \log(P(y_i)) \quad (11.12)$$

Instead of doing the standard training, we initialize two copies of critic: *final* and *active*. This is required in order to make the model adaptable with a delayed update paradigm which is described in the following section. We perform the forward pass using Eqns. 11.1-11.12 with the *final* version of the critic, compute the gradients with respect to its parameters ( $\theta_f = \{\theta_{f_1}, \dots, \theta_{f_k}\}$ ), and store them.

$$\frac{\partial H}{\partial \theta_f} = [\frac{\partial H}{\partial \theta_{f_1}}, \dots, \frac{\partial H}{\partial \theta_{f_k}}] \quad (11.13)$$

In order to mimic the behavior of standard batch-wise training, first a loss with respect to the *final* version of critic gets computed and the corresponding gradients are stored for each sample in the batch. Following this, the *active* version parameters ( $\theta_a = \{\theta_{a_1}, \dots, \theta_{a_k}\}$ ) of critic are updated using the following update mechanism.

$$\theta_{a_j} = \theta_{a_j} + \sum_{i \in \text{batch}} \frac{\partial H_i}{\partial \theta_{f_j}} \quad (11.14)$$

Finally, the parameters of the *final* version of the critic are updated with its *active* version parameters using a straight assignment ( $\theta_f = \theta_a$ ) once an entire batch is looked at.

We adopt another paradigm of training which turns a sub-optimal critic into an optimal one. It involves weighted updating instead of doing a straight assignment as above. We use this paradigm during the fine tuning phase. The update mechanism is as follows,

$$\theta_f = \theta_a \times \alpha + \theta_f \times (1 - \alpha) \quad (11.15)$$

where  $\theta_a \in \{\theta_{a_1}, \dots, \theta_{a_k}\}$ ,  $\theta_f \in \{\theta_{f_1}, \dots, \theta_{f_k}\}$ . Hyperparameter  $\alpha$  is set to 0.1 in all experiments.

**Training the Actor:** We adopt the policy gradient method [225] to update the actor along with a delayed reward. We use the following policy to sample an action  $a_t$  at each time step  $t$ ,

$$\pi_{\theta}(a_t | \mathbf{s}_t; \theta) = \max(e^{-5}, \min(\sigma(\mathbf{s}_t \mathbf{W} + \mathbf{b}), (1 - e^{-5}))) \quad (11.16)$$

Here  $\pi_{\theta}(a_t | \mathbf{s}_t; \theta)$  denotes the probability of choosing  $a_t$  and  $\{\mathbf{W}, \mathbf{b}\}$  is the set of parameters of the actor policy network. We include the representation of the counterpart sentence, the current input, the hidden representation of word-level GRU<sup>w</sup> and the hidden representation of

phrase-level GRU<sup>p</sup> as state. Formally, it is defined as

$$\mathbf{s}_t = [\tilde{\mathbf{h}}_T, \mathbf{x}_t; \mathbf{h}_t^w; \mathbf{h}_{t-1}^p] \quad (11.17)$$

Here,  $\tilde{\mathbf{h}}_T$  is the summary vector of the counterpart sentence generated by the critic GRU. We use the same policy network to sample actions for the two sentences being compared. Using this state and policy, we perform action sampling for the whole sequence to obtain the delayed reward [259] as follows,

$$action = \begin{cases} 1, & \text{if } P(\mathcal{Y}) > \pi_\theta \\ 0, & \text{if } P(\mathcal{Y}) < \pi_\theta \end{cases} \quad (11.18)$$

where  $\mathcal{Y}$  is a uniform random variable and  $\pi_\theta$  is the policy from Eqn. 11.16. For each sentence in the pair, the parameters of our policy net are optimized using the REINFORCE algorithm [243] defined as follows,

$$\begin{aligned} J(\theta) &= \mathbb{E}_{(\mathbf{s}_t, a_t) \sim P_\theta(\mathbf{s}_t, a_t)} r(\mathbf{s}_1 a_1 \dots \mathbf{s}_T a_T) \\ &= \sum_{\mathbf{s}_1 a_1 \dots \mathbf{s}_T a_T} P_\theta(\mathbf{s}_1 a_1 \dots \mathbf{s}_T a_T) R_T \\ &= \sum_{\mathbf{s}_1 a_1 \dots \mathbf{s}_T a_T} P_\theta(\mathbf{s}_1) \prod_t \pi_\theta(a_t | \mathbf{s}_t) P_\theta(\mathbf{s}_{t+1} | \mathbf{s}_t, a_t) R_T \end{aligned} \quad (11.19)$$

Since our states are fully deterministic, we can make  $P_\theta(\mathbf{s}_1) = P_\theta(\mathbf{s}_{t+1} | \mathbf{s}_t, a_t) = 1$  in Eqn. 11.19. The final objective function involves two terms to cover the action sampling for both of the sentences,

$$J(\theta) = \sum_{\mathbf{s}_1 a_1 \dots \mathbf{s}_T a_T} \prod_{t \in l} \pi_\theta(a_t | \mathbf{s}_t) R_T + \sum_{\mathbf{s}_1 a_1 \dots \mathbf{s}_T a_T} \prod_{t \in r} \pi_\theta(a_t | \mathbf{s}_t) R_T \quad (11.20)$$

where  $l$  denotes the left sentence and  $r$  denotes the right sentence. By applying the likelihood ratio trick, we update the policy network with the following gradient

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{t \in l} R_T \frac{\partial}{\partial \theta} \log \pi_\theta(a_t | \mathbf{s}_t, \theta_t) + \sum_{t \in r} R_T \frac{\partial}{\partial \theta} \log \pi_\theta(a_t | \mathbf{s}_t, \theta_t) \quad (11.21)$$

The objective is to maximize the expected delayed reward which is computed using the logarithm of the output probability distribution of the critic,  $\log P(y|\mathbf{X})$ , over just one sample. The policy network controls the number of deleted words by jointly maximizing a unimodal

function  $f(x) = x + 0.1/x$  and the reward. The final reward  $R_T$  is defined as

$$R_T = \log P(y|\mathbf{X}) - \frac{\gamma Z}{2} \left( (L'_l/L_l + 0.1L_l/L'_l - 0.6 + (L'_r/L_r + 0.1L_r/L'_r - 0.6)) \right) \quad (11.22)$$

where  $L'_l$  and  $L'_r$  denotes the number of phrases predicted by the actor, and  $L_l$  and  $L_r$  are the actual lengths for the left and right sentences, respectively.  $\gamma$  is a hyper-parameter and  $Z$  is the number of classes. The minimum of  $f(x)$  is  $1/\sqrt{10} = 0.316$  which encourages the actor to predict 3 ~ 4 phrase for a sentence with length 10.

Like the critic, we initialize two versions of the actor: *final* and *active*. For each sample in a batch, the *active* version of the actor policy network is updated by the gradients of the *final* version of the actor policy network parameters

$$\theta_{a_j} = \theta_{a_j} + \sum_{i \in \text{batch}} \frac{\partial J(\theta)_i}{\partial \theta_{f_j}} \quad (11.23)$$

Once a batch is looked at, the *final* version of the actor policy is updated in a similar way as the critic using Eqn. 11.15

**Workflow:** We now give a high level view of our entire model's workflow which has been strongly influenced by the two algorithms found in Chapter 10.

We start by training the critic in a delayed strategy to get a sub-optimal version by first initializing two copies of it (*final* and *active*). The *active* version gets updated by the *final* version gradients computed for each sample in a batch. During this, we keep the parameters of the *final* version fixed. Once the entire batch is looked at, we use the *active* version parameter values to update the *final* version parameters through a straightforward assignment.

After training this sub-optimal critic, we use a weighted delayed strategy to train an actor with a policy gradient method. We start by initializing two copies of actor (*final* and *active*). For each sample in the batch, a state is calculated using Eqn. 11.17 and later used in Eqn. 11.16 to obtain the policies ( $\pi_\theta$ 's) for every word appearing in the two participating sentences in a sample. Next, a set of actions ( $a_t$ 's) are calculated using the sampling strategy in Eqn. 11.18 using the prior calculated policies. As mentioned before, these actions represent whether a word belongs to the inside (0) or is the end (1) of a phrase. Each action in the current as well as the previous time step jointly represents a type of structure according to Table 11.1. Next, the sentences in a data sample along with these sampled actions are used with a trained critic to calculate a reward ( $R_T$ ). This reward is then regularized with a term that reflects the percentage of the number of phrases that the actor predicts from both of the sentences (the subtrahend in Eqn. 11.22). We are not only motivated to get the predicted sample correct but also to get around 3 or 4 phrases in a sentence of length 10. It is to be noted that unlike  $s_t$ 's and  $a_t$ 's,

these rewards ( $R_T$ 's) are calculated for each sample in the batch rather than for every word of a sentence in a sample. We repeat this strategy  $N$  times and store all of the  $s_t$ 's,  $a_t$ 's and  $R_T$ 's. Next, we calculate an average reward. We compute the gradients of Eqn. 11.21  $N$  times using the  $N$  sets of the logarithm of policies and the  $N$  differences between the  $N$  rewards and the average reward. Finally, for each actor parameter, we add its  $N$  gradients and update it using a weighted delayed strategy (Eqn. 11.15).

After the actor is trained, the previously trained sub-optimal critic is further tuned using the response of this trained actor. Rather than looking at the sentence pairs and their real phrase structure, now the critic looks at the actor-predicted phrase structures of those sentences. For each sample in the batch, a loss is calculated based on the critic response and the *active* version is updated through the gradients of the parameters of the *final* version as before. It should be mentioned that sometimes the actor can err by generating a weird phrase structure due to the random sampling strategy. To counteract this, the update of the critic should not rely entirely on the actor's decision during this fine-tuning phase. So with the *active* version parameter values, instead of using a straight assignment as before, we use Eqn. 11.15 to update the *final* version of the critic once an entire batch is looked at.

## 11.4 Datasets, Experimental Setup, Results and Analysis

In this section, we explain the experimental setup along with a thorough analysis of the results obtained. We first describe our training corpora as well as their statistics in terms of number of samples: train, test and validation. Following this, we explain the technical details of our proposed architecture along with its hyper-parameter settings. We also present a comparison between the results obtained with our RL model and some of the top performing models on these aforementioned datasets. Additionally, we give a qualitative analysis by showing the predictions of our models on some random test samples for all of the tasks. Finally, we conclude this section by giving some insight into the performance of our model by analyzing the generated structures.

**Datasets:** We use three datasets for our model evaluation: paraphrase identification, natural language inference, and question-answer pair modelling.

- **MSRP:** Given a pair of sentences, the task is to identify whether or not they are paraphrases of each other [69]. Train:4076; Test:1725; Valid:N/A.
- **SICK:** The dataset contains sentences derived from video and image annotations and the task is to classify a given sentence pair into three classes: Entailment, Neutral and Contradiction [151]. Train:4500; Test:4927; Valid:500.

- **AI2-8grade:** The task is to do a true-false question selection where each data sample consists of a pair of sentences with one being the question and the other being the evidence. The evidences are formed by replacing the *wh* in the question with an answer [30]. Train:12689; Test:11359; Valid:2483.

**Experimental Setup:** The GRU hidden dimension is set to 650 for the MSRP dataset, 700 for the SICK dataset and 400 for the AI2-8grade dataset. Word vectors are initialized with the 300 dimension GloVe embeddings [183] and are not updated during training. To smooth the update during critic training, the gradients are divided by  $B^2$  where  $B$  is the batch size which is 5. Next, we add the gradient norms of all the parameters as  $L = \sqrt{\sum_{i=1}^k \|\nabla \theta_i^2\|}$ . The learning rate is reduced by a factor of  $\frac{5}{L}$  if  $L$  is more than 5. To smooth the policy gradient update, we add a  $\gamma$  and  $Z$ -scaled regularized reward as shown in Eqn. [11.22]. The value of  $\gamma$  is set to 0.1 and  $Z$  is the number of classes. During training, the critic parameters are updated using stochastic gradient descent [38] with classification layer dropout values and learning rates ranging from 0.01 to 0.50. For training the actor, we use Adam [109] to update the parameters with a varying range of classification layer dropout values and learning rates (i.e., 0.01-0.50). During random sampling, the sample count  $N$  is set to 5 by validating on the validation sets.

**Results and Analysis:** Table [11.2] shows the performance of our model on the three datasets compared against some of the top performing models. It is to be noted that we limit our comparison to CNN, LSTM and Transformer based models which are of similar size. We do not compare with very large models such as BERT, OpenAI GPT and XLNet as they have billions of parameters and leverage almost the entire internet text in their training. Some of the models that we are comparing against are designed for a specific task [30, 39, 101, 122, 163, 264] and some of them are designed for general purpose [9, 58, 135, 263]. Apart from these, we also compare against InferSent, LSTM, BiGRU Last Encoder, Inner Attention and ConvNet Encoder using the official implementations<sup>1</sup> but with the same settings as ours. As can be seen, our Critic + RL is getting a fair amount of boost on the MSRP and SICK datasets but not a significant gain on the AI2-8grade dataset. Also our Critic + RL model is getting better performance than all the other models on MSRP (75.35% Acc.) and SICK (85.10% Acc.) datasets and is only behind ConvNet Encoder [263] (75.26% Acc.) on the AI2-8grade dataset (74.78% Acc.). These results indicate that our actor has generated some good task specific phrase structures which allows our Critic to perform even better than what it was doing with the standard parser generated phrase structures.

Table [11.3] shows how our model is performing on some examples from the held out test set of the three datasets. In the first two examples from the SICK dataset, the phrase cutoff

<sup>1</sup> Available: <https://github.com/facebookresearch/InferSent>

Model	MSRP Acc.	AI2-8grade Acc.	SICK Acc.
ConvNet Encoder [263] †	73.96	<b>75.26</b>	83.82
Inner Attention [135] †	69.74	74.73	72.01
BiGRU Last Encoder [58] †	70.46	74.61	81.47
LSTM [58] †	70.74	74.93	76.80
InferSent [58] †	74.46	74.71	84.07
InferSent + RL [9] †	74.74	73.84	84.57
Critic	75.07	74.74	84.43
Critic + RL	75.35	74.78	<b>85.10</b>
Seq-LSTMs [270]	71.70	63.30	-
Seq-GRUs [270]	71.80	62.40	-
Tree LSTM [270]	73.50	69.10	-
Tree LSTM + Attn. [270]	<b>75.80</b>	72.50	-
Tree GRU [270]	73.96	70.60	-
Tree GRU + Attn. [270]	74.80	72.10	-
RNN [30]	-	36.10	-
CNN [30]	-	38.40	-
RNN-CNN [30]	-	37.60	-
Attn1511 [30]	-	35.80	-
Ubu.RNN [30]	-	44.10	-
Illinois-LH [122]	-	-	84.60
UNAL NLP [101]	-	-	83.10
ECNU [264]	-	-	83.60
SNLI-Transfer 3-class LSTM [39]	-	-	80.80
MaLSTM features + LSTM [163]	-	-	84.20

Table 11.2: Performance comparison of our model on different tasks against some existing top performing models. We mark models that we implemented as †. We only compare against those models that are similar in terms of the number of parameters to ours, ignoring large models such as BERT, OpenAI GPT and XLNet.

point is at the same place for both the hypothesis and the premise. In the first example, “slicing an onion” gets compared against “cutting an onion” whereas in the second example “putting a baby in a waste bin” gets compared against “putting a child in a waste bin”. In the third example, even though we are getting different cutoff points, the phrase structures from both sentences are similar (i.e. “on a park”-“on a bench”, “people”-“men” and “sitting”-“dressed”). Sometimes the actor generates single word phrases because if that word is placed inside a phrase, attention on it will be distributed over the other participant words. This behavior is very dominant in the samples from the MSRP dataset. In the first two examples from MSRP dataset, most of the phrases are single worded. However, in the third example we see larger structures as well as a correct prediction: “Venture Exchange composite gained N points” and

Dataset	Sentence 1	Sentence 2	GT	Pr
SICK	A person   is slicing an onion	One person   is cutting an onion	E	E
	A woman is   putting a baby in a waste bin	A woman is   putting a child in a waste bin	E	E
	Two   people   are   sitting   on a park   bench   on a sunny day	Two   men   dressed   in   white and   black   are   sitting   on a bench	N	N
MSRP	There   are   N Democrats in the   Assembly   and N Republicans	Democrats   dominate   the   Assembly   while   Republicans   control   the   Senate	0	0
	Licensing   revenue   slid   N   percent   however   to   N million	License sales a   key   measure of demand   fell   N percent to N million	1	1
	The composite rose N points on the week while   the   TSX   Venture Exchange composite gained N points	On   the   week   the   Dow Jones industrial average rose   N points while   the   Nasdaq Stock Market gained N points	0	0
AI2-8grade	Abnormal   cell division may result in   cancer	Mutations   in   and tumor suppressor genes may lead   to   cancer	1	1
	Heat   energy   from the   Sun   is transferred to   Earth   primarily by   conduction processes	The bonfire from the opening   image has a lot of thermal energy	0	0
	The   growth   of Trees   usually occurs first   in   primary succession on a bare rock	This is the species that first   lives in   the   habitat	0	0

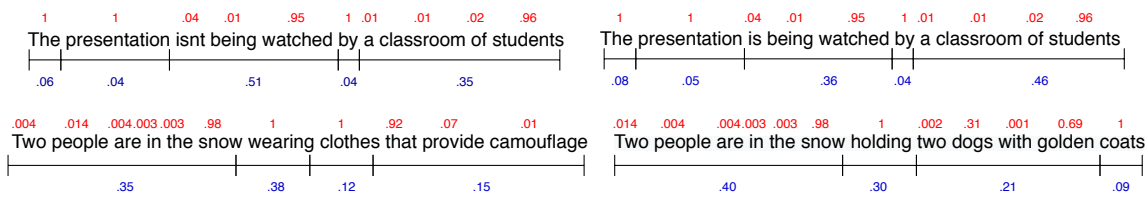
Table 11.3: Example predictions from the test set. **GT**: ground truth, **Pr**: predicted.

Figure 11.2: Attention weights assigned by our critic. Red indicates attention over the words within a phrase. Blue indicates attention over the phrases within a sentence.

“Nasdaq Stock Market gained N points”. In the AI2-8grade dataset, the sentences are quite long and we get very good phrase structures compared to the other two datasets. If the class label is 1, the phrase structures from both sentences are very much similar, whereas for class label 0, the model first provides a meaningful phrase chunk from both sentences and then does the comparison based on that.

Figure 11.2 shows the amount of attention our critic is putting over the raw words and the actor predicted phrases in some examples. In the first example, the phrase structures of the



Dataset	$w$	$1 - w$	Acc.
MSRP	0.12	0.88	75.40
AI2-8grade	0.07	0.93	74.78
SICK	0.35	0.65	85.81

Table 11.4: The  $w$  values that give the best results when combining the Critic ( $w$ ) with the trained critic + RL ( $1 - w$ ).

hypothesis and premise are exactly the same and phrases like “isn’t being watched” and “a classroom of students” from hypothesis and “is being watched” and “a classroom of students” from the premise are getting high attention as their contents should be focussed on. The class decision of this example is contradiction and our Critic + RL model correctly predicts it. In the second example, our actor provides reasonably good phrase structures, too. Some of them are single word phrases (i.e., “wearing”, “clothes”, “holding”) and some are longer phrases (i.e., “Two people are in the snow” and “that provide camouflage”). As can be seen, the generated phrases from the hypothesis and premise are getting similar attention weights (i.e., the first two phrases have the major contents and they are getting 73% attention weight in both sentences). Finally, our Critic + RL model predicts the class decision as neutral which is correct.

We also perform an ensemble experiment involving both our Critic and Critic + RL models where we take the weighted average of their decisions. We assign a weight  $w$  over Critic’s decision and  $1 - w$  over Critic + RL’s decision. Table 11.4 reports the  $w$  values along with the performance of this experiment. Accuracy jumped from 74.35% to 75.40% for MSRP, 85.10% to the 85.81% for SICK and there is no jump for the AI2-8grade dataset. However, as seen before, the phrase structures of the samples from AI2-8grade dataset are comparatively better than the other two datasets, so a significant improvement with an ensemble approach should not be expected.

As our actor training involves exploring the action space using random sampling, we perform an additional experiment to verify whether by giving some random phrase structure can the critic perform better or not. We also create additional phrase structures (trigram, fourgram, fivegram and sixgram) assuming the actor produces these perfect  $n$ -grams. Finally, we also consider two extreme conditions: 1) every word in the sentence is a phrase itself (all 1s), and 2) all the words in the sentence fall under just one phrase ( $n - 1$  0s, single 1). The four random experiments, all with different seeds, give us some indication of baselines. We do not analyze AI2-8grade dataset because the default constituency structure is already giving almost all single word phrases. Table 11.5 summarizes all these experiments.



Phrase Structure	Model	MSRP Acc.	SICK Acc.
all 1s	Critic	75.18	85.02
	Critic + RL	74.46	85.18
$n - 1$ 0s, single 1	Critic	74.79	84.78
	Critic + RL	75.12	85.06
Random 1 and 2	Critic	73.74	81.06
Random 3 and 4	Critic	72.79	82.77
Trigram	Critic	74.63	84.01
Fourgram	Critic	75.01	82.95
Fivegram	Critic	75.01	83.28
Sixgram	Critic	75.01	82.12

Table 11.5: Comparison on artificial phrase structures.

## 11.5 Conclusion

In this chapter, we propose an actor-critic-based reinforcement learning framework to train a sentence pair modelling architecture. We divide our learning framework into three stages. In the first stage, our critic looks at the real (i.e., parse tree generated) phrase level view of a sentence pair when comparing them to solve a specific task. In the second stage, our actor learns to generate task specific artificial phrase structures using a policy gradient method. To do that, it not only takes just the phrase and word level information of each sentence but it also considers the mutual information shared between the two sentences. Finally, in the third stage, the already trained critic gets fine-tuned to adapt with the actor generated artificial structures without compromising the actual performance. Results show that our model generates some meaningful phrase structures and gets on par or better performance compared to the existing models.

## Chapter 12

# Multilingual Semantic Textual Similarity using Multilingual Word Representations

This chapter is based on the paper titled “Multilingual Semantic Textual Similarity using Multilingual Word Representations” co-authored with Chahna Dixit, Robert E. Mercer, Atif Khan, Muhammad Rifayat Samee, and Felipe Urrea that appeared in the 14th International Conference on Semantic Computing (ICSC 2020) [5].

In Natural Language Processing, doing the semantic textual similarity (STS) task between monolingual sentences is itself taxing. In this chapter, we extend this problem to the multilingual STS scenario, making it even more challenging. We approach this problem using a multilingual representation of words where words having similar meaning across different languages are aligned. We prepare a very large multilingual STS dataset by scraping several multilingual government, insurance, and bank websites. Because these websites have only positive examples for doing multilingual STS we develop an algorithm for generating negative examples using latent Dirichlet allocation and OpenAI-GPT. We are able to train well performing models for datasets combining English, French, and Spanish. We get very good transfer performance across languages as well as domains. We also show our model’s state-of-the-art paraphrase detection performance on the Microsoft Research Paraphrase Corpus.

### 12.1 Introduction

Semantic textual similarity (STS) is a task that determines the extent to which two given content pieces are similar in meaning. It is considered one of the most important natural language processing (NLP) tasks. Similar to most other NLP tasks, a fundamental challenge in this task is to learn meaningful and high-quality input representations by transforming the input space to a high dimensional vector space where each dimension captures important aspects of the

underlying structure of the data [32, 125]. The task is already a challenging one in the monolingual setting, i.e., when computing similarity for two content pieces in the same language. In this work, we focus on the more difficult case of multilingual semantic textual similarity which deals with estimating the similarity of two content pieces in different languages.

Traditional approaches for the STS problem [264] using Wordnet [157] and latent semantic analysis [64] to generate sentence representations have recently been overtaken by neural network-based approaches [58], [45], and [163]. But these latter works focus on a single language. Previous works on handling multilingual STS problems include the use of machine translation (MT) systems limiting the model scope to the monolingual space [142]. Adding an MT system causes a severe increase in model complexity. One other approach applies bilingual word representations that bypasses the complexity induced by an MT system but fails to take advantage of the increasing amount of data available for more than one language pair [16].

A large amount of research exists to produce efficient word embeddings for multiple languages. But, for most of them, the embeddings are not aligned across different languages. However, two works [60, 103] that are interested in aligning word representations across multiple languages are widely used. The first work [60] uses an adversarial training strategy where a discriminator learns to discriminate between mapped source and target embeddings and the mapping. The second work [103] maps two independently trained monolingual embeddings using a linear projection matrix with constraints.

In this chapter, we approach multilingual STS as a binary classification task and learn a language-agnostic, scalable model by utilizing multilingual word representations. We limit the scope of our problem to enterprise business content which led us to collect large multilingual corpora from different domains (government, insurance, and banks). For cases where only positive pairs are available, we propose an algorithm to generate realistic negative samples by including knowledge from topic modelling [37] and OpenAI-GPT [188] representations. Our experimental results show that the learned model can be used for other languages or domains irrespective of the language or domain used for training. We further show the value of our work by achieving state-of-the-art transfer learning (along with minor fine-tuning) performance on the publicly available Microsoft Research Paraphrase corpus [69].

## 12.2 The Model

In this section, we describe our model in detail. At first we briefly explain our model's architecture and how it gets trained. Following this, we give an overview of the dataset collection followed by negative examples generation without having any prior knowledge about them.

### 12.2.1 Model

We have chosen to use Infersent [58], an LSTM based model, to compute the representations of a pair of sentences,  $a$  and  $b$ , and then compare the representations for an underlying task. LSTMs are a standard tool to encode a sentence. They maintain contextual information of the input by integrating a loop allowing information to flow from one word to the next. The model first traverses each sentence as a sequence of  $T$  words  $\{x_t\}_{t=1,\dots,T}$  from both left to right and right to left and generates two hidden representations at each time step  $\vec{h}_t, \overleftarrow{h}_t \forall t \in [1, \dots, T]$ . During input, it considers the vector representation of each word ( $x_t$ ) in the sentence from a pre-trained word embedding model and these representations are not further trained with the network parameters.

$$\begin{aligned}\vec{h}_t &= \overrightarrow{\text{LSTM}}_t(x_1, \dots, x_T) \\ \overleftarrow{h}_t &= \overleftarrow{\text{LSTM}}_t(x_1, \dots, x_T) \\ h_t &= [\vec{h}_t, \overleftarrow{h}_t]\end{aligned}\tag{12.1}$$

After this, the model employs a max (or mean) pooling block to summarize the hidden states in one dense representation.

$$h = \text{maxpool}(h_1, \dots, h_T)\tag{12.2}$$

The next steps are to infer the similarity between the two representations ( $h_a, h_b$ ) using standard matching methods and to project the resultant vector into the space of classes (which is two in our case),  $y$ , through a series of fully connected layers as follows

$$x = (h_a, h_b, |h_a - h_b|, h_a * h_b)\tag{12.3}$$

$$P(y|\mathbf{X}) = \sigma(\mathbf{W}_1 \sigma(\mathbf{W}_2 x + \mathbf{b}_2) + \mathbf{b}_1)\tag{12.4}$$

In Eqn. 12.3,  $x$  represents the concatenation of all of the feature vectors that we have calculated. As feature vectors, we used the two representations as is, the absolute value of their difference, and their componentwise product. The absolute value tells the distance between the vectors in the representation space. The componentwise product provides an elementwise comparison of the signs of the input representations: positive if the signs are the same, negative if different. In Eqn. 12.4,  $P(\cdot)$  represents the model's predicted probability distribution over classes using the feature vector  $x$ .  $\mathbf{W}$ 's and  $\mathbf{b}$ 's are the weights and biases of the classifier network. Finally, the model is trained by optimizing a task specific loss function as follows

$$H(p, q) = - \sum_{i=1}^n Q(y_i) \log(P(y_i))\tag{12.5}$$

Dataset	Domain	Language pairs	#Sentence pairs
<b>EN-FR-G</b>	Government	English-French	158,639
<b>EN-FR-IB</b>	Insurance, Banking	English-French	24,616
<b>EN-ES-IB</b>	Insurance, Banking	English-Spanish	27,907

Table 12.1: Details of collected datasets.

Here,  $Q(\cdot)$  is a one hot vector representation of the true probability distribution of the actual label. The multiplication of  $P(\cdot)$  and  $Q(\cdot)$  indicates how close the model's decision over the actual class is with respect to the true distribution. Finally, rather than maximizing this objective, we put a negative sign in front and minimize it.

## 12.2.2 Dataset Collection and Preprocessing

We scraped several bilingual websites containing webpages with similar content but in two languages. The four essential steps in data collection are: data scraping, HTML parsing, text translation and text alignment. Data scraping yields a set of parallel HTML files which are parsed to extract clean raw text using the Python library *inscriptis* [241]. We retain only those parallel text files that have an equal number of lines as this is important for our text alignment step. Our data collection approach is designed for the English language; hence, we translate all of the text files for the counterpart language into English using the Python library *mtranslate* [17]. After translating the non-English text files, we align the corresponding lines based on the hypothesis that the contents of two parallel text files should be somewhat in the same order. For a given pair of parallel text files, we use word frequency based cosine distance to measure the distance between the lines in each corresponding line pair. The line pairs with cosine distance greater than 0.6 are considered to be misaligned. The parallel text files with misaligned lines are manually aligned by rearranging or discarding certain lines. We extract the semantically similar sentence pairs (called positive pairs) from the final set of aligned parallel text files. As part of the preprocessing, each sentence is tokenized using the NLTK sentence tokenizer and unique sentence pairs are used to form the dataset. We restrict the minimum and maximum length of sentences (total words) to 4 and 200, respectively. The minimum was chosen to remove the non sentences (e.g., titles) and the maximum was chosen because of LSTM's learning limitations.

We have chosen websites in the government, insurance, and banking domains, but this type of data collection approach can be adopted for any other industry vertical that has a bilingual website. The languages of interest here are English, French, and Spanish and we end up creating three datasets based on the language and domain. Details of these datasets are shown in Table 12.1. A more detailed explanation of these steps can be found in Chapter 13.

The above mentioned procedure yields positive pairs; however, in order for the model to

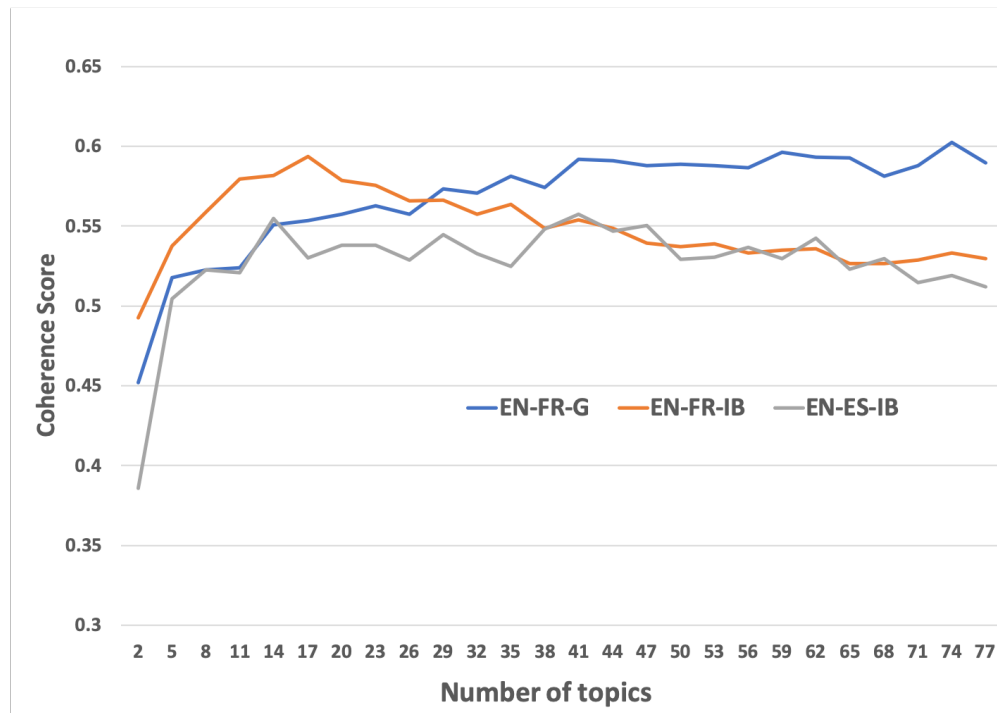


Figure 12.1: Topic coherence score vs number of topics.

differentiate between similar and not similar pairs, we also need sentence pairs that are not semantically similar (i.e., negative pairs). In the next subsection, we explain the approach for generating negative sample pairs.

### 12.2.3 Negative Sample Pairs Generation

We define a negative sample pair as the sentence pair having similar topic, but describing some different aspect of the topic. For a given query sentence from the positive sample pairs, its corresponding negative sentence is formed by sampling from different webpages having similar topic as that of the query sentence.

We start by training a range of unsupervised LDA [37] models on the English versions of the three datasets. Each webpage is considered as a single document. The LDA model with maximum coherence score is chosen as the best topic model. Figure 12.1 shows the coherence score vs. number of topics plot for the **EN-FR-G**, **EN-FR-IB**, and **EN-ES-IB** datasets where the optimal number of topics are 74, 17 and 41, respectively. We use the following parameters for training LDA: *random\_state*=100, *update\_every*=1, *chunksize*=100, *passes*=300, *alpha*=auto, *per\_word\_topics*=True.

After training and selecting the best LDA model, we infer the vector representation of each English document in the dataset with a document vector. Next, we use a pre-trained

**Algorithm 12.1** Negative sample selection with LDA-LM

---

```

Pretrained LDA model:  $L$ 
Pretrained OpenAI-GPT model:  $M$ 
Input English document:  $D$ 
Topic of  $D$  according to  $L$ :  $T$ 
Set of negative samples:  $N$ 
List of documents with same topic as  $T$ :  $A$ 
Number of sentences to be selected:  $n$ 
Input sentence from document  $D$ :  $S$ 
 $N \leftarrow \emptyset$ 
for  $i < n$  do
   $x \leftarrow \text{NULL}$ 
  for document  $d \in A$  do
    for sentence  $s \in \text{document } d$  do
      if  $0.80 < \text{Cosine}(M(S), M(s)) < 0.90$  then
         $x \leftarrow s$ 
        break
      end if
    end for
    if  $x \neq \text{NULL}$  then
      break
    end if
  end for
   $N \leftarrow \text{multilingual counterpart of } x$ 
   $i \leftarrow i + 1$ 
end for

```

---

OpenAI-GPT model [188] to get the vector representation of each sentence in each English document. Then, for each sentence  $S$  in an English document, its most relevant topic  $T$  is obtained according to the topic distribution of the document. Next, in the set of documents  $A$  having the same topic  $T$ , we collect the sentences (and their multilingual counterparts) having cosine similarity with sentence  $S$  in the range 0.8-0.9. Based on our definition of the negative samples, we choose the similarity threshold range to be 0.8-0.9 by experimenting with different range of values; which gives us samples that belong to a similar topic. However, this threshold range can be adjusted based on the application requirements.

Then, we generate  $n$  such sentences to create the  $n$  negative sample pairs in the multilingual space by pairing  $S$  with the appropriate multilingual counterparts of each of these sentences. Note that we have chosen  $n$  to be 10 in order to obtain sufficient negative samples, but not all of these 10 pairs are used for creating the final corpus. Algorithm 12.1 makes precise the above steps. A more detailed explanation of these steps can be found in Chapter 13.

Dataset	Train	Validation	Test
EN-FR-G	195,303	48,826	61,033
EN-FR-IB	29,546	7,389	9,237
EN-ES-IB	34,447	8,613	10,766

Table 12.2: Sentence pair counts for dataset partitions.

## 12.3 Experimental Setup and Analysis

In this section, we explain the experimental setup and provide the results obtained together with a thorough analysis. We first describe our training corpora as well as the public benchmark dataset [69] used in other standard sentence pair modelling studies. Following this, we explain the technical details of our proposed architecture along with its hyper-parameter settings. We also present the detailed results obtained with our model and compare with some of the top performing models on our selected public dataset.

### 12.3.1 Dataset Preparation

For each of the three datasets, we use the positive and negative sample pairs, described earlier, to create a balanced 10-fold training and validation partition for 10-fold cross validation experiments. We also create a test set for testing. The dataset partition details are given in Table 12.2. For evaluation on a public dataset, we chose the Microsoft Research Paraphrase corpus where the task is to do paraphrase identification. Because of the way we prepare our corpus, it aligns well with this kind of task. This dataset has 9,877 sentence pairs, 8,152 in the training set and 1,725 in the test set.

### 12.3.2 Model Parameters and Training Details

The LSTM hidden state dimension is set to 600. Multilingual word vectors are initialized with the 300 dimension MUSE embeddings [60] and are not updated during training. To smooth the update, the gradients are divided by  $B^2$  where  $B$  is the batch size which is set to 512. The learning rate is decaying proportional to  $\frac{maxNorm}{\sqrt{\sum_{i=1}^k \|\nabla \theta_i^2\|}}$  if  $\sqrt{\sum_{i=1}^k \|\nabla \theta_i^2\|}$  is more than  $maxNorm$ , which is 5 for our experiments. We use Adam as the optimization algorithm and the dropout in the classification layer is set to 0.5.

### 12.3.3 Results and Analysis

Table 12.3 shows a few examples from our dataset and our model's predictions for them. The pairs which are tagged as negative are created using Algorithm 12.1 whereas the positively



Dataset	Sentence 1	Sentence 2	GT	Pr
EN-FR	The Cannabis Act proposes many rules that would protect youth from accessing cannabis.	Le projet de loi sur le cannabis prévoit de nombreuses dispositions pour empêcher les jeunes d’avoir accès au cannabis.	1	1
	The authorized health care practitioner’s licence information	Numéro de téléphone et adresse électronique de la personne morale	0	0
	What can be deducted from an employee’s pay cheque?	Quand l’employeur doit-il verser l’indemnité de congé annuel?	0	0
EN-ES	Use window sheet kits	Usa kits de aislamiento para ventanas	1	1
	It will only take a minute and won’t impact your credit score	Díganos quién es y qué le gusta, para ver qué ofertas están	0	0
	List out your debt	Fíjate un presupuesto semanal, empezando el lunes	0	0

Table 12.3: Example predictions from the test set. **GT**: ground truth, **Pr**: predicted.

Model	Validation set Accuracy		
	Mean	Max voting	Avg. voting
<b>EN-FR</b>	95.41±0.39	95.76	95.97
<b>EN-ES</b>	96.35±0.57	97.07	97.22
<b>EN-FR-ES</b>	95.03±0.24	95.40	95.59

Table 12.4: 10-fold cross validation performance of different models (mean includes standard deviation).

tagged pairs are provided from the English-French and the English-Spanish aligned sentences from the appropriate web pages. It can be seen in **EN-FR**, the two negative sentences talk about the same topics as their counterpart English sentences, but the content differs. In **EN-ES**’s second pair, the English sentence talks about a credit score while the Spanish sentence talks about some offers which are somehow related. Here in the third pair, the English sentence talks about debt whereas the Spanish sentence talks about budget, which are not exactly related but somehow gets used in the same context.

Table 12.4 shows the 10-fold cross validation performance of three models trained with different data based on the language pairs and domains. The model **EN-FR** is trained on the government domain, whereas **EN-ES** and **EN-FR-ES** are trained on the insurance and bank domains. We summarize the performances over all the folds in three different ways. Firstly, we report the mean and standard deviation of all performances. Following this, we report the results of the ensemble experiment where we use max voting and average voting as our ensemble methods. It can be seen that the average voting achieves the better performance among all of these methods getting 95.97%, 97.22% and 95.95% accuracy for **EN-FR**, **EN-ES** and **EN-FR-ES**, respectively.

Model	Test set accuracy		
	en-fr	en-es	en-en (MSRP)
<b>EN-FR</b>	95.64	95.91	76.05
<b>EN-ES</b>	87.15	97.25	75.07
<b>EN-FR-ES</b>	94.91	98.34	76.00

Table 12.5: Cross corpus performance on test set (Accuracy). Rows indicate training language pairs, and columns indicate testing language pairs.

Model	Accuracy
InferSent [58]	74.46
LSTM [58]	70.74
BiGRU Last Encoder [58]	70.46
Tree LSTM [227]	73.50
ConvNet Encoder [263]	73.96
Ours (Transfer + Finetuning)	<b>76.05</b>

Table 12.6: Performance comparison of our model on the MSRP dataset against some existing top performing models.

Table 12.5 reports the cross corpus performance of the three models (models are in uppercase and datasets are in lowercase). We have not included **en-fr-es** for the test purpose because samples in this dataset are taken from both **en-fr** and **en-es**. The performance scores are reported over the test set that we create for each of these datasets as shown in Table 12.2. We have chosen to use the best model on each of these language pairs out of the 10 models that we create during the 10-fold cross validation. It can be seen that **EN-FR** shows very good performance on **en-es** (95.91%) and it is doing better than its own test set (95.64%). However, when we test the **EN-ES** model on **en-fr** dataset, the performance drops with respect to when the test set is **en-es**; the relatively smaller size of training data for **EN-ES** as compared to **EN-FR** can be one of the reasons for this performance drop. When trained on **en-fr-es** the performance on the other two language pairs compare well. We also report the performance of the models when tested on MSRP which is an **en-en** corpus. The performance on this dataset is elaborated in Table 12.6.

Table 12.6 reports the performance of our model on the MSRP task compared to some of the existing top performing models. As we can see, Infsent trained on the MSRP training set from scratch yields an accuracy of 74.46% [58], whereas transferring the weights from the model pre-trained on the data created using Algorithm 12.1 gives an accuracy of 76.05%. We are also doing better than Tree LSTM [227] (73.50% accuracy) which uses additional parse information and ConvNet Encoder [263] (73.96%) which uses a complex and expensive convolution operation over multiple channels.

## 12.4 Conclusion

In this chapter, we develop a multilingual semantic textual similarity extraction (STS) model leveraging multilingual word representations. We also propose an algorithm using LDA and OpenAI-GPT for generating realistic negative examples where the generated sentence talks about the same topic but in a different context. Our algorithm generates very good negative examples in the presence of only positive examples. We show that given good multilingual word representations, the language dependency barrier can be easily removed without harming the performance on multilingual semantic textual similarity task. Our experiments prove that having adequate data in one language pair certainly helps in other language pairs when performing a downstream task.

## Chapter 13

# Multilingual Corpus Creation for Multilingual Semantic Similarity Task

This chapter is based on the paper titled “Multilingual Corpus Creation for Multilingual Semantic Similarity Task” co-authored with Chahna Dixit, Robert E. Mercer, Atif Khan, Muhammad Rifayat Samee, and Felipe Urrea that appeared in Proceedings of the 12th Language Resources and Evaluation Conference (LREC 2020) [4].

In natural language processing, the performance of a semantic similarity task relies heavily on the availability of a large corpus. Various monolingual corpora are available (mainly English); but multilingual resources are very limited. In this work, we describe a semi-automated framework to create a multilingual corpus which can be used for the multilingual semantic similarity task. The similar sentence pairs are obtained by crawling bilingual websites, whereas the dissimilar sentence pairs are selected by applying topic modeling and an Open-AI GPT model on the similar sentence pairs. We focus on websites in the government, insurance, and banking domains to collect English-French and English-Spanish sentence pairs; however, this corpus creation approach can be applied to any other industry vertical provided that a bilingual website exists. We also show experimental results for multilingual semantic similarity to verify the quality of the corpus and demonstrate its usage.

### 13.1 Introduction

Semantic similarity, one of the important natural language processing (NLP) tasks, aims to measure the distance between two given content pieces in terms of their meaning. Traditionally, WordNet-based similarity measures such as Lin, Resnik, Jiang and Conrath [40] as well as statistical approaches including Latent Semantic Analysis (LSA) [124] and Pointwise Mutual

Information (PMI) [265] have been used to solve this problem. Recently with the advent of deep learning, the use of deep neural networks has gained popularity in solving this task; for example Siamese recurrent networks [164] and convolutional neural networks [202]. However, a major factor affecting the success of deep networks is the availability of substantially large and good quality corpora [66,110].

The most popular benchmark dataset for semantic similarity is the Semantic Textual Similarity (STS) dataset from SemEval tasks. The latest STS17 dataset [44] includes monolingual as well as cross-lingual sentence pairs for English, Arabic and Spanish languages. Nonetheless, the STS corpus requires a classification score ranging from 0 to 5 measuring the degree of similarity between the sentence pairs. We approach multilingual semantic similarity as a binary classification problem which has required us to collect a large corpus of our own based on the domain and language requirements of our application.

The collection of an entirely new and large corpus in itself is a challenging task; more specifically, textual data for NLP problems require human expertise and domain knowledge of the application. Above all, the acquisition of a multilingual corpus also demands some amount of linguistic knowledge. This leads to an increasing interest in developing an automated or semi-automated approach for building a multilingual corpus. Several corpus creation approaches have been published focusing on multiple languages and application domains. [173] proposed a web crawler to acquire parallel language resources for European languages. [207] developed a parallel corpus of scientific articles in English, Portuguese and Spanish languages by first acquiring documents from the Scielo database [171] and then aligning sentences from document pairs of different languages. Few other approaches exist, but in all of these works the generated corpus is meant to be utilized for machine translation. Apart from this, existing techniques focus on curating similar sentence pairs, but rarely talk about dissimilar sentence pair generation.

Bilingual sentence alignment lies at the heart of collecting similar pairs for a multilingual corpus. Maligna, a bilingual sentence alignment tool [99] does this by using statistical machine translation and a few sentence alignment algorithms to align sentences from document pairs. The tool is mainly used to align text for a machine translation dataset. While the corpus for machine translation requires perfect alignment among the sentence pairs, this is not true for the semantic similarity task since we are not looking for an exact translation of a sentence with another.

Considering all of the aspects discussed above for multilingual corpus creation specific to the semantic similarity problem, in this work, we describe a semi-automated approach to build a large corpus of English-French and English-Spanish sentence pairs that can be used for the multilingual semantic similarity task. The approach is based on scraping documents from

Sentence pairs	Label
We will get back to you by next week	Positive
We will contact you soon	
We will get back to you by next week	Positive
Nous vous contacterons la semaine prochaine	
You must pay your taxes	Negative
Ontario has high tax rate	

Table 13.1: Positive and negative sentence pair examples

bilingual websites and aligning the document pairs at the sentence and/or paragraph level. We have considered websites from government, insurance, and banking domains; but the advantage of this approach is that it can be applied for any language and domain or industry that has a website with bilingual content. We also plan to open-source the collected multilingual corpus for use by other researchers.

## 13.2 Architecture

Multilingual semantic similarity as a binary classification task requires a dataset consisting of bilingual sentence pairs labelled as either semantically similar or dissimilar. For simplicity, we will term the similar sentence pairs as positive samples and dissimilar pairs as negative samples. In this section, we provide detailed information on collecting the positive and negative samples for the corpus. The positive sample selection is a semi-automated approach that involves crawling multiple websites followed by bilingual sentence alignment along with an additional filtering process. It is to be noted that, these positive sample pairs are obtained from the same webpage of two different languages. On the other hand, the hypothesis for negative samples is that the sentence pairs should have similar topics determined by some automatic means but talk about a different aspect of this topic. Hence, the negative sentence pairs are formed by sampling from different webpages having a similar topic. To do this, we utilize a well-known topic modelling algorithm, LDA [37] and a sentence representation model, OpenAI GPT [188] on top of the positive samples. Table 13.1 shows some examples of positive and negative sentence pairs.

### 13.2.1 Positive Sample Selection

The positive sample selection approach consists of four main steps: data crawling, HTML parsing, text translation and text alignment. Each of these steps is explained in detail in the following subsections.

### 13.2.1.1 Data Crawling

In the first step we give the base URL of a bilingual (or multilingual) website of interest as input to a web crawler built using a Python library called *Scrapy* [116]. *Scrapy* is a fast high-level framework that crawls websites to extract structured data. The web crawler finds all the URLs from a given webpage URL and crawls each of those URLs recursively to extract the data. This process goes on in an iterative fashion where the input to a particular iteration is the list of URLs obtained from the previous iteration. We run the crawler for as long as no new webpages are being crawled.

For a given webpage URL, the key point is finding the corresponding parallel webpage URL in the counterpart language. This can be searched by finding a pattern in the HTML code of several webpages of that website. The crawler outputs several HTML files where each HTML file corresponds to a single webpage. In the end, these HTML files undergo post-processing to delete the webpages that do not have a parallel webpage in the counterpart language.

We denote the parallel sets of HTML files as  $H_{l_1}$  (HTML files for language  $l_1$ ) and  $H_{l_2}$  (parallel HTML files for counterpart language  $l_2$ ), where  $H_{l_1}$  and  $H_{l_2}$  have an equal number of files after post-processing.

### 13.2.1.2 HTML Parsing

We use the Python library *inscriptis* [241] to extract all of the text content from the HTML files. The extracted text is then split into lines where each line can be a word, a sentence or a paragraph. We then discard those lines that do not contain at least one alphabetic character as well as the lines containing just one word. Next, a text file is generated for each HTML file which contains the parsed and clean raw text from that HTML. In order to build the corpus, we retain only those pairs of parallel text files that have equal numbers of lines. The reason for this step is that our text alignment approach is based on the line order of the file. More details on the alignment process can be found in Subsection 13.2.1.4. We term  $T_{l_1}$  as the set of text files corresponding to  $H_{l_1}$  and  $T_{l_2}$  as the set of text files from  $H_{l_2}$ , where  $T_{l_1}$  and  $T_{l_2}$  have an equal number of files but may not be the same as the number of files in  $H_{l_1}$  and  $H_{l_2}$  due to the refinement process.

The HTML tags in the parallel HTML files can also be leveraged to extract more text content and remove additional noise of headers, footers, titles, etc. from the webpage. We experimented with extracting text content based on the *class* attribute of parallel `<div>` tags which helped to retain more files when applying post-processing based on the length of text files.

### 13.2.1.3 Text Translation

Our text alignment approach works on parallel text files in the same language. So, any one set from the parallel set of text files must be translated to another language. We translate all

of the text files in any other language into English using the Python library `mtranslate` [17] which implements the Google Translate API. Now, if  $l_1$  represents the English language and  $l_2$  represents any language other than English, then each text file  $t_{l_1}^k$  in  $T_{l_1}$  will correspond to two parallel files –  $t_{l_2}^k$  from  $T_{l_2}$  in the counterpart language and  $t_{l_2'}^k$  from  $T_{l_2'}$  which consists of text files from  $T_{l_2}$  translated into English. Here,  $k$  represents the  $k^{th}$  file in the set of text files.

#### 13.2.1.4 Text Alignment

The alignment of text is the most important step in the framework of selecting positive samples. The approach is based on the hypothesis that the contents of two parallel bilingual webpages appear in somewhat the same order. So most of the parallel text files should be aligned; however, there will be exceptions in some cases. Hence, we devise the text alignment approach in such a way that the alignment check for a particular pair of text files is line-based and one-to-one. This means that a line at a given position in  $t_{l_1}^k$  is checked against a single line at the same position in  $t_{l_2'}^k$ .

We use word frequency-based cosine distance as a distance measure between each line in the files  $t_{l_1}^k$  and  $t_{l_2'}^k$ . This basic measure seems to work well in aligning semantically similar content pairs. The positions (or indices) of line pairs with cosine distance greater than 0.6 are recorded as being misaligned. The set of indices for misaligned lines  $I^k$  is refined further such that if a particular index in the set does not have a consecutive misaligned line, then that index is discarded from  $I^k$ . The intuition behind this step is that the translation may have affected the cosine distance to record it as misaligned.

Finally, the files having empty  $I^k$  are considered to be aligned and the remaining files are aligned manually based on the indices in  $I^k$ . The manual alignment of files involves rearranging certain lines or discarding lines that do not have a match in the corresponding parallel file. The manual alignment is only done for the files that have the number of misaligned lines under a certain threshold. The aligned set of parallel text files containing semantically similar positive sample pairs are denoted as  $P_{l_1}$  and  $P_{l_2}$ . Here,  $p_{l_1}^k \in P_{l_1}$  and  $p_{l_2}^k \in P_{l_2}$  are the  $k^{th}$  parallel files obtained after alignment of files  $t_{l_1}^k$  and  $t_{l_2'}^k$ . Each corresponding line pair from the parallel files is considered as a positive sample pair.

### 13.2.2 Negative Sample Selection

In this subsection we explain our negative sample selection approach which is applied over each language pair and domain individually. For the three domains i.e., government, insurance, and banking, we divide our aligned parallel files into three sets: English-French Government **EN-FR-G**, English-French Insurance Banking **EN-FR-IB**, and English-Spanish Insurance Banking **EN-ES-IB**. A detailed description of these partitions is given in Section [13.3]



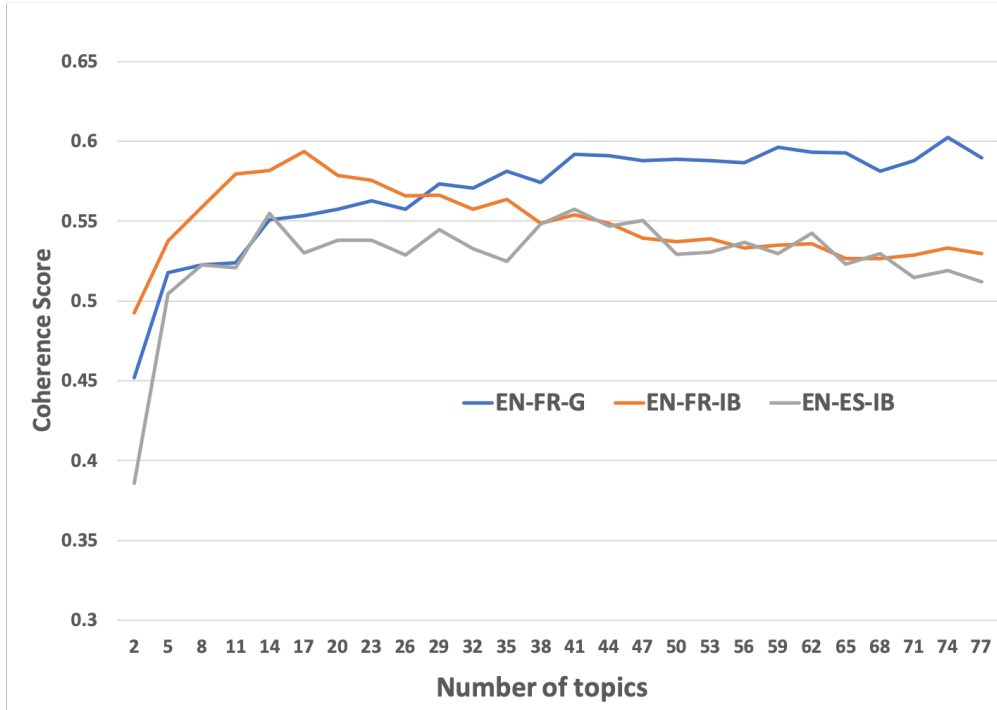


Figure 13.1: Topic coherence score vs number of topics.

Our negative sample selection approach starts by training a range of unsupervised LDA [37] models on  $P_{l1}$  where  $l1$  is constrained to be the English language. Each file  $p_{l1}^k$  in  $P_{l1}$  is considered as a single document  $D$ . The LDA model with the maximum coherence score is chosen as the best topic model. Figure 13.1 shows the coherence score vs. number of topics plot for **EN-FR-G**, **EN-FR-IB** and **EN-ES-IB** where the optimal number of topics are 74, 17 and 41, respectively. We use the following parameters for training LDA: *random\_state*=100, *update\_every*=1, *chunksize*=100, *passes*=300, *alpha*=auto, *per\_word\_topics*=True.

Using the best LDA model, we represent each English document as a document vector which is a probability distribution over all the topics. Then, for an input sentence  $S$ , its dominant topic  $T$  is obtained according to this topic distribution. Next, we use a pretrained OpenAI-GPT model [188] to get the vector representation  $M(s)$ , where  $s$  represents sentences from all of the English documents. We then extract a set of documents  $A$  having the same topic  $T$ , and collect the sentences (and their multilingual counterparts) having cosine similarity with input sentence  $S$  in the range 0.8-0.9. The cosine similarity between the two sentences is calculated from the vector representations  $M(.)$  of those sentences. Based on our definition of the negative samples, we choose the similarity threshold range to be 0.8-0.9; which gives us samples that belong to a similar topic. However, this threshold range can be adjusted based on the application requirements.

Following the above mentioned steps, we select  $n$  sentences for each  $S$  and then create  $n$

**Algorithm 13.1** Negative sample selection with LDA-LM

---

```

Pretrained LDA model:  $L$ 
Pretrained OpenAI-GPT model:  $M$ 
Input English document:  $D$ 
Topic of  $D$  according to  $L$ :  $T$ 
Set of negative samples:  $N$ 
List of documents with same topic as  $T$ :  $A$ 
Number of sentences to be selected:  $n$ 
Input sentence from document  $D$ :  $S$ 
 $N \leftarrow \emptyset$ 
for  $i < n$  do
   $x \leftarrow \text{NULL}$ 
  for document  $d \in A$  do
    for sentence  $s \in \text{document } d$  do
      if  $0.80 < \text{Cosine}(M(S), M(s)) < 0.90$  then
         $x \leftarrow s$ 
        break
      end if
    end for
    if  $x \neq \text{NULL}$  then
      break
    end if
  end for
   $N \leftarrow \text{multilingual counterpart of } x$ 
   $i \leftarrow i + 1$ 
end for

```

---

negative sentence pairs in the multilingual space by pairing  $S$  with the appropriate multilingual counterparts of each of these sentences. Algorithm 13.1 makes precise the above steps. For our experiments we choose the value of  $n$  to be 10. This yields a sufficient number of negative samples for the corpus. However, not all the samples are used to build the corpus. In the end, the negative sentence pairs are sampled in order to create a balanced dataset with respect to the total number of positive sentence pairs.

## 13.3 Corpus Details

We scraped 11,156 bilingual webpages pairs in total, out of which approximately 9.25% were discarded based on the filtering and alignment process described in Subsections 13.2.1.2 and 13.2.1.4. Hence, 10,124 text file pairs were used to create the positive and negative sentence pairs.

The final corpus consists of 351,334 English-French (EN-FR) sentence pairs and 53,826 English-Spanish (EN-ES) sentence pairs summing up to a total of 405,160 sentence pairs with

Language	Sentence 1	Sentence 2	Label
EN-FR	There are minimum and maximum permissible withdrawals from the plan each year.	Les retraits sont soumis à des minimums et à des maximums annuels admissibles.	Positive
	At the eye of a hurricane there is a clam area of blue sky.	Dans l'oeil d'un ouragan, il y a une zone calme de beau temps.	Positive
	Guide T4002, Business and Professional Income	Formulaire T4A, État du revenu de pension, de retraite, de rente ou d'autres sources	Negative
	What can be deducted from an employee's pay cheque?	Quand l'employeur doit-il verser l'indemnité de congé annuel?	Negative
EN-ES	To focus on the love and fun a pet can bring, instead of the extra cost, all pet parents should consider purchasing pet insurance from a reputable, caring company.	Con el fin de enfocarse en el amor y la alegría que una mascota puede ofrecer, y no en los costos adicionales, todos los "papás" de mascotas deberían pensar en comprar un seguro de mascotas de una compañía de reputación que se preocupe por sus clientes.	Positive
	Don't stress if you lose track of your phone—all mobile wallet transactions require the verification you set up, like a fingerprint scan.	No se estrese si pierde su teléfono, todas las transacciones de la billetera móvil requieren la verificación que usted haya establecido, como una huella digital.	Positive
	Use these tips to get your bike in top shape for the new riding season.	Si los carros deportivos son una pérdida total o son robados, normalmente cuesta más reemplazarlos.	Negative
	All coverages are subject to the terms, provisions, exclusions, and conditions in the policy itself and in any endorsements.	Ésta es sólo una descripción general de las coberturas de los tipos de seguros disponibles y no representa una declaración de contrato.	Negative

Table 13.2: Examples from collected multilingual corpus

202,580 sentence pairs for each class – positive and negative. Some examples from the collected corpus are shown in Table 13.2

As mentioned before, we created our multilingual corpus by scraping from 5 different bilingual websites. Three had content in English and French, while the remaining two were in English and Spanish. The topics of these website contents were government, insurance, and banking. In order to do an extensive evaluation, we divided the entire corpus based on language pairs and domain. The government domain was only available for English-French pair, so we created one dataset – **EN-FR-G**. The insurance and banking domains were available for

Dataset	Train	Validation	Test
<b>EN-FR-G</b>	195,303	48,826	61,033
<b>EN-FR-IB</b>	29,546	7,389	9,237
<b>EN-ES-IB</b>	34,447	8,613	10,766

Table 13.3: Sentence pair counts for dataset partitions

both English-French and English-Spanish pairs. So we created two more datasets using these – **EN-FR-IB** and **EN-ES-IB**.

For each of the three datasets, we used the positive and negative sample pairs, described earlier, to create a balanced 10-fold training and validation partition for doing 10-fold cross validation experiments. We also created a test set for testing.

The dataset partition details are given in Table 13.3. In order to verify the quality of our corpus we have evaluated our model on a benchmark dataset that has been supplemented with our corpus. We chose the well known Microsoft Research Paraphrase Corpus (MSRP) where the task is to do paraphrase identification [69]. Because of the way we prepared our corpus, it aligns well with this kind of task. The original MSRP dataset has 5,801 sentence pairs, 4,076 in the training set and 1,725 in the test set. Adding our corpus to the MSRP training set shows an increase in performance on the MSRP test set. We hypothesize that this indicates that our corpus is of good quality.

## 13.4 Evaluation Experiments

In this section, we present a thorough analysis of all the evaluation experiments that we did to validate our corpus. We first describe the model architecture which we used to solve the multilingual semantic similarity task. Following this, we explain the training details of the model along with its hyper-parameter settings. We also present the detailed results obtained with our experiments and compare the transfer performance of our selected model with some of the top performing models on the MSRP dataset.

### 13.4.1 Model Architecture, Parameters and Training Details

We have chosen to use InferSent [58], an LSTM based model, to compute the representations of a pair of sentences,  $a$  and  $b$ , and then compare the representations for an underlying task. The model first traverses each sentence as a sequence of  $T$  words  $\{x_t\}_{t=1,\dots,T}$  from both left to right and right to left and generates two hidden representations at each time step  $\vec{h}_t, \overleftarrow{h}_t \forall t \in [1, \dots, T]$ . During input, it considers the vector representation of each word ( $x_t$ ) in the sentence

from a pre-trained word embedding model.

$$\begin{aligned}\vec{h}_t &= \overrightarrow{\text{LSTM}}_t(x_1, \dots, x_T) \\ \overleftarrow{h}_t &= \overleftarrow{\text{LSTM}}_t(x_1, \dots, x_T) \\ h_t &= [\vec{h}_t, \overleftarrow{h}_t]\end{aligned}\tag{13.1}$$

After this, the model employs a max (or mean) pooling block to summarize the hidden states in one dense representation.

$$h = \text{maxpool}(h_1, \dots, h_T)\tag{13.2}$$

The next steps are to infer the similarity between the two representations ( $h_a, h_b$ ) using standard matching methods and to project the resultant vector into the space of classes  $y$  (which is two in our case) through a series of fully connected layers as follows

$$x = (h_a, h_b, |h_a - h_b|, h_a * h_b)\tag{13.3}$$

$$P(y|\mathbf{X}) = \sigma(\mathbf{W}_1 \sigma(\mathbf{W}_2 x + \mathbf{b}_2) + \mathbf{b}_1)\tag{13.4}$$

Finally, the model is trained by optimizing a task specific loss function as follows

$$H(p, q) = - \sum_{i=1}^n Q(y_i) \log(P(y_i))\tag{13.5}$$

The LSTM hidden state dimension is set to 600. Multilingual word vectors are initialized with the 300 dimension MUSE embeddings [60] and are not updated during training. To smooth the update, the gradients are divided by  $B^2$  where  $B$  is the batch size which is set to 512. The learning rate is reduced by a factor of 2 if  $\sqrt{\sum_{i=1}^k \|\nabla \theta_i^2\|}$  is more than a threshold, which is 5 for our experiments. We use Adam as the optimization algorithm and the dropout in the classification layer is set to 0.5. The number of topics parameter is described in Subsection 13.2.2.

## 13.4.2 Results and Analysis

We train three different models with different combinations of training data depending on the language pairs and domain. The models **EN-FR** and **EN-ES** use **EN-FR-G** and **EN-ES-IB** datasets respectively, whereas the **EN-FR-ES** model uses all three datasets. Table 13.4 shows the 10-fold cross validation performance of all of these models. We summarize the performances over all the folds in three different ways. Firstly, we report the mean accuracies along with the standard deviation over all the folds for all three models. Following this, we report

Model	Validation set Accuracy		
	Mean	Max voting	Avg. voting
<b>EN-FR</b>	95.41 $\pm$ 0.39	95.76	95.97
<b>EN-ES</b>	96.35 $\pm$ 0.57	97.07	97.22
<b>EN-FR-ES</b>	95.03 $\pm$ 0.24	95.40	95.59

Table 13.4: 10-fold cross validation performance of different models (mean includes standard deviation)

Model	Test set accuracy		
	en-fr	en-es	en-en (MSRP)
<b>EN-FR</b>	95.64	95.91	76.05
<b>EN-ES</b>	87.15	97.25	75.07
<b>EN-FR-ES</b>	94.91	98.34	76.00

Table 13.5: Cross corpus performance (Accuracy). Rows indicate training language pairs, and columns indicate testing language pairs

the results of the ensemble experiment where we use max voting and average voting as our ensemble methods. It can be seen that the average voting achieves the better performance among all of these methods getting 95.97%, 97.22% and 95.95% accuracy for **EN-FR**, **EN-ES** and **EN-FR-ES**, respectively.

Table 13.5 reports the cross corpus performance of the three models (models are in upper-case and datasets are in lowercase). We have not included **en-fr-es** for test purposes because samples in this dataset are taken from the English-French (**en-fr**) and English-Spanish (**en-es**) pairs. The performance scores are reported over the test set that we create for each of these datasets as shown in Table 13.3. It is to be noted that the models **EN-FR**, **EN-ES** and **EN-FR-ES** are trained on the English-French (**en-fr**), English-Spanish (**en-es**) and English-French-Spanish (**en-fr-es**) datasets, respectively. We have chosen to use the best model on each of these datasets out of the 10 models that we create during the 10-fold cross validation. It can be seen that the **EN-FR** model shows very good performance over **en-es** (95.91%) and it is doing even better than its own test set (95.64%). When tested on **en-fr**, the performance of the **EN-ES** model drops with respect to **en-es** being the test set; the relatively smaller size of training data for **EN-ES** as compared to **EN-FR** can be one of the reasons for this performance drop. When trained on **en-fr-es** the performance of **EN-FR-ES** on the two language pairs compare well. We also report the performance of the models when tested on MSRP which is a **en-en** corpus. We believe that this good cross corpus performance is because the word embeddings are aligned in the same semantic space.

Table 13.6 reports the performance of the model on the MSRP task compared to some of the existing top performing models. As we can see, InferSent trained on the MSRP training

Model	Accuracy
InferSent [58]	74.46
LSTM [58]	70.74
BiGRU Last Encoder [58]	70.46
Tree LSTM [227]	73.50
ConvNet Encoder [263]	73.96
Ours (Transfer + Finetuning)	<b>76.05</b>

Table 13.6: Performance comparison on the MSRP dataset against some existing top performing models.

Dataset	Sentence 1	Sentence 2	GT	Pr
EN-FR	The Cannabis Act proposes many rules that would protect youth from accessing cannabis.	Le projet de loi sur le cannabis prévoit de nombreuses dispositions pour empêcher les jeunes d’avoir accès au cannabis.	1	1
	The authorized health care practitioner’s licence information	Numéro de téléphone et adresse électronique de la personne morale	0	0
	What can be deducted from an employee’s pay cheque?	Quand l’employeur doit-il verser l’indemnité de congé annuel?	0	0
EN-ES	Use window sheet kits	Usa kits de aislamiento para ventanas	1	1
	It will only take a minute and won’t impact your credit score	Díganos quién es y qué le gusta, para ver qué ofertas están	0	0
	List out your debt	Fíjate un presupuesto semanal, empezando el lunes	0	0

Table 13.7: Example predictions from the test set. **GT**: ground truth, **Pr**: predicted.

set from scratch yields an accuracy of 74.46% [58], whereas transferring the weights from the model pretrained on our dataset gives an accuracy of 76.05%. It is to be noted that we are also doing better than Tree LSTM [227] (73.50% accuracy) which uses additional parse information and ConvNet Encoder (73.96%) which uses a complex and expensive convolution operation over multiple channels.

Table 13.7 shows the models’ predictions on a few examples from our dataset. It can be seen in **EN-FR** that the two negative sentences talk about the same topics as their counterpart English sentences, but the contents differ. In **EN-ES**’s second pair, the English sentence talks about a credit score while the Spanish sentence talks about some offers which are somehow related. Here in the third pair, the English sentence talks about debt whereas the Spanish sentence talks about budget, which are not exactly related but somehow gets used in the same context. This justifies our hypothesis of choosing topic related negative examples.

Our discussion on the experimental results shows that the semantic similarity models trained using the collected multilingual corpus perform well across different languages and

domains. It is important to understand that like any other curated dataset, this corpus may have some amount of noise in terms of alignment. However, the results of transfer learning on the MSRP benchmark dataset verifies the quality of the dataset.

## 13.5 Conclusion

In this chapter, we develop a multilingual corpus for doing the multilingual semantic similarity task. We investigate this similarity problem as a binary classification task. To obtain the positive examples, we adopt web crawling of bilingual sentence pairs followed by a set of careful preprocessing steps to align them. We focus on websites in the government, insurance, and banking domain to collect English-French and English-Spanish sentence pairs. To create the bilingual sentence pairs of the negative class, we propose an algorithm utilizing LDA and OpenAI-GPT. Using this algorithm, we can create synthetic non-similar bilingual sentence pairs, where the participating entities talk about the same topic with some differing content. Our corpus creation approach can be applied to any other industry vertical provided that a bilingual website exists. To evaluate the quality of the corpus, we create a pre-trained multilingual version of InferSent and show that we obtain better transfer learning performance over a well known public dataset – MSRP.



# Chapter 14

## Conclusions

This thesis presents some state-of-the-art deep learning architectures to address a range of classical and complex natural language processing (NLP) problems. We present each of these works as a chapter and at the same time have made them self-contained. All of the works have presented a thorough investigation, have good theoretical foundations, have provided extensive experimentation, and have been published in some of the top tier conferences. Readers are presented with good insight in how to look at a natural language problem, think intuitively, and gradually build a model that addresses that problem. Throughout this thesis we have been consistent in blending the cutting edge deep learning models with classical natural language knowledge. In this chapter we summarize all of our key findings along with our major contributions. We compare our models with other like models and show which model is best suited on which type of task through some empirical results and provide an explanation. We also present the limitations of this thesis and will complete the discussion by giving future directions for this research and what are the possible areas for improvement.

### 14.1 Key Findings

This study provides a wide research scope to the reader. We explore a range of NLP problems and provide models to address them. There are five overlapping problems that we use to evaluate the effectiveness of most of the deep learning models that we design. Four of them deal with a pair of sentences (i.e. SICK-E, SICK-R, MSRP and AI2-8grade), so we term them sentence pair modelling tasks. The remaining one (i.e. SST) deals with sentiment classification tasks. A quick summary of the performances of our models on these problems is given in Table [14.1](#). On the SICK-E task, sequential models perform much better compared to the tree structured models and the Infersent + RL ensemble was the best among all. However, on the SICK-R task, the tree structured models are much superior compared to the sequential models.

Model	SICK-E	SICK-R	SST	MSRP	AI2-8grade
Dependency Tree Transformer	82.95	0.2774	83.12	70.34	-
Constituency Tree Transformer	82.72	0.3012	86.66	71.70	-
Dependency Tree Transformer + edge label	83.32	0.2627	83.75	71.96	-
Dependency Tree LSTM + attention	-	0.2518	-	-	-
Constituency Tree LSTM + attention	-	0.2435	-	-	-
Transformer Non Siamese	83.17	0.5042	-	75.90	75.29
Transformer Siamese	85.22	0.4969	-	76.80	77.29
LSTM <sub>RMC</sub> + FLMP	85.38	0.3852	-	74.67	74.72
LSTM <sub>RMC</sub> + VLMP (WINDOW SIZE = 5)	84.28	0.2925	-	75.89	74.72
Infersent + RL	84.57	-	-	74.74	73.84
Infersent + RL Ensemble	86.12	-	-	76.12	74.91
Infersent + RL version 2	84.43	-	-	75.07	74.74
Infersent + RL Ensemble version 2	85.10	-	-	75.35	74.78
Transfer Learning	-	-	-	76.05	-

Table 14.1: Summary of all of the model performances on the overlapping datasets.

On the SST task, the constituency tree transformer is best among all. This task has not been extensively explored in our study as we are mainly interested in the sentence pair modelling task. On the paraphrase identification task, the Siamese Transformer, which is from the family of sequential models, is doing better than all of the other models. Finally, on the AI2-8grade dataset, again the Siamese Transformer stands out, giving the best performance. Looking at the overall scenario, we can conclude that sequential models perform much better compared to the tree structured models in sentence pair modelling tasks.

Apart from that, we also explored some classical NLP problems (i.e., part of speech tagging, named entity recognition and chunking) as a sequence tagging problem where a plain LSTM based tagger is getting a state-of-the-art result. We revisit the word sense disambiguation problem as a sequence to sequence learning problem and show that it can also be a potential solution to this problem. Finally, for the protein-protein interaction problem, we find that the tree version models are superior to the sequential models.

## 14.2 Major Contributions

This study explores a wide variety of NLP problems and tries to solve them using some novel deep learning models with intuitive architectures. We try to give a clear explanation about the type of modules used in these architectures and why they were used, to make things more intuitive and interpretable. Apart from this intuitive design, we also achieved state-of-the-art performance on some tasks at the time those models were published. We now summarize all of the major contributions of this study as follows:

- Introduce a multitask model that addresses a set of classical NLP tasks using only one training strategy. The tasks are part of speech tagging, named entity recognition, and chunking.
- Address another classical NLP problem named word sense disambiguation by utilizing a well-known NLP paradigm called sequence to sequence learning.
- Provide a way of encoding the attention module inside a tree structured long short term memory cell.
- Introduce a cross attention mechanism when designing a sentence pair modelling task.
- Increase the memory depth of relational recurrent neural network through a variable length memory pointer.
- Design a tree version of the transformer encoder that can encode an entire grammar tree.
- Use the reinforcement learning algorithm to compress a sentence using the label as a reward marker.
- Utilize the reinforcement learning algorithm to generate a phrase representation of a sentence without having any specific knowledge source.
- Provide a way of creating negative samples when only positive samples are available.
- Provide a multilingual sentence comparison model compatible with English, French, and Spanish languages.

### 14.3 Limitations of the Study

In this thesis, we carefully look at all aspects of an architecture, analyze the pros and cons of every linguistic feature that were added, and achieve consistently good results all the time. However, there still are a few limitations which need to be considered.

This study considers only one level of hierarchy, word level to sentence level. However, there are more levels, i.e., phrase level and document level. The document level hierarchy is out of the scope of this thesis as the datasets that we explored all deal with sentences. However, we believe that we failed to encode phrase level information in most of the tasks except for SST which is already tagged at the phrase level and the task is to classify the sentiment of each phrase. It is possible to rethink some of the models with additional modules that can capture the phrase structure of a text piece.

Another constraint that we want to mention here is the hardware limitation. In a few models, we saw a pattern that the performance gradually improves with increasing batch size. But for each task, we were limited to a maximum batch size because of the GPU memory limitation. This problem is not associated just with the batch size, we faced some other design constraints as well, i.e., number of layers, dimension of each layer, etc. For example, in Chapters 1 and 2, we only use bigrams, but it is possible to use trigrams, 4-grams, and 5-grams as well. However, in the later chapters we overcome some of these problems by training our models on Compute Canada clusters.

Most of the models that are designed specifically for SICK-E, SICK-R, MSRP and AI2-8grade tasks take around 4 to 8 minutes per epoch and a maximum 50 epochs to converge. Hyperparameter tuning on these models are performed using grid search and random search. However, when we perform the tasks using the reinforcement learning paradigm, it takes around 15 to 20 hours for training. And because of the hardware limitations as mentioned above, we were unable to perform a grid search for hyperparameter tuning for the reinforcement learning models. We only tuned the models on a fixed set of hyperparameters individually.

## 14.4 Recommendations for Future Research

In some sections of Chapters 2 to 13 we suggest some work that needs to be done as a future research related to that topic. Some of this recommended research was accomplished in later chapters as the thesis progressed. In addition to that, there are some suggestions that are yet to be explored and can be seen as possibilities to be explored in future research.

In Chapter 2, we looked at suffix features of a word. There are, in total, 137 suffixes available in the resource referenced in Chapter 2; however, we only looked at the top 10 most frequent suffixes that appeared in our dataset. There is room to explore all of the suffix features and see whether this improves the results.

In Chapter 3, we mention that the idea can be easily extended to address a multi-fact question answering problem. However, we later shift our focus towards the semantic similarity genre and didn't follow through on this idea due to time limitations.

In Chapter 5, we use the standard Tree LSTM cell followed by structured attention instead of using the attentive version that we design in Chapter 4. It would be interesting to replace this stacked model with the attentive version. Also, as we were looking at the tree version of a sentence with the tree version of LSTM, we hypothesize that we can perform breadth first traversal or depth first traversal of the grammar tree and then use the standard LSTM cell with or without the structured attention module instead of the tree LSTM version. This idea is yet to be explored.

In Chapter 9, we mention that there are different ways to encode dependency edge information and we use one that seems more intuitive to us. When making this design choice, we had to limit ourselves to not using too many parameters because of hardware limitations. But given that GPU memory size is increasing, it is possible to explore more design options, such as different combinations of transformations, concatenations, and residual connections to encode these dependency edge labels.

In Chapter 11, we explored that our reinforcement learning framework works well with shorter sentences and can give reasonable phrase structures. And in Chapter 10, we were compressing a sentence by deleting some irrelevant words. One possible solution to the problem that we faced in Chapter 11 is to first shorten a sentence by deleting irrelevant words using the model from Chapter 10 and then further work on that. In addition, it would be interesting to see how the models found in other parts of the thesis would react to the compressed sentences that are obtainable from the model designed in Chapter 10.

In Chapters 12 and 13, we fine tune only on the MSRP dataset. However, it is possible to explore other similar kinds of datasets that align with the language structure of the corpus that we created and fine tune on them. Also, we use the standard LDA model to extract the topics; however, there are some other alternatives such as latent semantic indexing, independent component analysis, probabilistic latent semantic indexing, non-negative matrix factorization, and Gamma-Poisson distribution that are yet to be explored.

Based on the experimental results, it is evident that the sequential models get superior performance compared to the tree-structured models for most of the tasks. When attention modules are combined with sequential models, they are able to see the entire input context; however, attention modules with the tree-structured models only see a local region within a subtree. It would be interesting to follow up on some of the recent works which successfully encode global information with attention over trees [100, 169].

Because of the hardware limitations as mentioned above, we were unable to perform extensive hyperparameter tuning over all of the models. As deep learning and reinforcement learning models are very sensitive to hyperparameters, it would be interesting to revisit our models with extensive hyperparameter tuning based on some well known hyperparameter search approaches such as random search [35], Spearmint [74] and Hyperopt [36].

We design a number of models that can give effective sentence representation. One possible research direction would be to apply these representations to analyze medical documents and retrieve contents related to medicines that can help health workers in quickly finding scientifically based answers and avoiding misinformation during a time of crisis. Another possible research direction would be to analyze the conversation on social media regarding a public health sensitive topic such as COVID-19 and supply different responses to topics like anger,

myths, country-wise health scenario, etc.

## 14.5 A Final Word

In this study, we design a set of deep learning models that incorporate linguistic knowledge and address a wide variety of closely related NLP tasks. As we have seen, most of these complex architecture designs have a common motivation – to generate effective sentence representations. Having an effective domain specific sentence representations makes solving a wide variety of tasks on that domain easier. There is a coherency in the kind of research and the order in which we did it which helped us build a strong foundation and prepared us to address more complex and advanced tasks as presented in the future work section. Through our extensive experiments it became clear that additional linguistic information does help the modern deep learning architecture in gaining more insight about the data. Another set of our works prove that a simple tree-based model on many occasions outperforms a complex sequential model with a large number of modules. Furthermore, we also show that it is possible to design such a model that utilizes traditional linguistic information as well as modern deep learning modules. One of our intuitions was that in generating a sentence representation, not all the words contribute with the same impact, some contribute more, some contribute less and some not at all. We verified this hypothesis using two of our existing works: the RL-based work gives a discrete distribution of which words to keep and which words to delete. On the other hand, the Transformer-based work gives a continuous distribution over the contribution of the words. In order to be more specific, in one of our other works, we design a relational memory based architecture where we see that this continuous distribution behavior over the contribution actually works better within a window rather than over the entire set of words in the sentence. We also hypothesize that if we have good word representations across multiple languages, where the words having the same meaning are aligned, then the sentence vectors that we get across multiple languages will also become language agnostic.

In summary, all of the works that we have done in this thesis follow a close coherency and are mostly related with each other. The kind of goals that we have achieved are also based on proper human-like intuition that we want the future AI should have. So far, we have seen that these modern data driven deep learning architectures are superior in almost all cases, but we also prove that adding human-like knowledge (i.e., linguistic information and intuitive design principle) brings another level of expertise into these modern gems.

# Bibliography

- [1] Eneko Agirre and Philip Edmonds. *Word sense disambiguation: Algorithms and applications*, volume 33. Springer Science & Business Media, 2007.
- [2] Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84, 2014.
- [3] Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. Weighted transformer network for machine translation. *arXiv preprint arXiv:1711.02132*, 2017.
- [4] Mahtab Ahmed, Chahna Dixit, Robert E Mercer, Atif Khan, Muhammad Rifayat Samee, and Felipe Urrea. Multilingual corpus creation for multilingual semantic similarity task. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 4190–4196, 2020.
- [5] Mahtab Ahmed, Chahna Dixit, Robert E. Mercer, Atif Khan, Muhammad Rifayat Samee, and Felipe Urrea. Multilingual semantic textual similarity using multilingual word representations. In *Proceedings of the 2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, pages 194–198, 2020.
- [6] Mahtab Ahmed, Jumayel Islam, Muhammad Rifayat Samee, and Robert E. Mercer. Identifying protein-protein interaction using Tree-LSTM and structured attention. In *Proceedings of the 2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 224–231, 2019.
- [7] Mahtab Ahmed and Robert E. Mercer. Efficient transformer-based sentence encoding for sentence pair modelling. In Marie-Jean Meurs and Frank Rudzicz, editors, *Advances in Artificial Intelligence*, pages 146–159. Springer International Publishing, 2019.
- [8] Mahtab Ahmed and Robert E. Mercer. Investigating relational recurrent neural networks with variable length memory pointer. In *Proceedings of the 33rd Canadian Conference on Artificial Intelligence*, 2020.

- [9] Mahtab Ahmed and Robert E Mercer. Modelling sentence pairs via reinforcement learning: An actor-critic approach to learn the irrelevant words. In *Proceedings of The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)*, pages 7358–7366, 2020.
- [10] Mahtab Ahmed and Robert E. Mercer. Encoding dependency information inside tree transformer. In *Proceedings of the 34th Canadian Conference on Artificial Intelligence*, 2021.
- [11] Mahtab Ahmed, Muhammad Rifayat Samee, and Robert E. Mercer. Improving neural sequence labelling using additional linguistic information. In *Proceedings of the 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 650–657, 2018.
- [12] Mahtab Ahmed, Muhammad Rifayat Samee, and Robert E. Mercer. A novel neural sequence model with multiple attentions for word sense disambiguation. In *Proceedings of the 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 687–694, 2018.
- [13] Mahtab Ahmed, Muhammad Rifayat Samee, and Robert E Mercer. Improving tree-LSTM with tree attention. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 247–254, 2019.
- [14] Mahtab Ahmed, Muhammad Rifayat Samee, and Robert E Mercer. You only need attention to traverse trees. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 316–322, 2019.
- [15] Antti Airola, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC bioinformatics*, 9(11):S2, 2008.
- [16] Hanan Aldarmaki and Mona Diab. GWU NLP at SemEval-2016 shared task 1: Matrix factorization for crosslingual STS. In *Proc. of the 10th Int. Workshop on Semantic Evaluation (SemEval-2016)*, pages 663–667, 2016.
- [17] Arnaud Aliès. mtranslate. <https://github.com/mouuff/mtranslate>, 2016.
- [18] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.



- [19] Rie Kubota Ando and Tong Zhang. A high-performance semi-supervised learning method for text chunking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 1–9, 2005.
- [20] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *arXiv preprint arXiv:1601.03764*, 2016.
- [21] Jimmy Lei Ba et al. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [22] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [23] Alan Baddeley. *Working Memory, Thought, and Action*, volume 45 of *Oxford Psychology Series*. OUP Oxford, 2007.
- [24] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- [25] Satanjeev Banerjee and Ted Pedersen. An adapted Lesk algorithm for word sense disambiguation using WordNet. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 136–145, 2002.
- [26] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don’t count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, 2014.
- [27] Sergey Bartunov. Adagram. <https://github.com/sbos/AdaGram.jl>, 2017.
- [28] Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry Vetrov. Breaking sticks and ambiguities with adaptive skip-gram. In *Artificial Intelligence and Statistics*, pages 130–138, 2016.
- [29] Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. An enhanced Lesk word sense disambiguation algorithm through a distributional semantic model. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1591–1600, 2014.
- [30] Petr Baudiš, Silvestr Stanko, and Jan Šedivý. Joint learning of sentence embeddings for relevance and entailment. In *Proc. of the 1st Wksp. on Representation Learning for NLP*, pages 8–17, 2016.

- [31] Jon A Benediktsson, Philip H Swain, and Okan K Ersoy. Neural network approaches versus statistical methods in classification of multisource remote sensing data. *IEEE Transactions on Geoscience and Remote Sensing*, 28, 1990.
- [32] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.
- [33] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155, 2003.
- [34] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.
- [35] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.
- [36] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, 2015.
- [37] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.
- [38] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proc. of Computational Statistics 2010*, (COMPSTAT’2010), pages 177–186, 2010.
- [39] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.
- [40] Alexander Budanitsky and Graeme Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Comput. Linguist.*, 32(1):13–47, March 2006.
- [41] Francis Buekenhout and Monique Parker. The number of nets of the regular convex polytopes in dimension  $\leq 4$ . *Discrete Mathematics*, 186(1-3):69–94, 1998.
- [42] Razvan Bunescu, Ruifang Ge, Rohit J. Kate, Edward M. Marcotte, Raymond J. Mooney, Arun K. Ramani, and Yuk Wah Wong. Comparative experiments on learning information extractors for proteins and their interactions. *Artif. Intell. Med.* 33 (2), pages 139—155, 2005.

- [43] Andrei M Butnaru, Radu Tudor Ionescu, and Florentina Hristea. Shotgunwsd: An unsupervised algorithm for global word sense disambiguation inspired by dna sequencing. *arXiv preprint arXiv:1707.08084*, 2017.
- [44] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada, August 2017. Association for Computational Linguistics.
- [45] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiacob, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuanc, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.
- [46] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiacob, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuanc, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.
- [47] Daniel M Cer, Marie-Catherine De Marneffe, Daniel Jurafsky, and Christopher D Manning. Parsing to Stanford Dependencies: Trade-offs between speed and accuracy. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pages 1628–1632, 2010.
- [48] Yung-Chun Chang, Chun-Han Chu, Yu-Chen Su, Chien Chin Chen, and Wen-Lian Hsu. Pipe: a protein–protein interaction passage extraction module for biocreative challenge. *Database*, 2016, 2016.
- [49] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.
- [50] Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*, 2016.
- [51] Muthuraman Chidambaram, Yinfei Yang, Daniel Cer, Steve Yuan, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Learning cross-lingual sentence representations via a multi-task dual-encoder model. *arXiv preprint arXiv:1810.12836*, 2018.

- [52] Hai Leong Chieu and Hwee Tou Ng. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th International Conference on Computational Linguistics-Volume 1*, pages 1–7, 2002.
- [53] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [54] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. In *International Conference on Learning Representations*, 2017.
- [55] Junyoung Chung et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- [56] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine learning*, pages 160–167, 2008.
- [57] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.
- [58] Alexis Conneau et al. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.
- [59] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, 2017.
- [60] Alexis Conneau, Guillaume Lample, Marc’Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*, 2017.
- [61] David PA Corney, Bernard F Buxton, William B Langdon, and David T Jones. Biorat: extracting biological information from full-length papers. *Bioinformatics*, 20(17):3206–3213, 2004.
- [62] Ishita Dasgupta, Demi Guo, Andreas Stuhlmüller, Samuel J Gershman, and Noah D Goodman. Evaluating compositionality in sentence embeddings. *arXiv preprint arXiv:1802.04302*, 2018.

- [63] Marie-Catherine De Marneffe and Christopher D Manning. Stanford typed dependencies manual. Technical report, Stanford University, 2008.
- [64] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.
- [65] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1 (Long and Short Papers)*, pages 4171–4186, 2019.
- [66] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.
- [67] Bhuwan Dhingra, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Linguistic knowledge as memory for recurrent neural networks. *arXiv preprint arXiv:1703.02620*, 2017.
- [68] Jing Ding, Daniel Berleant, Dan Nettleton, and Eve Syrkin Wurtele. Mining medline: abstracts, sentences, or phrases? in *Proceedings of the pacific symposium on biocomputing*, pages 326–337, 2002.
- [69] Bill Dolan, Chris Quirk, and Chris Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proc. of the 20th Int. Conf. on Computational Linguistics*, page 350, 2004.
- [70] Bill Dolan, Chris Quirk, and Chris Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 350–356, 2004.
- [71] Changshun Du and Lei Huang. Text classification research with attention-based recurrent neural networks. *International Journal of Computers Communications & Control*, 13(1):50–61, 2018.
- [72] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for on-line learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

- [73] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*, 2015.
- [74] Katharina Eggensperger, Matthias Feurer, Frank Hutter, James Bergstra, Jasper Snoek, Holger Hoos, and Kevin Leyton-Brown. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, volume 10, page 3, 2013.
- [75] Asif Ekbal, S Mondal, and Sivaji Bandyopadhyay. POS tagging using HMM and rule-based chunking. *The Proceedings of SPSAL*, 8(1):25–28, 2007.
- [76] Thomas S Ferguson. A bayesian analysis of some nonparametric problems. *The Annals of Statistics*, pages 209–230, 1973.
- [77] Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. Named entity recognition through classifier combination. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003-Volume 4*, pages 168–171, 2003.
- [78] Kuniyiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.
- [79] Katrin Fundel, Robert Küffner, and Ralf Zimmer. Relex—relation extraction using dependency parse trees. *Bioinformatics* 23 (3), pages 365–371, 2007.
- [80] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*, 2017.
- [81] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. In *Ninth International Conference on Artificial Neural Networks ICANN*, pages 850–855, 1999.
- [82] Anthony TC Goh. Back-propagation neural networks for modeling complex systems. *Artificial Intelligence in Engineering*, 9(3):143–151, 1995.
- [83] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine learning*, pages 369–376, 2006.

- [84] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.
- [85] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing machines. *arXiv preprint arXiv:1410.5401*, 2014.
- [86] Cristian Grozea. Finding optimal parameter settings for high performance word sense disambiguation. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.
- [87] Kazuma Hashimoto and Yoshimasa Tsuruoka. Neural machine translation with source-side latent graph parsing. *arXiv preprint arXiv:1702.02265*, 2017.
- [88] W Keith Hastings. *Monte Carlo Sampling Methods using Markov Chains and Their Applications*. Oxford University Press, 1970.
- [89] Hua He, Kevin Gimpel, and Jimmy Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586, 2015.
- [90] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
- [91] Karl Moritz Hermann et al. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.
- [92] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [93] Yu-Lun Hsieh, Yung-Chun Chang, Nai-Wen Chang, and Wen-Lian Hsu. Identifying protein-protein interactions in biomedical literature using recurrent neural networks with long short-term memory. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 240–245, 2017.
- [94] Phu Mon Htut, Kyunghyun Cho, and Samuel Bowman. Grammar induction with neural language models: An unusual replication. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 371–373, November 2018.

- [95] Chung-Chi Huang and Zhiyong Lu. Community challenges in biomedical text mining over 10 years: success, failure and the future. *Briefings in bioinformatics*, 17(1):132–144, 2015.
- [96] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.
- [97] Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 897–907, 2016.
- [98] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th Int. Joint Conf. on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, 2015.
- [99] Krzysztof Jassem and Jaroslaw Lipski. A new tool for the bilingual text aligning at the sentence level. In *Proceedings of 16th International Conference on Intelligent Information Systems*, pages 279–286, 06 2008.
- [100] Ruyi Ji, Longyin Wen, Libo Zhang, Dawei Du, Yanjun Wu, Chen Zhao, Xianglong Liu, and Feiyue Huang. Attention convolutional binary neural tree for fine-grained visual categorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10468–10477, 2020.
- [101] Sergio Jimenez et al. UNAL-NLP: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 732–742, 2014.
- [102] Søren Johansen and Katarina Juselius. Maximum likelihood estimation and inference on cointegration—with applications to the demand for money. *Oxford Bulletin of Economics and statistics*, 52(2):169–210, 1990.
- [103] Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proc. of the 2018 Conf. on Emp. Methods in Nat. Lang. Proc.*, 2018.



- [104] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proc. of the 15th Conf. of the European Chapter of the Association for Computational Linguistics, Vol. 2 Short Papers*, pages 427–431, 2016.
- [105] Mikael Kågebäck and Hans Salomonsson. Word sense disambiguation using a Bidirectional LSTM. *arXiv preprint arXiv:1606.03568*, 2016.
- [106] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, 2013.
- [107] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.
- [108] Seonho Kim, Juntae Yoon, Jihoon Yang, and Seog Park. Walk-weighted subsequence kernels for protein-protein interaction extraction. *BMC bioinformatics*, 11(1):107, 2010.
- [109] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
- [110] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2, NIPS’15*, pages 3294–3302, Cambridge, MA, USA, 2015. MIT Press.
- [111] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.
- [112] Dan Klein and Christopher D Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proc. of the 42nd Annual Meeting on Association for Computational Linguistics*, page 478, 2004.
- [113] Barbara J Knowlton, Robert G Morrison, John E Hummel, and Keith J Holyoak. A neurocomputational system for relational reasoning. *Trends in Cognitive Sciences*, 16(7):373–381, 2012.
- [114] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54, 2003.

- [115] Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.
- [116] Dimitrios Kouzis-Loukas. *Learning Scrapy*. Packt Publishing Ltd, 2016.
- [117] Martin Krallinger, Florian Leitner, Carlos Rodriguez-Penagos, and Alfonso Valencia. Overview of the protein-protein interaction annotation extraction task of biocreative ii. *Genome biology*, 9(2):S4, 2008.
- [118] Taku Kudo and Yuji Matsumoto. Chunking with support vector machines. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*, pages 1–8, 2001.
- [119] Taku Kudoh and Yuji Matsumoto. Use of support vector learning for chunk identification. In *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language learning-Volume 7*, pages 142–144, 2000.
- [120] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387, 2016.
- [121] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, 2001.
- [122] Alice Lai et al. Illinois-LH: A denotational and distributional approach to semantics. In *Proc. of the 8th Int. Workshop on Semantic Evaluation*, pages 329–334, 2014.
- [123] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.
- [124] Thomas K Landauer, Danielle S McNamara, Simon Dennis, and Walter Kintsch. *Handbook of latent semantic analysis*. Psychology Press, 2013.
- [125] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Int. Conf. on Machine Learning*, pages 1188–1196, 2014.

- [126] Claudia Leacock and Martin Chodorow. Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283, 1998.
- [127] Robert Leaman and Zhiyong Lu. Taggerone: joint named entity recognition and normalization with semi-markov models. *Bioinformatics*, 32(18):2839–2846, 2016.
- [128] Yoong Keok Lee, Hwee Tou Ng, and Tee Kiah Chia. Supervised word sense disambiguation with support vector machines and multiple knowledge sources. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.
- [129] Artuur Leeuwenberg, Aleksey Buzmakov, Yannick Toussaint, and Amedeo Napoli. Exploring pattern structures of syntactic trees for relation extraction. In *International Conference on Formal Concept Analysis*, pages 153–168. Springer, 2015.
- [130] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Molding CNNs for text: non-linear, non-consecutive convolutions. *arXiv preprint arXiv:1508.04112*, 2015.
- [131] Jure Leskovec, Natasa Milic-Frayling, and Marko Grobelnik. Extracting summary sentences based on the document semantic graph. manuscript, 2005.
- [132] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.
- [133] Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eudard Hovy. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185*, 2015.
- [134] Ziran Li, Ning Ding, Zhiyuan Liu, Haitao Zheng, and Ying Shen. Chinese relation extraction with multi-grained information and external linguistic knowledge. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4377–4386, 2019.
- [135] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.
- [136] Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of lstms to learn syntax-sensitive dependencies. *arXiv preprint arXiv:1611.01368*, 2016.

- [137] Biao Liu, Minlie Huang, Jiashen Sun, and Xuan Zhu. Incorporating domain and sentiment supervision in representation learning for domain adaptation. In *Proc. of the Twenty-Fourth Int. Joint Conf. on Artificial Intelligence*, pages 1277–1283, 2015.
- [138] Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. Empower sequence labeling with task-aware neural language model. *arXiv preprint arXiv:1709.04109*, 2017.
- [139] Shengyu Liu, Buzhou Tang, Qingcai Chen, and Xiaolong Wang. Drug-drug interaction extraction via convolutional neural networks. *Computational and mathematical methods in medicine*, 2016, 2016.
- [140] Wenfeng Liu, Peiyu Liu, Yuzhen Yang, Yaling Gao, and Jing Yi. An attention-based syntax-tree and tree-lstm model for sentence summarization. *International Journal of Performability Engineering*, 13(5):775, 2017.
- [141] Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. Learning natural language inference using bidirectional lstm model and inner-attention. *arXiv preprint arXiv:1605.09090*, 2016.
- [142] Chi-kiu Lo, Cyril Goutte, and Michel Simard. CNRC at Semeval-2016 task 1: Experiments in crosslingual semantic textual similarity. In *Proc. of the 10th Int. Workshop on Sem. Eval. (SemEval-2016)*, pages 668–673, 2016.
- [143] Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. In *6th International Conference on Learning Representations (ICLR) Conference Track Proceedings*, 2018.
- [144] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.
- [145] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via Bi-directional LSTM-CNNs-CRF. *arXiv preprint arXiv:1603.01354*, 2016.
- [146] Nabin Maharjan, Rajendra Banjade, Nobal Bikram Niraula, and Vasile Rus. Sema-ligner: A method and tool for aligning chunks with semantic relation types and semantic similarity scores. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*, pages 1207–1211, 2016.
- [147] Christopher D Manning. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 171–189, 2011.

- [148] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.
- [149] Mitchell Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. Treebank-3 ldc99t42. CD-ROM. Philadelphia, Penn.: Linguistic Data Consortium, 1999.
- [150] Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 1–8, 2014.
- [151] Marco Marelli et al. SemEval-2014 Task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proc. of the 8th Int. Wkshp. on Semantic Evaluation*, pages 1–8, 2014.
- [152] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, Roberto Zamparelli, et al. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*, pages 216–223, 2014.
- [153] Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 591–598, 2000.
- [154] Ryan McDonald, Koby Crammer, and Fernando Pereira. Flexible text segmentation with structured multilabel classification. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 987–994, 2005.
- [155] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with Bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, 2016.
- [156] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

- [157] George A Miller. Wordnet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.
- [158] Makoto Miwa, Rune Sætre, Yusuke Miyao, and Jun’ichi Tsujii. A rich feature vector for protein-protein interaction extraction from multiple corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 121–130. Association for Computational Linguistics, 2009.
- [159] Makoto Miwa, Rune Sætre, Yusuke Miyao, and Jun’ichi Tsujii. Protein–protein interaction extraction by leveraging multiple kernels and parsers. *International journal of medical informatics*, 78(12):e39–e46, 2009.
- [160] Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. *Advances in neural information processing systems*, 21:1081–1088, 2008.
- [161] Andrea Moro and Roberto Navigli. Semeval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 288–297, 2015.
- [162] Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244, 2014.
- [163] Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, volume 16, pages 2786–2792, 2016.
- [164] Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI’16, pages 2786–2792. AAAI Press, 2016.
- [165] Roberto Navigli. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10, 2009.
- [166] Roberto Navigli, Kenneth C Litkowski, and Orin Hargraves. Semeval-2007 task 07: Coarse-grained english all-words task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 30–35. Association for Computational Linguistics, 2007.
- [167] Claire Nédellec. Learning language in logic – genic interaction extraction challenge. in *Proceedings of the 4th Learning Language in Logic Workshop*, pages 31–37, 2005.

- [168] Nam Nguyen and Yunsong Guo. Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the 24th International Conference on Machine learning*, pages 681–688, 2007.
- [169] Xuan-Phi Nguyen, Shafiq Joty, Steven CH Hoi, and Richard Socher. Tree-structured attention with hierarchical accumulation. *arXiv preprint arXiv:2002.08046*, 2020.
- [170] Jessica Ouyang and Kathy McKeown. Neural network alignment for sentential paraphrases. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4724–4735, Florence, Italy, July 2019. Association for Computational Linguistics.
- [171] Abel L Packer. The scielo open access: a gold way from the south. *Canadian Journal of Higher Education*, 39(3):111–126, 2009.
- [172] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(4):694–707, 2016.
- [173] Vassilis Papavassiliou, Prokopis Prokopidis, and Stelios Piperidis. Discovering Parallel Language Resources for Training MT Engines. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 7-12, 2018 2018. European Language Resources Association (ELRA).
- [174] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, 2016.
- [175] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proc. of the 2016 Conf. on Empirical Methods in Natural Language Processing*, pages 2249–2255, November 2016.
- [176] Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, 2016.
- [177] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

- [178] Alexandre Passos, Vineet Kumar, and Andrew McCallum. Lexicon infused phrase embeddings for named entity resolution. *arXiv preprint arXiv:1404.5367*, 2014.
- [179] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [180] Yifan Peng, Samir Gupta, Cathy Wu, and Vijay Shanker. An extended dependency graph for relation extraction in biomedical texts. *Proceedings of BioNLP 15*, pages 21–30, 2015.
- [181] Yifan Peng and Zhiyong Lu. Deep learning for extracting protein-protein interactions from biomedical literature. *arXiv preprint arXiv:1706.01556*, 2017.
- [182] Yifan Peng, Chih-Hsuan Wei, and Zhiyong Lu. Improving chemical disease relation extraction with rich features and weakly labeled data. *Journal of cheminformatics*, 8(1):53, 2016.
- [183] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.
- [184] Ahmad Pesaranghader, Ali Pesaranghader, Stan Matwin, and Marina Sokolova. One single deep bidirectional LSTM network for word sense disambiguation of text data. In *Advances in Artificial Intelligence - 31st Canadian Conference on Artificial Intelligence, Canadian AI 2018, Toronto, ON, Canada, May 8-11, 2018, Proceedings*, pages 96–107, 2018.
- [185] Sampo Pyysalo, Antti Airola, Juho Heimonen, Jari Björne, Filip Ginter, and Tapio Salakoski. Comparative analysis of five protein-protein interaction corpora. *BMC Bioinformatics*, 9(3):S6, 2008.
- [186] Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björnee, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Bioinfer: a corpus for information extraction in the biomedical domain. *BMC Bioinformatics* 8(50), pages 1–24, 2007.
- [187] Chanqin Quan, Lei Hua, Xiao Sun, and Wenjun Bai. Multichannel convolutional neural network for biological relation extraction. *BioMed research international*, 2016, 2016.



- [188] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.
- [189] Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1156–1167, 2017.
- [190] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155, 2009.
- [191] Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Conference on Empirical Methods in Natural Language Processing*, 1996.
- [192] Marek Rei. Semi-supervised multitask learning for sequence labeling. *arXiv preprint arXiv:1704.07156*, 2017.
- [193] Tim Rocktäschel et al. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.
- [194] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
- [195] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, pages 1842–1850, 2016.
- [196] Adam Santoro and et al. Relational recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 7299–7310, 2018.
- [197] Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter W. Battaglia, and Timothy P. Lillicrap. A simple neural network module for relational reasoning. *CoRR*, abs/1706.01427, 2017.
- [198] Daniel L Schacter and Endel Tulving. *Memory Systems*. MIT Press, 1994.
- [199] Isabel Segura-Bedmar, Paloma Martínez, and César de Pablo-Sánchez. A linguistic rule-based approach to extract drug-drug interactions from pharmacological documents. *BMC Bioinformatics*, 12(2):S1, 2011.
- [200] Rico Sennrich and Barry Haddow. Linguistic input features improve neural machine translation. *arXiv preprint arXiv:1606.02892*, 2016.

- [201] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141, 2003.
- [202] Yang Shao. HCTI at SemEval-2017 task 1: Use convolutional neural network to evaluate semantic textual similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 130–133, Vancouver, Canada, August 2017. Association for Computational Linguistics.
- [203] Hong Shen and Anoop Sarkar. Voting between multiple data representations for text chunking. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 389–400, 2005.
- [204] Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. Neural language modeling by jointly learning syntax and lexicon. In *6th International Conference on Learning Representations (ICLR) Conference Track Proceedings*, 2018.
- [205] Ayush Singhal, Michael Simmons, and Zhiyong Lu. Text mining genotype-phenotype relationships from biomedical literature for database curation and precision medicine. *PLoS computational biology*, 12(11):e1005017, 2016.
- [206] Benjamin Snyder and Martha Palmer. The english all-words task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.
- [207] Felipe Soares, Viviane Moreira, and Karin Becker. A large parallel corpus of full-text scientific articles. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).
- [208] Richard Socher and et al. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809, 2011.
- [209] Richard Socher, Eric H Huang, Jeffrey Pennington, Christopher D Manning, and Andrew Y Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809, 2011.
- [210] Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proc. of the 2012 Joint*

- Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, 2012.
- [211] Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, 2012.
- [212] Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218, 2014.
- [213] Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th Int. Conf. on Machine Learning (ICML-11)*, pages 129–136, 2011.
- [214] Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML’11*, pages 129–136, 2011.
- [215] Richard Socher, Alex Perelygin, Jean Wu, JasExploiting mas-sively parallel news sourceson Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of the 2013 Conf. on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.
- [216] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.
- [217] Anders Søgaard. Semisupervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 48–52, 2011.
- [218] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

- [219] Carlo Strapparava, Alfio Gliozzo, and Claudiu Giuliano. Pattern abstraction and term similarity for word sense disambiguation: 1st at senseval-3. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.
- [220] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. Weakly supervised memory networks. *CoRR*, abs/1503.08895, 2015.
- [221] Md Arafat Sultan, Steven Bethard, and Tamara Sumner. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. *Transactions of the Association for Computational Linguistics*, 2:219–230, 2014.
- [222] Xu Sun. Structure regularization for structured prediction. In *Advances in Neural Information Processing Systems*, pages 2402–2410, 2014.
- [223] Xu Sun, Louis-Philippe Morency, Daisuke Okanohara, and Jun’ichi Tsujii. Modeling latent-dynamic in shallow parsing: A latent conditional model with improved inference. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 841–848, 2008.
- [224] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.
- [225] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063, 2000.
- [226] Kaveh Taghipour and Hwee Tou Ng. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 314–323, 2015.
- [227] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.
- [228] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the*

- 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, 2015.
- [229] Trias Thireou and Martin Reczko. Bidirectional long short-term memory networks for predicting the subcellular localization of eukaryotic proteins. *IEEE/ACM transactions on computational biology and bioinformatics*, 4(3), 2007.
- [230] Sebastian B. Thrun. Efficient exploration in reinforcement learning. Technical report, Carnegie Mellon University, 1992.
- [231] Domonkos Tikk, Philippe Thomas, Peter Palaga, Jörg Hakenberg, and Ulf Leser. A comprehensive benchmark of kernel methods to extract protein–protein interactions from literature. *PLoS computational biology*, 6(7):e1000837, 2010.
- [232] Erik F Tjong Kim Sang and Sabine Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning-Volume 7*, pages 127–132, 2000.
- [233] Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003-Volume 4*, pages 142–147, 2003.
- [234] K. Tran and Y. Bisk. Inducing Grammars with and for Neural Machine Translation. *ArXiv e-prints*, May 2018.
- [235] Ke Tran and Yonatan Bisk. Inducing grammars with and for neural machine translation. In *Proc. of the 2nd Workshop on Neural Machine Translation and Generation*, page 25–35, July 2018.
- [236] Rocco Tripodi and Marcello Pelillo. A game-theoretic approach to word sense disambiguation. *Computational Linguistics*, 43(1):31–70, 2017.
- [237] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394, 2010.
- [238] William Thomas Tutte. Graph theory, volume 21 of encyclopedia of mathematics and its applications, 1984.

- [239] Sofie Van Landeghem, Yvan Saeys, Bernard De Baets, and Yves Van de Peer. Extracting protein-protein interactions from text using rich feature vectors and feature selection. In *3rd International symposium on Semantic Mining in Biomedicine (SMBM 2008)*, pages 77–84. Turku Centre for Computer Sciences (TUCS), 2008.
- [240] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.
- [241] Albert Weichselbraun, Max Göbel, and Fabian Odoni. inscriptis. <https://github.com/weblyzard/inscriptis>, 2016.
- [242] Adina Williams, Andrew Drozdov\*, and Samuel R. Bowman. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics*, 6:253–267, 2018.
- [243] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.
- [244] Yinfei Yang, Gustavo Hernandez Abrego, Steve Yuan, Mandy Guo, Qinlan Shen, Daniel Cer, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Improving multilingual sentence embedding using bi-directional dual encoder with additive margin softmax. *arXiv preprint arXiv:1902.08564*, 2019.
- [245] Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-Hsuan Sung, et al. Multilingual universal sentence encoder for semantic retrieval. *arXiv preprint arXiv:1907.04307*, 2019.
- [246] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. XLNnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, pages 5753–5763, 2019.
- [247] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.
- [248] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016*

- Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.
- [249] Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. Semi-markov phrase-based monolingual alignment. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 590–600, 2013.
- [250] David Yarowsky. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 88–95, 1994.
- [251] Wenpeng Yin et al. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Trans. of the Assoc. for Computational Linguistics*, 4:259–272, 2016.
- [252] Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. Learning to compose words into sentences with reinforcement learning. In *International Conference on Learning Representations*, 2017.
- [253] Lei Yu, Cyprien de Masson d’Autume, Chris Dyer, Phil Blunsom, Lingpeng Kong, and Wang Ling. Sentence encoding with tree-constrained relation networks. *arXiv preprint arXiv:1811.10475*, 2018.
- [254] Ruichi Yu, Ang Li, Vlad I Morariu, and Larry S Davis. Visual relationship detection with internal and external linguistic knowledge distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1974–1982, 2017.
- [255] Xiang Yu, Agnieszka Faleńska, and Ngoc Thang Vu. A general-purpose tagger with convolutional neural networks. *arXiv preprint arXiv:1706.01723*, 2017.
- [256] Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. Semi-supervised word sense disambiguation with neural models. *arXiv preprint arXiv:1603.07012*, 2016.
- [257] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, 2014.
- [258] Tianyang Zhang, Minlie Huang, and Li Zhao. Learning structured representation for text classification via reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 6053–6060, 2018.

- [259] Tianyang Zhang, Minlie Huang, and Li Zhao. Learning structured representation for text classification via reinforcement learning. In *Proc. of the Thirty-Second AAAI Conf. on Artificial Intelligence*, pages 6053–6060, 2018.
- [260] Yijia Zhang, Hongfei Lin, Zhihao Yang, and Yanpeng Li. Neighborhood hash graph kernel for protein–protein interaction extraction. *Journal of biomedical informatics*, 44(6):1086–1092, 2011.
- [261] Yijia Zhang, Hongfei Lin, Zhihao Yang, Jian Wang, Shaowu Zhang, Yuanyuan Sun, and Liang Yang. A hybrid model based on neural networks for biomedical relation extraction. *Journal of biomedical informatics*, 81:83–92, 2018.
- [262] Yue Zhang, Qi Liu, and Linfeng Song. Sentence-state lstm for text representation. *arXiv preprint arXiv:1805.02474*, 2018.
- [263] Han Zhao, Zhengdong Lu, and Pascal Poupart. Self-adaptive hierarchical sentence model. In *IJCAI*, pages 4069–4076, 2015.
- [264] Jiang Zhao, Tiantian Zhu, and Man Lan. Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 271–277, 2014.
- [265] Jiang Zhao, Tiantian Zhu, and Man Lan. Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 271–277, 01 2014.
- [266] Zhehuan Zhao, Zhihao Yang, Hongfei Lin, Jian Wang, and Song Gao. A protein-protein interaction extraction approach based on deep neural network. *International Journal of Data Mining and Bioinformatics*, 15(2):145–164, 2016.
- [267] Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657, 2013.
- [268] Zhi Zhong and Hwee Tou Ng. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 system demonstrations*, pages 78–83. Association for Computational Linguistics, 2010.



- [269] GuoDong Zhou and Jian Su. Named entity recognition using an HMM-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 473–480, 2002.
- [270] Yao Zhou, Cong Liu, and Yan Pan. Modelling sentence pairs with tree-structured attentive encoder. *arXiv preprint arXiv:1610.02806*, 2016.

# **Appendix A**

## **Copyright Forms of the Papers**

There are eleven published papers included in this thesis. Copyright release forms are included in this appendix.

**SPRINGER NATURE LICENSE  
TERMS AND CONDITIONS**

Mar 01, 2021

---

This Agreement between Mr. MAHTAB AHMED ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

License Number	5016931043755
License date	Feb 27, 2021
Licensed Content Publisher	Springer Nature
Licensed Content Publication	Springer eBook
Licensed Content Title	Investigating Relational Recurrent Neural Networks with Variable Length Memory Pointer
Licensed Content Author	Mahtab Ahmed, Robert E. Mercer
Licensed Content Date	Jan 1, 2020
Type of Use	Thesis/Dissertation
Requestor type	academic/university or research institute
Format	electronic
Portion	full article/chapter
Will you be	no

translating?


Circulation/distribution 500 - 999

Author of this Springer  
Nature content yes

Title Generating Effective Sentence Representations in Semantic Space  
using Deep Learning and Reinforcement Learning

Institution name University of Western Ontario

Expected presentation  
date Apr 2021

Requestor Location  
Mr. MAHTAB AHMED  
  
Attn: Mr. MAHTAB AHMED

Total 0.00 CAD

Terms and Conditions

### Springer Nature Customer Service Centre GmbH Terms and Conditions

This agreement sets out the terms and conditions of the licence (the **Licence**) between you and **Springer Nature Customer Service Centre GmbH** (the **Licensor**). By clicking 'accept' and completing the transaction for the material (**Licensed Material**), you also confirm your acceptance of these terms and conditions.

#### 1. Grant of License

**1.1.** The Licensor grants you a personal, non-exclusive, non-transferable, world-wide licence to reproduce the Licensed Material for the purpose specified in your order only. Licences are granted for the specific use requested in the order and for no other use, subject to the conditions below.

**1.2.** The Licensor warrants that it has, to the best of its knowledge, the rights to license reuse of the Licensed Material. However, you should ensure that the material you are requesting is original to the Licensor and does not carry the copyright of

another entity (as credited in the published version).

**1. 3.** If the credit line on any part of the material you have requested indicates that it was reprinted or adapted with permission from another source, then you should also seek permission from that source to reuse the material.

## 2. Scope of Licence

**2. 1.** You may only use the Licensed Content in the manner and to the extent permitted by these Ts&Cs and any applicable laws.

**2. 2.** A separate licence may be required for any additional use of the Licensed Material, e.g. where a licence has been purchased for print only use, separate permission must be obtained for electronic re-use. Similarly, a licence is only valid in the language selected and does not apply for editions in other languages unless additional translation rights have been granted separately in the licence. Any content owned by third parties are expressly excluded from the licence.

**2. 3.** Similarly, rights for additional components such as custom editions and derivatives require additional permission and may be subject to an additional fee. Please apply to [Journalpermissions@springernature.com/bookpermissions@springernature.com](mailto:Journalpermissions@springernature.com/bookpermissions@springernature.com) for these rights.

**2. 4.** Where permission has been granted **free of charge** for material in print, permission may also be granted for any electronic version of that work, provided that the material is incidental to your work as a whole and that the electronic version is essentially equivalent to, or substitutes for, the print version.

**2. 5.** An alternative scope of licence may apply to signatories of the [STM Permissions Guidelines](#), as amended from time to time.

## 3. Duration of Licence

**3. 1.** A licence for is valid from the date of purchase ('Licence Date') at the end of the relevant period in the below table:

Scope of Licence	Duration of Licence
Post on a website	12 months
Presentations	12 months
Books and journals	Lifetime of the edition in the language purchased

## 4. Acknowledgement

**4. 1.** The Licensor's permission must be acknowledged next to the Licenced Material in print. In electronic form, this acknowledgement must be visible at the same time as the figures/tables/illustrations or abstract, and must be hyperlinked to the journal/book's homepage. Our required acknowledgement format is in the Appendix below.

## 5. Restrictions on use

**5. 1.** Use of the Licensed Material may be permitted for incidental promotional use and minor editing privileges e.g. minor adaptations of single figures, changes of format, colour and/or style where the adaptation is credited as set out in Appendix 1 below. Any other changes including but not limited to, cropping, adapting, omitting material that affect the meaning, intention or moral rights of the author are strictly prohibited.

**5. 2.** You must not use any Licensed Material as part of any design or trademark.

**5. 3.** Licensed Material may be used in Open Access Publications (OAP) before publication by Springer Nature, but any Licensed Material must be removed from OAP sites prior to final publication.

## 6. Ownership of Rights

**6. 1.** Licensed Material remains the property of either Licensor or the relevant third party and any rights not explicitly granted herein are expressly reserved.

## 7. Warranty

IN NO EVENT SHALL LICENSOR BE LIABLE TO YOU OR ANY OTHER PARTY OR ANY OTHER PERSON OR FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ARISING OUT OF OR IN CONNECTION WITH THE DOWNLOADING, VIEWING OR USE OF THE MATERIALS REGARDLESS OF THE FORM OF ACTION, WHETHER FOR BREACH OF CONTRACT, BREACH OF WARRANTY, TORT, NEGLIGENCE, INFRINGEMENT OR OTHERWISE (INCLUDING, WITHOUT LIMITATION, DAMAGES BASED ON LOSS OF PROFITS, DATA, FILES, USE, BUSINESS OPPORTUNITY OR CLAIMS OF THIRD PARTIES), AND WHETHER OR NOT THE PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY PROVIDED HEREIN.

## 8. Limitations

**8. 1. BOOKS ONLY:** Where 'reuse in a dissertation/thesis' has been selected the following terms apply: Print rights of the final author's accepted manuscript (for clarity, NOT the published version) for up to 100 copies, electronic rights for use only on a personal website or institutional repository as defined by the Sherpa guideline ([www.sherpa.ac.uk/romeo/](http://www.sherpa.ac.uk/romeo/)).

**8. 2.** For content reuse requests that qualify for permission under the [STM Permissions Guidelines](#), which may be updated from time to time, the STM Permissions Guidelines supersede the terms and conditions contained in this licence.

## 9. Termination and Cancellation

**9. 1.** Licences will expire after the period shown in Clause 3 (above).

**9. 2.** Licensee reserves the right to terminate the Licence in the event that payment is not received in full or if there has been a breach of this agreement by you.

### **Appendix 1 — Acknowledgements:**

#### **For Journal Content:**

Reprinted by permission from [the Licensor]: [Journal Publisher (e.g. Nature/Springer/Palgrave)] [JOURNAL NAME] [REFERENCE CITATION (Article name, Author(s) Name), [COPYRIGHT] (year of publication)]

#### **For Advance Online Publication papers:**

Reprinted by permission from [the Licensor]: [Journal Publisher (e.g. Nature/Springer/Palgrave)] [JOURNAL NAME] [REFERENCE CITATION (Article name, Author(s) Name), [COPYRIGHT] (year of publication), advance online publication, day month year (doi: 10.1038/sj.[JOURNAL ACRONYM].)]

#### **For Adaptations/Translations:**

Adapted/Translated by permission from [the Licensor]: [Journal Publisher (e.g. Nature/Springer/Palgrave)] [JOURNAL NAME] [REFERENCE CITATION (Article name, Author(s) Name), [COPYRIGHT] (year of publication)]

#### **Note: For any republication from the British Journal of Cancer, the following credit line style applies:**

Reprinted/adapted/translated by permission from [the Licensor]: on behalf of Cancer Research UK: : [Journal Publisher (e.g. Nature/Springer/Palgrave)] [JOURNAL NAME] [REFERENCE CITATION (Article name, Author(s) Name), [COPYRIGHT] (year of publication)]

#### **For Advance Online Publication papers:**

Reprinted by permission from The [the Licensor]: on behalf of Cancer Research UK: [Journal Publisher (e.g. Nature/Springer/Palgrave)] [JOURNAL NAME] [REFERENCE CITATION (Article name, Author(s) Name), [COPYRIGHT] (year of publication), advance online publication, day month year (doi: 10.1038/sj.[JOURNAL ACRONYM].)]

#### **For Book content:**

Reprinted/adapted by permission from [the Licensor]: [Book Publisher (e.g. Palgrave Macmillan, Springer etc)] [Book Title] by [Book author(s)] [COPYRIGHT] (year of publication)]

### **Other Conditions:**

Version 1.3

3/1/2021

RightsLink Printable License

**Questions? [customercare@copyright.com](mailto:customercare@copyright.com) or +1-855-239-3415 (toll free in the US) or +1-978-646-2777.**

---



**SPRINGER NATURE LICENSE  
TERMS AND CONDITIONS**

Mar 01, 2021

---

This Agreement between Mr. MAHTAB AHMED ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

License Number      5016930776959

License date      Feb 27, 2021

Licensed Content  
Publisher      Springer Nature

Licensed Content  
Publication      Springer eBook

Licensed Content Title      Efficient Transformer-Based Sentence Encoding for Sentence Pair  
Modelling

Licensed Content  
Author      Mahtab Ahmed, Robert E. Mercer

Licensed Content Date      Jan 1, 2019

Type of Use      Thesis/Dissertation

Requestor type      academic/university or research institute

Format      electronic

Portion      full article/chapter

Will you be      no

translating?


Circulation/distribution 500 - 999

Author of this Springer  
Nature content yes

Title Generating Effective Sentence Representations in Semantic Space  
using Deep Learning and Reinforcement Learning

Institution name University of Western Ontario

Expected presentation  
date Apr 2021

Requestor Location Mr. MAHTAB AHMED  


Attn: Mr. MAHTAB AHMED

Total 0.00 CAD

Terms and Conditions

### Springer Nature Customer Service Centre GmbH Terms and Conditions

This agreement sets out the terms and conditions of the licence (the **Licence**) between you and **Springer Nature Customer Service Centre GmbH** (the **Licensor**). By clicking 'accept' and completing the transaction for the material (**Licensed Material**), you also confirm your acceptance of these terms and conditions.

#### 1. Grant of License

**1.1.** The Licensor grants you a personal, non-exclusive, non-transferable, world-wide licence to reproduce the Licensed Material for the purpose specified in your order only. Licences are granted for the specific use requested in the order and for no other use, subject to the conditions below.

**1.2.** The Licensor warrants that it has, to the best of its knowledge, the rights to license reuse of the Licensed Material. However, you should ensure that the material you are requesting is original to the Licensor and does not carry the copyright of

another entity (as credited in the published version).

**1. 3.** If the credit line on any part of the material you have requested indicates that it was reprinted or adapted with permission from another source, then you should also seek permission from that source to reuse the material.

## 2. Scope of Licence

**2. 1.** You may only use the Licensed Content in the manner and to the extent permitted by these Ts&Cs and any applicable laws.

**2. 2.** A separate licence may be required for any additional use of the Licensed Material, e.g. where a licence has been purchased for print only use, separate permission must be obtained for electronic re-use. Similarly, a licence is only valid in the language selected and does not apply for editions in other languages unless additional translation rights have been granted separately in the licence. Any content owned by third parties are expressly excluded from the licence.

**2. 3.** Similarly, rights for additional components such as custom editions and derivatives require additional permission and may be subject to an additional fee. Please apply to [Journalpermissions@springernature.com/bookpermissions@springernature.com](mailto:Journalpermissions@springernature.com/bookpermissions@springernature.com) for these rights.

**2. 4.** Where permission has been granted **free of charge** for material in print, permission may also be granted for any electronic version of that work, provided that the material is incidental to your work as a whole and that the electronic version is essentially equivalent to, or substitutes for, the print version.

**2. 5.** An alternative scope of licence may apply to signatories of the [STM Permissions Guidelines](#), as amended from time to time.

## 3. Duration of Licence

**3. 1.** A licence for is valid from the date of purchase ('Licence Date') at the end of the relevant period in the below table:

Scope of Licence	Duration of Licence
Post on a website	12 months
Presentations	12 months
Books and journals	Lifetime of the edition in the language purchased

## 4. Acknowledgement

**4. 1.** The Licensor's permission must be acknowledged next to the Licenced Material in print. In electronic form, this acknowledgement must be visible at the same time as the figures/tables/illustrations or abstract, and must be hyperlinked to the journal/book's homepage. Our required acknowledgement format is in the Appendix below.

## 5. Restrictions on use

**5. 1.** Use of the Licensed Material may be permitted for incidental promotional use and minor editing privileges e.g. minor adaptations of single figures, changes of format, colour and/or style where the adaptation is credited as set out in Appendix 1 below. Any other changes including but not limited to, cropping, adapting, omitting material that affect the meaning, intention or moral rights of the author are strictly prohibited.

**5. 2.** You must not use any Licensed Material as part of any design or trademark.

**5. 3.** Licensed Material may be used in Open Access Publications (OAP) before publication by Springer Nature, but any Licensed Material must be removed from OAP sites prior to final publication.

## 6. Ownership of Rights

**6. 1.** Licensed Material remains the property of either Licensor or the relevant third party and any rights not explicitly granted herein are expressly reserved.

## 7. Warranty

IN NO EVENT SHALL LICENSOR BE LIABLE TO YOU OR ANY OTHER PARTY OR ANY OTHER PERSON OR FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ARISING OUT OF OR IN CONNECTION WITH THE DOWNLOADING, VIEWING OR USE OF THE MATERIALS REGARDLESS OF THE FORM OF ACTION, WHETHER FOR BREACH OF CONTRACT, BREACH OF WARRANTY, TORT, NEGLIGENCE, INFRINGEMENT OR OTHERWISE (INCLUDING, WITHOUT LIMITATION, DAMAGES BASED ON LOSS OF PROFITS, DATA, FILES, USE, BUSINESS OPPORTUNITY OR CLAIMS OF THIRD PARTIES), AND WHETHER OR NOT THE PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY PROVIDED HEREIN.

## 8. Limitations

**8. 1. BOOKS ONLY:** Where 'reuse in a dissertation/thesis' has been selected the following terms apply: Print rights of the final author's accepted manuscript (for clarity, NOT the published version) for up to 100 copies, electronic rights for use only on a personal website or institutional repository as defined by the Sherpa guideline ([www.sherpa.ac.uk/romeo/](http://www.sherpa.ac.uk/romeo/)).

**8. 2.** For content reuse requests that qualify for permission under the [STM Permissions Guidelines](#), which may be updated from time to time, the STM Permissions Guidelines supersede the terms and conditions contained in this licence.

## 9. Termination and Cancellation

**9. 1.** Licences will expire after the period shown in Clause 3 (above).

**9. 2.** Licensee reserves the right to terminate the Licence in the event that payment is not received in full or if there has been a breach of this agreement by you.

### **Appendix 1 — Acknowledgements:**

#### **For Journal Content:**

Reprinted by permission from [the Licensor]: [Journal Publisher (e.g. Nature/Springer/Palgrave)] [JOURNAL NAME] [REFERENCE CITATION (Article name, Author(s) Name), [COPYRIGHT] (year of publication)]

#### **For Advance Online Publication papers:**

Reprinted by permission from [the Licensor]: [Journal Publisher (e.g. Nature/Springer/Palgrave)] [JOURNAL NAME] [REFERENCE CITATION (Article name, Author(s) Name), [COPYRIGHT] (year of publication), advance online publication, day month year (doi: 10.1038/sj.[JOURNAL ACRONYM].)]

#### **For Adaptations/Translations:**

Adapted/Translated by permission from [the Licensor]: [Journal Publisher (e.g. Nature/Springer/Palgrave)] [JOURNAL NAME] [REFERENCE CITATION (Article name, Author(s) Name), [COPYRIGHT] (year of publication)]

#### **Note: For any republication from the British Journal of Cancer, the following credit line style applies:**

Reprinted/adapted/translated by permission from [the Licensor]: on behalf of Cancer Research UK: : [Journal Publisher (e.g. Nature/Springer/Palgrave)] [JOURNAL NAME] [REFERENCE CITATION (Article name, Author(s) Name), [COPYRIGHT] (year of publication)]

#### **For Advance Online Publication papers:**

Reprinted by permission from The [the Licensor]: on behalf of Cancer Research UK: [Journal Publisher (e.g. Nature/Springer/Palgrave)] [JOURNAL NAME] [REFERENCE CITATION (Article name, Author(s) Name), [COPYRIGHT] (year of publication), advance online publication, day month year (doi: 10.1038/sj.[JOURNAL ACRONYM].)]

#### **For Book content:**

Reprinted/adapted by permission from [the Licensor]: [Book Publisher (e.g. Palgrave Macmillan, Springer etc)] [Book Title] by [Book author(s)] [COPYRIGHT] (year of publication)]

### **Other Conditions:**

Version 1.3

3/1/2021

RightsLink Printable License

**Questions? [customercare@copyright.com](mailto:customercare@copyright.com) or +1-855-239-3415 (toll free in the US) or +1-978-646-2777.**

---



RightsLink®



Home



Help



Email Support



MAHTAB AHMED ▾



## Improving Neural Sequence Labelling Using Additional Linguistic Information

Conference Proceedings:

2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)

Author: Mahtab Ahmed

Publisher: IEEE

Date: Dec 2018

Copyright © 2018, IEEE

### Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis online.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK

CLOSE WINDOW



RightsLink®



Home



Help



Email Support



MAHTAB AHMED ▾



## A Novel Neural Sequence Model with Multiple Attentions for Word Sense Disambiguation

Conference Proceedings:

2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)

Author: Mahtab Ahmed

Publisher: IEEE

Date: Dec 2018

Copyright © 2018, IEEE

### Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis online.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK

CLOSE WINDOW





Home



Help



Email Support



MAHTAB AHMED ▾



### Improving Tree-LSTM with Tree Attention

Conference Proceedings:

2019 IEEE 13th International Conference on Semantic Computing (ICSC)

Author: Mahtab Ahmed

Publisher: IEEE

Date: Jan. 2019

Copyright © 2019, IEEE

#### Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

[BACK](#)
[CLOSE WINDOW](#)



RightsLink®



Home



Help



Email Support



MAHTAB AHMED ▾



## Identifying Protein-Protein Interaction Using Tree LSTM and Structured Attention

Conference Proceedings:

2019 IEEE 13th International Conference on Semantic Computing (ICSC)

Author: Mahtab Ahmed

Publisher: IEEE

Date: Jan. 2019

Copyright © 2019, IEEE

### Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis online.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK

CLOSE WINDOW



RightsLink®



Home



Help



Email Support



Sign in



Create Account



## Multilingual Semantic Textual Similarity using Multilingual Word Representations

Conference Proceedings:

2020 IEEE 14th International Conference on Semantic Computing (ICSC)

Author: Mahtab Ahmed

Publisher: IEEE

Date: Feb. 2020

Copyright © 2020, IEEE

### Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis online.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK

CLOSE WINDOW



# Association for the Advancement of Artificial Intelligence

2275 East Bayshore Road, Suite 160

Palo Alto, California 94303 USA

## AAAI COPYRIGHT FORM

Title of Article/Paper: Modelling Sentence Pairs via Reinforcement Learning: An Actor-Critic Approach to Lear

Publication in Which Article/Paper Is to Appear: Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAI

Author's Name(s): Mahtab Ahmed and Robert E. Mercer

Please type or print your name(s) as you wish it (them) to appear in print

### PART A – COPYRIGHT TRANSFER FORM

The undersigned, desiring to publish the above article/paper in a publication of the Association for the Advancement of Artificial Intelligence, (AAAI), hereby transfer their copyrights in the above article/paper to the Association for the Advancement of Artificial Intelligence (AAAI), in order to deal with future requests for reprints, translations, anthologies, reproductions, excerpts, and other publications.

This grant will include, without limitation, the entire copyright in the article/paper in all countries of the world, including all renewals, extensions, and reversions thereof, whether such rights current exist or hereafter come into effect, and also the exclusive right to create electronic versions of the article/paper, to the extent that such right is not subsumed under copyright.

The undersigned warrants that they are the sole author and owner of the copyright in the above article/paper, except for those portions shown to be in quotations; that the article/paper is original throughout; and that the undersigned right to make the grants set forth above is complete and unencumbered.

If anyone brings any claim or action alleging facts that, if true, constitute a breach of any of the foregoing warranties, the undersigned will hold harmless and indemnify AAAI, their grantees, their licensees, and their distributors against any liability, whether under judgment, decree, or compromise, and any legal fees and expenses arising out of that claim or actions, and the undersigned will cooperate fully in any defense AAAI may make to such claim or action. Moreover, the undersigned agrees to cooperate in any claim or other action seeking to protect or enforce any right the undersigned has granted to AAAI in the article/paper. If any such claim or action fails because of facts that constitute a breach of any of the foregoing warranties, the undersigned agrees to reimburse whomever brings such claim or action for expenses and attorneys' fees incurred therein.

### Returned Rights

In return for these rights, AAAI hereby grants to the above author(s), and the employer(s) for whom the work was performed, royalty-free permission to:

1. Retain all proprietary rights other than copyright (such as patent rights).
2. Personal reuse of all or portions of the above article/paper in other works of their own authorship. This does not include granting third-party requests for reprinting, republishing, or other types of reuse. AAAI must handle all such third-party requests.
3. Reproduce, or have reproduced, the above article/paper for the author's personal use, or for company use provided that AAAI copyright and the source are indicated, and that the copies are not used in a way that implies AAAI endorsement of a product or service of an employer, and that the copies per se are not offered for sale. The foregoing right shall not permit the posting of the article/paper in electronic or digital form on any computer network, except by the author or the author's employer, and then only on the author's or the employer's own web page or ftp site. Such web page or ftp site, in addition to the aforementioned requirements of this Paragraph, shall not post other AAAI copyrighted materials not of the author's or the employer's creation (including tables of contents with links to other papers) without AAAI's written permission.
4. Make limited distribution of all or portions of the above article/paper prior to publication.
5. In the case of work performed under a U.S. Government contract or grant, AAAI recognized that the U.S. Government has royalty-free permission to reproduce all or portions of the above Work, and to authorize others to do so, for official U.S. Government purposes only, if the contract or grant so requires.

In the event the above article/paper is not accepted and published by AAAI, or is withdrawn by the author(s) before acceptance by AAAI, this agreement becomes null and void.

(1)



Author/Authorized Agent for Joint Author's Signature

November 21, 2019

Date

Employer for whom work was performed

Title (if not author)

(For jointly authored Works, all joint authors should sign unless one of the authors has been duly authorized to act as agent for the others.)

# Association for the Advancement of Artificial Intelligence

2275 East Bayshore Road, Suite 160

Palo Alto, California 94303 USA

## PART B – U.S. GOVERNMENT EMPLOYEE CERTIFICATION

This will certify that all authors of the above article/paper are employees of the U.S. Government and performed this work as part of their employment, and that the article/paper is therefore not subject to U.S. copyright protection. The undersigned warrants that they are the sole author/translator of the above article/paper, and that the article/paper is original throughout, except for those portions shown to be in quotations.

(2)

---

U.S. Government Employee Authorized Signature

---

Date

---

Name of Government Organization

---

Title (if not author)

*(Please read and sign and return Part B **only** if you are a government employee and created your article/paper as part of your employment. If your work was performed under Government contract, but you are not a Government employee, sign only at signature line (1) in Part A and see item 5 under returned rights. Authors who are U.S. government employees should also sign signature line (1) in Part A above to enable AAAI to claim and protect its copyright in international jurisdictions.)*

## PART C–CROWN COPYRIGHT CERTIFICATION

This will certify that all authors of the above article/paper are employees of the British or British Commonwealth Government and prepared the Work in connection with their official duties , and that the article/paper is therefore subject to Crown Copyright and is not assigned to AAAI as set forth in the first sentence of the Copyright Transfer Section in Part A. The undersigned warrants that they are the sole author/translator of the above article/paper, and that the article/paper is original throughout, except for those portions shown to be in quotations, and acknowledges that AAAI has the right to publish, distribute, and reprint the Work in all forms and all media.

(3)

---

British or British Commonwealth Government Employee Authorized Signature

---

Date

---

Name of Government Organization

---

Title (if not author)

*(Please read and sign and return Part C **only** if you are a British or British Commonwealth Government employee and the Work is subject to Crown Copyright. Authors who are British or British Commonwealth government employees should also sign signature line (1) in Part A above to indicate their acceptance of all terms other than the copyright transfer.)*

## You Only Need Attention to Traverse Trees

Mahtab Ahmed, Muhammad Rifayat Samee, Robert E. Mercer

### Abstract

In recent NLP research, a topic of interest is universal sentence encoding, sentence representations that can be used in any supervised task. At the word sequence level, fully attention-based models suffer from two problems: a quadratic increase in memory consumption with respect to the sentence length and an inability to capture and use syntactic information. Recursive neural nets can extract very good syntactic information by traversing a tree structure. To this end, we propose Tree Transformer, a model that captures phrase level syntax for constituency trees as well as word-level dependencies for dependency trees by doing recursive traversal only with attention. Evaluation of this model on four tasks gets noteworthy results compared to the standard transformer and LSTM-based models as well as tree-structured LSTMs. Ablation studies to find whether positional information is inherently encoded in the trees and which type of attention is suitable for doing the recursive traversal are provided.

[PDF](#)[BibTeX](#)[Search](#)[Video](#)

**Anthology ID:** P19-1030

**Volume:** [Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics](#)

**Month:** July

**Year:** 2019

**Address:** Florence, Italy

**Venue:** [ACL](#)

**SIG:** -

**Publisher:** Association for Computational Linguistics

**Note:** -

**Pages:** 316–322

**Language:** -

**URL:** <https://www.aclweb.org/anthology/P19-1030>

**DOI:** [10.18653/v1/P19-1030](https://doi.org/10.18653/v1/P19-1030)

**Bib Export formats:** [BibTeX](#) [MODS XML](#) [EndNote](#) [Copy BibTeX to Clipboard](#)

**PDF:** <https://www.aclweb.org/anthology/P19-1030.pdf>

**Video:** <https://vimeo.com/384000960>

## What is the copyright for materials in the ACL Anthology?

The ACL materials that are hosted in the Anthology are licensed to the general public under a liberal usage policy that allows unlimited reproduction, distribution and hosting of materials on any other website or medium, for non-commercial purposes. Prior to 2016, all ACL materials are licensed using the [Creative Commons 3.0 BY-NC-SA](#) (Attribution, Non-Commercial, Share-Alike) license. As of 2016, this policy has been relaxed further, and all subsequent materials are available to the general public on the terms of the [Creative Commons 4.0 BY](#) (Attribution) license; this means both commercial and non-commercial use is explicitly licensed to all.

Note that these policies only cover ACL materials. As with [the DOIs](#), this policy does not cover third-party materials. For reproduction privileges for such materials, please approach the respective organizations.



## Multilingual Corpus Creation for Multilingual Semantic Similarity Task

Mahtab Ahmed, Chahna Dixit, Robert E. Mercer, Atif Khan, Muhammad Rifayat Samee, Felipe Urra

### Abstract

In natural language processing, the performance of a semantic similarity task relies heavily on the availability of a large corpus. Various monolingual corpora are available (mainly English); but multilingual resources are very limited. In this work, we describe a semi-automated framework to create a multilingual corpus which can be used for the multilingual semantic similarity task. The similar sentence pairs are obtained by crawling bilingual websites, whereas the dissimilar sentence pairs are selected by applying topic modeling and an Open-AI GPT model on the similar sentence pairs. We focus on websites in the government, insurance, and banking domains to collect English-French and English-Spanish sentence pairs; however, this corpus creation approach can be applied to any other industry vertical provided that a bilingual website exists. We also show experimental results for multilingual semantic similarity to verify the quality of the corpus and demonstrate its usage.

[PDF](#)[BibTeX](#)[Search](#)

**Anthology ID:** 2020.lrec-1.516

**Volume:** [Proceedings of the 12th Language Resources and Evaluation Conference](#)

**Month:** May

**Year:** 2020

**Address:** Marseille, France

**Venue:** [LREC](#)

**SIG:** –

**Publisher:** European Language Resources Association

**Note:** –

**Pages:** 4190–4196

**Language:** English

**URL:** <https://www.aclweb.org/anthology/2020.lrec-1.516>

**DOI:** –

**Bib Export formats:** [BibTeX](#) [MODS XML](#) [EndNote](#) [Copy BibTeX to Clipboard](#)

**PDF:** <https://www.aclweb.org/anthology/2020.lrec-1.516.pdf>

## What is the copyright for materials in the ACL Anthology?

The ACL materials that are hosted in the Anthology are licensed to the general public under a liberal usage policy that allows unlimited reproduction, distribution and hosting of materials on any other website or medium, for non-commercial purposes. Prior to 2016, all ACL materials are licensed using the [Creative Commons 3.0 BY-NC-SA](#) (Attribution, Non-Commercial, Share-Alike) license. As of 2016, this policy has been relaxed further, and all subsequent materials are available to the general public on the terms of the [Creative Commons 4.0 BY](#) (Attribution) license; this means both commercial and non-commercial use is explicitly licensed to all.

Note that these policies only cover ACL materials. As with [the DOIs](#), this policy does not cover third-party materials. For reproduction privileges for such materials, please approach the respective organizations.

# Appendix B

## Supporting Materials

### B.1 Software Packages

1. The implementation of the paper titled “Improving Neural Sequence Labelling using Additional Linguistic Information” (Chapter 2).

Platform: Pytorch

Download link: <https://github.com/navid5792/Sequece-Labelling>

2. The implementation of the paper titled “A Novel Neural Sequence Model with Multiple Attentions for Word Sense Disambiguation” (Chapter 3).

Platform: Pytorch

Download link: [https://github.com/navid5792/wsd\\_s2s](https://github.com/navid5792/wsd_s2s)

3. The implementation of the paper titled “Improving Tree-LSTM with Tree Attention” (Chapter 4).

Platform: Pytorch

Download link: [https://github.com/navid5792/tree\\_attention/tree/master](https://github.com/navid5792/tree_attention/tree/master)

4. The implementation of the paper titled “Identifying Protein-Protein Interaction using Tree LSTM and Structured Attention” (Chapter 5).

Platform: Pytorch

Download link: <https://github.com/navid5792/ppi>

5. The implementation of the paper titled “Efficient Transformer-based Sentence Encoding for Sentence Pair Modelling” (Chapter 6).

Platform: Pytorch

Download link: <https://github.com/navid5792/SiameseUSE>



6. The implementation of the paper titled “Investigating Relational Recurrent Neural Networks with Variable Length Memory Pointer in Sentence Pair Modelling Tasks” (Chapter 7).

Platform: Pytorch

Download link: [https://github.com/navid5792/relational\\_memory\\_NN](https://github.com/navid5792/relational_memory_NN)

7. The implementation of the paper titled “You Only Need Attention to Traverse Trees” (Chapter 8).

Platform: Pytorch

Download link: <https://github.com/navid5792/Tree-Transformer>

8. The implementation of the paper titled “Encoding Dependency Information inside Tree Transformer” (Chapter 9).

Platform: Pytorch

Download link: <https://github.com/navid5792/Tree-Transformer>

9. The implementation of the paper titled “Modelling Sentence Pairs via Reinforcement Learning: An Actor-Critic Approach to Learn the Irrelevant Words” (Chapter 10).

Platform: Pytorch

Download link: [https://github.com/navid5792/SS\\_actor\\_critic](https://github.com/navid5792/SS_actor_critic)

10. The implementation of the paper titled “Learning to Compare Sentence Pairs in Phrase Level: An Actor-Critic Approach” (Chapter 11).

Platform: Pytorch

Download link: <https://github.com/navid5792/HS-LSTM-RL/tree/master>

## B.2 Datasets

1. Sense-tagged Semantic Corpus 3.0.

Download link: <https://www.kaggle.com/nltkdata/semtcor-corpus>

2. MASC: An Open Language Data Community Resource.

Download link: <http://www.anc.org/data/masc/>

3. Senseval 3.

Download link: <https://web.eecs.umich.edu/~mihalcea/senseval/senseval3/data.html>

4. Penn TreeBank (PTB) POS tagging.

Download link: <https://catalog.ldc.upenn.edu/LDC99T42>

5. CoNLL 2000 chunking.

Download link: <https://www.clips.uantwerpen.be/conll2000/chunking/>

6. CoNLL 2003 NER.

Download link: <https://www.clips.uantwerpen.be/conll2003/ner/>

7. Protein Protein Interaction problem. All the five datasets (AIMed, Bionfer, IEPA, HPRD50, and LLL) are available in the following link.

Download link: <http://mars.cs.utu.fi/PPICorpora/>

8. Sentences Involving Compositional Knowledge.

Download link: <https://github.com/brmson/dataset-sts/tree/master/data/sts/sick2014>

9. Microsoft Research Paraphrase Corpus. Abbreviated as either MSRP or MRPC.

Download link: <https://deepai.org/dataset/mrpc>

10. AI2-8th Grade Science Questions, Hypothesis Based Evaluation.

Download link: <https://github.com/brmson/dataset-sts/tree/master/data/hypev/ai2-8grade>

11. Stanford Sentiment Treebank.

Download link: <https://nlp.stanford.edu/sentiment/index.html>

# Curriculum Vitae

**Name:** Mahtab Ahmed

**Education:** University of Western Ontario  
London, ON, Canada  
Ph.D. Computer Science, May 2021

Khulna University of Engineering & Technology  
Khulna, Bangladesh  
M.Sc. Computer Science and Engineering, July 2017

Khulna University of Engineering & Technology  
Khulna, Bangladesh  
B.Sc. Computer Science and Engineering, September 2014

**Honours and Awards:** Western Graduate Research Scholarship (WGRS), 2017—2021.  
Vector Institute Postgraduate Affiliate, 2018 - 2021  
University of Western Ontario Nomination for the  
Vanier Canada Graduate Scholarship, 2019  
Mitacs Accelerate Grant, 2019 - 2021

**Relevant Experience** Artificial Intelligence Intern  
Messagepoint, Toronto, ON, Canada  
November 2018 — April 2021.

Graduate Research and Teaching Assistant  
University of Western Ontario, London, ON, Canada  
September 2017 — April 2021.

Lecturer, Department of Computer Science and Engineering  
Khulna University of Engineering & Technology,  
Khulna, Bangladesh  
April 2015 — August 2017.

**Publications: Conference Papers**

1. Mahtab Ahmed and Robert E. Mercer, “Learning to Compare Sentence Pairs in Phrase Level: An Actor-Critic Approach,” *Work Completed* 2021. (To be submitted)
2. Brydon Parker, Alik Sokolov, Mahtab Ahmed, Matt Kalebic, Sedef Akinli Kocak and Ofer Shai, “Domain Specific Fine-tuning of Denoising Sequence-to-Sequence Models for Natural Language Summarization,” *Work Completed* 2021. (To be submitted)
3. Stella Y. Wu, Mahtab Ahmed and Bo Zhao, “FinanceBERT: Sentiment Classification and Extractive Summarization on Financial Text Using BERT,” 2021. (To be Submitted)
4. Mahtab Ahmed and Robert E. Mercer, “Encoding Dependency Information inside Tree Transformer,” *34th Canadian Conference on Artificial Intelligence*, Ottawa, Canada, 2021. (Accepted)
5. Mahtab Ahmed and Robert E. Mercer, “Investigating Relational Recurrent Neural Networks with Variable Length Memory Pointer in Sentence Pair Modelling Tasks,” *33rd Canadian Conference on Artificial Intelligence*, Ottawa, Canada, 2020.
6. Mahtab Ahmed, Chahna Dixit, Robert E. Mercer, Atif Khan, Muhammad Rifayat Samee, and Felipe Urrea, “Multilingual Corpus Creation for Multilingual Semantic Similarity Tasks,” *12th Language Resources and Evaluation Conference (LREC)*, Marseille, France, 2020.
7. Mahtab Ahmed, Chahna Dixit, Robert E. Mercer, Atif Khan, Muhammad Rifayat Samee, and Felipe Urrea, “Multilingual Semantic Textual Similarity using Multilingual Word Representations,” *14th IEEE International Conference on Semantic Computing (ICSC)*, California, USA, 2020.
8. Mahtab Ahmed and Robert E. Mercer, “Modelling Sentence Pairs via Reinforcement Learning: An Actor-Critic Approach to Learn the Irrelevant Words,” *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, New York, USA, 2020.
9. Mahtab Ahmed, Muhammad Rifayat Samee and Robert E. Mercer, “You Only Need Attention to Traverse Trees,” *57th Annual Meeting of the Association for Computational Linguistics (ACL)*, Florence, Italy, 2019.
10. Mahtab Ahmed and Robert E. Mercer, “Efficient Transformer-based Sentence Encoding for Sentence Pair Modelling,” *32nd Canadian Conference on Artificial Intelligence*, May 2019. “**Best Student Paper Award**”

11. Mahtab Ahmed, Muhammad Rifayat Samee and Robert E. Mercer, “Improving Tree LSTM with Tree Attention,” *13th IEEE International Conference on Semantic Computing (ICSC)*, California, USA, Jan 2019. “**Best Paper Award**” and “**Best Student Paper Award**”.
12. Mahtab Ahmed, Jumayel Islam, Muhammad Rifayat Samee and Robert E. Mercer, “Identifying Protein-Protein Interaction using Tree LSTM and Structured Attention,” *13th IEEE International Conference on Semantic Computing (ICSC)*, California, USA, Jan 2019.
13. Mahtab Ahmed, Muhammad Rifayat Samee and Robert E. Mercer, “Improving Neural Sequence Labelling using Additional Linguistic Information,” *17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Florida, USA, Dec 2018.
14. Mahtab Ahmed, Muhammad Rifayat Samee and Robert E. Mercer, “A Novel Neural Sequence Model with Multiple Attentions for Word Sense Disambiguation,” *17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Florida, USA, Dec 2018.
15. Sudipta Singha Roy, Mahtab Ahmed and M. A. H. Akhand, “Classification of Massive Noisy Image Using Auto-encoders and Convolutional Neural Network,” in *8th International Conference on Information Technology (ICIT)*, Jordan, 2017.
16. Tanzim Mahmud, K. M. Azharul Hasan, Mahtab Ahmed and Thwoi Hla Ching Chak, “A Rule Based Approach for NLP Based Query Processing,” in *2nd International Conference on Electrical Information and Communication technology (EICT 2015)*, Khulna, Bangladesh, pp.78-82, Dec 2015.

#### **Publications: Journal Papers**

1. Mahtab Ahmed, Chahna Dixit, Robert E. Mercer, and Atif Khan “A Model for Multilingual Semantic Textual Similarity using a Business Enterprise-Oriented Corpus,” *Natural Language Processing Research, Special Issue on Multilingual Representations for NLP*, 2021. (Submitted)
2. M. A. H. Akhand, Mahtab Ahmed and M. M. Hafizur Rahman, “Convolutional Neural Network Training Incorporating Rotation Based Patterns and Handwritten Numeral Recognition of Major Indian Scripts,” *IETE Journal of Research*, vol. 64, no. 2, pp. 176–194, Jul 2017.

3. Sudipta Singha Roy, Mahtab Ahmed and M. A. H. Akhand, “Noisy Image Classification Using Hybrid Deep Learning Methods,” *Journal of Information and Communication Technology (JICT)*, vol. 12, no. 4, pp. 330–352, Dec 2017.
4. M. A. H. Akhand, Mahtab Ahmed and M. M. Hafizur Rahman, “Convolutional Neural Network based Handwritten Bengali and Bengali-English Mixed Numeral Recognition,” *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, vol. 8, no. 9, pp. 40–50, Sep 2016.
5. Mahtab Ahmed, Pintu Chnadra Shill, Kaidul Islam and M. A. H. AKhand, “Acoustic Modeling of Bangla Words using Deep Belief Network,” *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, vol. 7, no. 10, pp. 19–27, Sep 2015.