

Neural Architectures for Natural Language Understanding

Yi Tay

School of Computer Science & Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2019

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

6 June 2019

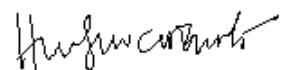
Date



Yi Tay

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.



6 June 2019

.....
Date

.....
Assoc Prof Hui Siu Cheung

Authorship Attribution Statement

This thesis contains material from **10** paper(s) published in the following peer-reviewed conferences) in which I am listed as an author.

Chapter 3 is published with material from the following papers: (1) Yi Tay, Luu Anh Tuan, Siu Cheung Hui - Compare, Compress and Propagate: Enhancing Neural Architectures with Alignment Factorization for Natural Language Inference, *Proceedings of the 2018 conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, Pages 1565-1575, Brussels, Belgium.

AND

(2) Yi Tay, Luu Anh Tuan, Siu Cheung Hui - Co-Stack Residual Affinity Networks with Multi-level Attention Refinement for Matching Text Sequences, *Proceedings of the 2018 conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, Pages 4492-4502, Brussels, Belgium.

The contributions of the co-authors are as follows:

- I came up with the key idea, designed all experiments, implemented all of the source code and conducted all experiments. I prepared the manuscript drafts (in entirety).
- The manuscripts were revised and edited mainly by A/P Hui Siu Cheung and Dr Luu Anh Tuan.

Chapter 4 is published with material from:

(3) Yi Tay, Luu Anh Tuan, Siu Cheung Hui - Multi-Cast Attention Networks, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2018)*, Pages 2299-2308, London, UK.

The contributions of the co-authors are as follows:

- I came up with the key idea, designed all experiments, implemented all of the source code and conducted all experiments. I prepared the manuscript drafts (in entirety).
- The manuscripts were revised and edited mainly by A/P Hui Siu Cheung and Dr Luu Anh Tuan.

Chapter 5 is published with material from:

- (4) Yi Tay, Luu Anh Tuan, Siu Cheung Hui, Jian Su - Densely Connected Attention Propagation for Reading Comprehension, *Advances in Neural Information Processing Systems (NeurIPS 2018)*, Pages 4906-4917, Montreal, Canada.

The contributions of the co-authors are as follows:

- I came up with the key idea, designed all experiments, implemented all of the source code and conducted all experiments. I prepared the manuscript drafts (in entirety).
- The manuscripts were revised and edited mainly by A/P Hui Siu Cheung, Dr Luu Anh Tuan and Dr Jian Su.

Chapter 6 is published with material from:

- (5) Yi Tay, Shuohang Wang, Anh Tuan Luu, Jie Fu, Minh C. Phan, Xingdi Yuan, Jinfeng Rao, Siu Cheung Hui, Aston Zhang - Simple and Effective Curriculum Pointer-Generator Networks for Reading Comprehension over Long Narratives - *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL 2019)*, Accepted, Florence, Italy.

The contributions of the co-authors are as follows:

- I came up with the key idea, designed all experiments, implemented all of the source code and conducted experiments. I prepared the manuscript drafts.
- Dr Shuohang Wang provided experimental results for his own R3 model as a competitor.
- Dr Jie Fu, Dr Jinfeng Rao, Minh C. Phan and Xingdi Yuan provided feedback about the initial idea and helped with editing the manuscript.
- The final revisions were mainly edited by A/P Hui Siu Cheung, Dr Luu Anh Tuan and Dr Aston Zhang.

Chapter 7 is published with material from:

- (10) Yi Tay, Luu Anh Tuan, Siu Cheung Hui - Multi-Pointer Co-Attention Networks for Recommendation, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2018)*, Pages 2309-2318, London, UK.

The contributions of the co-authors are as follows:

- I came up with the key idea, designed all experiments, implemented all of the source code and conducted all experiments. I prepared the manuscript drafts (in entirety).

-
- The manuscripts were revised and edited mainly by A/P Hui Siu Cheung and Dr Luu Anh Tuan.

Chapter 8 is published with material from:

(6) Yi Tay, Luu Anh Tuan, Siu Cheung Hui - Multi-Granular Sequence Encoding via Dilated Composition Units for Reading Comprehension, *Proceedings of the 2018 conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, Pages 2141-2151, Brussels, Belgium.

The contributions of the co-authors are as follows:

- I came up with the key idea, designed all experiments, implemented all of the source code and conducted all experiments. I prepared the manuscript drafts (in entirety).
- The manuscripts were revised and edited mainly by A/P Hui Siu Cheung and Dr Luu Anh Tuan.

Chapter 9 is published with material from:

(7) Yi Tay, Luu Anh Tuan, Siu Cheung Hui - Recurrently Controlled Recurrent Networks - *Advances in Neural Information Processing Systems (NeurIPS 2018)*, Pages 4731-4743, Montreal, Canada.

The contributions of the co-authors are as follows:

- I came up with the key idea, designed all experiments, implemented all of the source code and conducted all experiments. I prepared the manuscript drafts (in entirety).
- The manuscripts were revised and edited mainly by A/P Hui Siu Cheung and Dr Luu Anh Tuan.

Chapter 10 is published with material from:

(8) Yi Tay, Luu Anh Tuan, Siu Cheung Hui - Hyperbolic Representation Learning for Fast and Efficient Neural Question Answering, *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM 2018)*, Pages 583-591, Los Angeles, California.

The contributions of the co-authors are as follows:

- I came up with the key idea, designed all experiments, implemented all of the source code and conducted all experiments. I prepared the manuscript drafts (in entirety).

- The manuscripts were revised and edited mainly by A/P Hui Siu Cheung and Dr Luu Anh Tuan.

Chapter 11 is published with material from:

(9) Yi Tay, Aston Zhang, Anh Tuan Luu, Jinfeng Rao, Shuai Zhang, Shuohang Wang, Jie Fu and Siu Cheung Hui - Lightweight and Efficient Neural Natural Language Processing with Quaternion Networks - *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL 2019)*, Accepted, Florence, Italy.

The contributions of the co-authors are as follows:

- I came up with the key idea, designed all experiments, implemented all of the source code and conducted experiments. I prepared the manuscript drafts.
- Dr Aston Zhang did a comprehensive proof read on the mathematics. Shuai Zhang was involved in early discussions about Quaternion representations.
- Dr Jie Fu, Dr Jinfeng Rao and Dr Shuohang Wang provided feedback about the initial idea and helped with editing the manuscript.
- The final revisions were mainly edited by A/P Hui Siu Cheung, Dr Luu Anh Tuan and Dr Aston Zhang.

6 June 2019

.....
Date



.....
Yi Tay

Acknowledgements

I wish to express gratitude to my main advisor Prof Hui Siu Cheung (NTU) for the kind support and mentorship. I would also like to express gratitude to my A*Star supervisors Dr Luu Anh Tuan and Dr Jian Su for their guidance throughout the doctoral program.

I wish to express gratitude to my collaborators, colleagues, co-authors and friends who have made this research journey extremely fruitful and enjoyable. They are listed as follows: Minh C. Phan (NTU), Alvin Chan (NTU), Tung Nguyen Thomas (NTU), Yiding Liu (NTU), Chaoyue He (NTU), Lucas Vinh Tran (NTU), Daniel Rugeles (NTU), Shuai Zhang (UNSW), Dr Jie Fu (MILA), Dr Shuohang Wang (SMU), Dr Doyen Sahoo (SMU), Dr Aston Zhang (Amazon AI), Dr Jiatao Gu (Facebook AI Research), Qi Liu (Facebook AI Research), Dr Jinfeng Rao (Facebook Conversational AI) and Xingdi Yuan (Microsoft Research). I have definitely grown as a researcher, thanks to all of you.

I would also like to express gratitude to people who have given me their gracious support and guidance during my research journey: Prof Ong Yew Soon (NTU), Prof Jiang Jing (SMU), Prof Wei Lu (SUTD), Dr Nancy Chen (A*Star) and Dr Zhe Zhao (Google AI). I would also like to express gratitude to my TAC members Prof Zhang Jie (NTU) and Prof Hady Lauw (SMU) for their guidance.

Finally, I would like to thank my parents Tay Tiong Choon and Sabrina Yong Chi who have taken care of me all these years. I also thank my brothers Tay Ye, Tay Yang and Tay Yong for being such a wonderful family. Most importantly, I express my deepest gratitude to my wife Vanessa Chew for all the love.

“Part of the journey is the end”

—Tony Stark, Ironman

To my dear family

Abstract

Empowering machines with the ability to read and reason lives at the heart of Artificial Intelligence (AI) research. Language is ubiquitous, serving as a key communication mechanism that is woven tightly into the fabric of society and humanity. The pervasiveness of textual content is made evident by the billions of documents, social posts, and messages on the web. As such, the ability to make sense, reason and understand textual content has immense potential to benefit a large range of real-world applications such as search, question answering, recommender systems, and/or personal chat assistants.

This thesis tackles the problem of natural language understanding (NLU) and in particular, problem domains that fall under the umbrella of NLU, e.g., question answering, machine reading comprehension, natural language inference, retrieval-based NLU, etc. More specifically, we study machine learning models (in particular, neural architectures), for solving a suite of NLU problems. The key goal is to enable machines to be able to read and comprehend natural language.

We make several novel contributions in this thesis, mainly revolving around the design of neural architectures for NLU problems. The key contributions are listed as follows:

- We propose two new state-of-the-art neural models for natural language inference: ComProp Alignment-Factorized Encoders (CAFE) and Co-Stack Residual Affinity Networks (CSRAN). On the single model setting, CAFE and CSRAN achieves 88.5% accuracy and 88.7% accuracy respectively on the well-studied SNLI benchmark.
- We propose Multi-Cast Attention Networks (MCAN) for retrieval-based NLU. On Ubuntu dialogue corpus, MCAN outperforms the existing state-of-the-art models by 9%. MCAN also achieves the best performing score of 0.838 MAP and 0.904 MRR on the well-studied TrecQA dataset.

- We propose Densely Connected Attention Propagation (DecaProp), a new model designed for machine reading comprehension (MRC) on the web. We achieve state-of-the-art performance on reading tests on news and Wikipedia articles. DecaProp achieves 2.6% – 14.2% absolute improvement in F1 score over the existing state-of-the-art on four challenging MRC datasets.
- We propose Introspective Alignment Reader and Curriculum Pointer-Generator (IAL-CPG) model for reading and understanding long narratives. IAL-CPG achieves state-of-the-art performance on the NarrativeQA reading comprehension challenge. On metrics such as BLEU-4 and Rouge-L, we achieve a 17% relative improvement over prior state-of-the-art and a 10 times improvement in terms of BLEU-4 score over BiDAF, a strong span prediction based model.
- We propose Multi-Pointer Co-Attention Networks (MPCN) for recommendation with reviews. On Amazon Reviews dataset, MPCN improves the existing state-of-the-art DeepCoNN and D-ATT model by up to 71% and 5% respectively in terms of relative improvement.
- Moreover, we propose two novel general-purpose encoding units for sequence encoding for natural language understanding: Dilated Compositional Units (DCU) and Recurrently Controlled Recurrent Networks (RCRN). DCU achieves state-of-the-art on the RACE dataset, demonstrating improvement over LSTM/GRU encoders by 6%. On the other hand, RCRN outperforms stacked BiLSTMs and BiLSTMs across 26 NLP/NLU datasets.
- Finally, we propose two novel techniques for efficient training and inference of NLU models: HyperQA (Hyperbolic NLU) and Quaternion Attention/Quaternion Transformer Models. HyperQA outperforms strong attention and recurrent baselines while being extremely lightweight (40K to 90K parameters). On the other hand, Quaternion Attention/Quaternion Transformers enable up to 75% parameter reduction while maintaining competitive performance.

Contents

Acknowledgements	vii
Abstract	ix
List of Figures	xvii
List of Tables	xviii
1 Introduction	1
1.1 Motivation	1
1.2 Research Objectives	3
1.3 Problem Overview and Research Scope	4
1.3.1 Natural Language Inference (NLI)	4
1.3.2 Question Answering and Machine Reading Comprehension	6
1.3.3 Retrieval-based Natural Language Understanding	6
1.3.4 Natural Language Understanding for Recommender Systems	7
1.3.5 Neural Building Blocks for Natural Language Understanding	7
1.4 Major Contributions	7
1.4.1 Factorized Neural Attention Models for Natural Language Inference	8
1.4.2 Multi-Cast Attention Networks for Retrieval-based Natural Language Understanding	8
1.4.3 Densely Connected Attention Propagation for Machine Reading Comprehension	9
1.4.4 Introspective Curriculum Pointer-Generator for Machine Reading Comprehension over Long Narratives	9
1.4.5 Natural Language Understanding for Recommender Systems	10
1.4.6 Multi-Granular Sequence Encoders for Natural Language Understanding	10
1.4.7 Recurrently Controlled Recurrent Networks for Natural Language Understanding	11
1.4.8 Hyperbolic Representations for Natural Language Understanding	11

1.4.9	Quaternion Representations for Natural Language Understanding	11
1.5	Organization of the Thesis	12
2	Literature Review	15
2.1	Natural Language Inference	15
2.1.1	Related Work	16
2.1.2	Benchmarks	17
2.2	Retrieval-based Natural Language Understanding	17
2.2.1	Related Work	18
2.2.2	Benchmarks	19
2.3	Machine Reading Comprehension	19
2.3.1	Related Work	19
2.3.2	Benchmarks	20
2.4	Text-based Recommender Systems	21
2.4.1	Related Work	21
2.4.2	Benchmarks	22
2.5	Neural Building Blocks for NLU	23
2.5.1	Recurrent Neural Networks	23
2.5.2	Neural Attention	25
2.6	Summary	27
3	Factorized Neural Attention Models for Natural Language Inference	28
3.1	Introduction	28
3.2	Proposed Method (CAFE)	31
3.2.1	Input Encoding Layer	32
3.2.2	Soft-Attention Alignment Layer	32
3.2.3	Intra-Attention Alignment Layer	33
3.2.4	Alignment Factorization Layer	33
3.2.5	Inter-Alignment Factorization	34
3.2.6	Intra-Alignment Factorization	34
3.2.7	Propagation and Augmentation	35
3.2.8	Sequential Encoder Layer	35
3.2.9	Pooling Layer	36
3.2.10	Prediction Layer	36
3.2.11	Additional Discussion	36
3.3	Proposed Method (CSRAN)	37
3.3.1	Input Encoder	38
3.3.2	Stacked Recurrent Encoders	38
3.3.3	Multi-level Attention Refinement (MAR)	39
3.3.4	Co-Stacking	39
3.3.5	Bidirectional Alignment	40
3.3.6	Matching and Aggregation Layer	40

3.3.7	Output and Prediction Layer	41
3.4	Experiments (CAFE)	41
3.4.1	Experimental Setup	41
3.4.2	Implementation Details	42
3.4.3	Experimental Results	42
3.4.4	Ablation Study	45
3.4.5	Linguistic Error Analysis	46
3.4.6	Interpreting and Visualizing with CAFE	46
3.5	Experiments (CSRAN)	48
3.5.1	Experimental Setup	48
3.5.2	Experimental Results	49
3.6	Summary	50
4	Multi-Cast Attention Networks for Retrieval-based Natural Language Understanding	51
4.1	Introduction	51
4.2	Proposed Method	54
4.2.1	Input Encoder	54
4.2.2	Co-Attention	55
4.2.3	Multi-Cast Attention	57
4.2.4	Multi-Cast	59
4.2.5	LSTM Encoder	59
4.2.6	Pooling Operation	60
4.2.7	Prediction Layer and Optimization	60
4.2.8	Relation to CAFE	61
4.3	Experiments	61
4.3.1	Dialogue Prediction	61
4.3.2	Factoid Question Answering	63
4.3.3	Community Question Answering (cQA)	64
4.3.4	Tweet Reply Prediction	65
4.3.5	Ablation Analysis	67
4.3.6	In-depth Model Analysis	68
4.4	Summary	71
5	Densely Connected Attention Propagation for Machine Reading Comprehension	73
5.1	Introduction	73
5.2	Proposed Method	76
5.2.1	Bidirectional Attention Connectors (BAC)	76
5.2.2	Densely Connected Attention Propagation (DecaProp)	78
5.3	Experiments	81
5.3.1	Datasets and Competitor Baselines	82
5.3.2	Experimental Setup	83
5.3.3	Performance Results	83

5.3.4	Ablation Study	86
5.4	Summary	87
6	Introspective Curriculum Pointer-Generator for Machine Reading Comprehension over Long Narratives	89
6.1	Introduction	90
6.2	Proposed Method	91
6.2.1	Introspective Alignment Reader	92
6.2.2	Pointer-Generator Decoder	94
6.2.3	Curriculum Reading	96
6.3	Experiments	97
6.3.1	Experimental Setup	98
6.3.2	Experimental Results	99
6.3.3	Ablation Study	100
6.3.4	Qualitative Error Analysis	102
6.4	Summary	103
7	Natural Language Understanding for Recommender Systems	105
7.1	Introduction	105
7.2	Proposed Method	108
7.2.1	Input Encoding	108
7.2.2	Review-level Co-Attention	110
7.2.3	Review Pointers	111
7.2.4	Word-level Co-Attention	112
7.2.5	Multi-Pointer Learning	113
7.2.6	Prediction Layer	114
7.3	Experiments	115
7.3.1	Datasets	115
7.3.2	Compared Methods	116
7.3.3	Experimental Setup	117
7.3.4	Experimental Results	118
7.3.5	Hyperparameter & Ablation Analysis	119
7.3.6	In-Depth Model Analysis	122
7.4	Summary	124
8	Multi-Granular Sequence Encoders for Natural Language Understanding	126
8.1	Introduction	126
8.2	Proposed Method	129
8.2.1	Dilated Compositional Units (DCU)	129
8.2.2	Overall Model Architectures	133
8.3	Experiments	135
8.3.1	Datasets	135
8.3.2	Compared Methods	136

8.3.3	Implementation Details	137
8.3.4	Experimental Results on RACE	138
8.3.5	Experimental Results on SearchQA	139
8.3.6	Experimental Results on NarrativeQA	140
8.4	Summary	141
9	Recurrently Controlled Recurrent Networks for Natural Language Understanding	142
9.1	Introduction	142
9.2	Proposed Method	145
9.2.1	Controller Cell	145
9.2.2	Listener Cell	146
9.2.3	Overall RCRN Architecture, Variants and Implementation .	147
9.3	Experiments	148
9.3.1	Tasks and Datasets	148
9.3.2	Task-Specific Model Architectures and Implementation Details	149
9.3.3	Performance Results	151
9.3.4	Runtime Analysis	156
9.4	Summary	157
10	Hyperbolic Representations for Natural Language Understanding	158
10.1	Introduction	158
10.2	Proposed Method	161
10.2.1	Embedding Layer	162
10.2.2	Projection Layer	162
10.2.3	Learning QA Representations	162
10.2.4	Hyperbolic Representations of QA Pairs	163
10.2.5	Final Transform	164
10.2.6	Optimization and Learning	164
10.3	Experiments	165
10.3.1	Datasets	165
10.3.2	Compared Baselines	166
10.3.3	Evaluation Protocol	168
10.3.4	Results and Analysis	169
10.3.5	Effects of QA Embedding Size	172
10.3.6	Discussion and Analysis	173
10.3.7	Analysis of QA Embeddings	173
10.3.8	Analysis of Word Embeddings	174
10.4	Summary	176
11	Quaternion Representations for Natural Language Understanding	178
11.1	Introduction	178
11.2	Proposed Method	180
11.2.1	Background on Quaternion Algebra	180

11.2.2 Quaternion Models of Language	182
11.2.3 Quaternion Transformer	185
11.2.4 Embedding Layers	186
11.2.5 Connection to Real Components	186
11.3 Experiments	187
11.3.1 Pairwise Natural Language Understanding	187
11.3.2 Sentiment Analysis	189
11.3.3 Neural Machine Translation	190
11.3.4 Mathematical Language Understanding	191
11.3.5 Subject Verb Agreement	192
11.4 Summary	193
12 Conclusion	194
12.1 Overall Summary	194
12.2 Future Directions and Challenges	196
12.2.1 Multi-document Reasoning and Understanding	196
12.2.2 Conversational Language Understanding	196
12.2.3 Efficient Models for Reading Long Documents	197
12.2.4 Language Understanding for Fact Verification	197
Bibliography	198

List of Figures

1.1	Hierarchy of Proposed Contributions.	12
3.1	Architecture of the CAFE model.	31
3.2	Architecture of CSRAN.	38
3.3	Visualization of <i>six</i> propagated features.	47
4.1	Architecture of Multi-Cast Attention Networks.	54
4.2	Casted Attention Features on a <i>positive</i> sample.	68
4.3	Casted Attention Features on a <i>negative</i> sample.	69
4.4	Differences between Max and Mean-pooled Casted Attention.	71
5.1	Architecture of BAC.	76
5.2	Architecture of DecaProp.	78
5.3	DecaProp versus R-NET on NewsQA.	87
6.1	Architecture of IAL-CPG.	92
7.1	Architecture of MPCN.	109
7.2	Neural Network and Additive based Multi-Pointer Learning.	114
7.3	Visualization of Review-level Co-Attention.	123
8.1	Architecture of DCU.	130
8.2	Overview of the Fold-Unfold operation for $r_j = 2$	131
9.1	Architecture of RCNN.	145
10.1	Illustration of hyperbolic space.	160
10.2	Architecture of HyperQA.	161
10.3	Effects of QA embedding size on WikiQA.	173
10.4	Embeddings from HYPERQA.	174
10.5	Embeddings from CosineQA.	174
10.6	Histogram plots of embedding norms.	175
11.1	Quaternion weight sharing.	183

List of Tables

1.1	Overview of contributions.	12
2.1	Statistics of datasets.	16
3.1	Results (CAFE) on the SNLI dataset.	43
3.2	Results (CAFE) on MultiNLI and SciTail datasets.	44
3.3	Ablation study on MultiNLI development sets.	45
3.4	Linguistic Error Analysis on MultiNLI dataset.	46
3.5	Results (CSRAN) on single model SNLI dataset.	49
3.6	Results (CSRAN) on SciTail dataset.	49
3.7	Results (CSRAN) on Quora NLI dataset.	49
4.1	Results on Ubuntu Dialogue Corpus.	62
4.2	Results on TrecQA (<i>clean</i>) dataset.	64
4.3	Results on QatarLiving dataset.	65
4.4	Results on Tweets dataset.	66
4.5	Ablation analysis (validation set) on TrecQA dataset.	67
5.1	Results on NewsQA dataset.	84
5.2	Results on Quasar-T dataset.	84
5.3	Results on SearchQA (original) dataset.	85
5.4	Results on SearchQA dataset.	85
5.5	Results on NarrativeQA (Story Summaries) dataset.	85
5.6	Results on SQuAD (Dev Set) dataset.	86
5.7	Ablation study on NewsQA development set.	87
6.1	Results on NarrativeQA (Full) reading comprehension dataset.	100
6.2	Ablation results on NarrativeQA development set.	101
6.3	Qualitative analysis on NarrativeQA development set.	103
7.1	Results on benchmark datasets.	118
7.2	Ablation analysis on Amazon Product Reviews dataset.	120
7.3	Validation MSE when varying the number of pointers.	121
7.4	Excerpts from the MPCN’s pointer mechanism.	122
7.5	Analysis of multi-pointer behavior of MPCN.	124
8.1	Results on RACE dataset.	138

8.2	Results on SearchQA dataset.	139
8.3	Results on NarrativeQA dataset.	140
9.1	Results on Amazon Reviews dataset for sentiment analysis.	151
9.2	Results on SST-5 dataset for sentiment analysis.	152
9.3	Results on SST-2 dataset for sentiment analysis.	152
9.4	Results on IMDb dataset for sentiment analysis.	152
9.5	Results on TREC dataset for question classification.	153
9.6	Results on SNLI dataset for entailment classification.	154
9.7	Results on SciTail dataset for entailment classification.	154
9.8	Results on WikiQA and TrecQA datasets for answer retrieval.	155
9.9	Results on NarrativeQA dataset for machine reading comprehension	156
9.10	Training/Inference times on IMDb dataset.	156
10.1	Statistics of datasets.	166
10.2	Results on TrecQA (raw) dataset.	170
10.3	Results on TrecQA (clean) dataset.	170
10.4	Results on WikiQA dataset	170
10.5	Results on YahooCQA dataset.	171
10.6	Results on SemEvalCQA dataset.	171
10.7	Overall performance of HYPERQA.	172
10.8	Examples of words in each hierarchical level.	176
10.9	Analysis of QA pairs with respect to hierarchy.	176
11.1	Results for pairwise NLU tasks.	189
11.2	Results for sentiment analysis.	189
11.3	Results for neural machine translation.	190
11.4	Results for mathematical language understanding (MLU).	192
11.5	Results for subject-verb agreement (SVA).	192

Chapter 1

Introduction

The ability for machines to understand and reason with language lives at the heart of Artificial Intelligence (AI) research. After all, language not only plays a pivotal role in our daily lives but is also central to many forms of human communication. Language is ubiquitous, residing in documents, the world wide web, chat messages and/or social media. To this end, enabling machines to understand and reason with language has the potential to benefit a wide spectrum of these applications, e.g., question answering [1], search [2, 3], personal assistants and/or recommender systems [4]. This field, known as *natural language understanding* (NLU), involves teaching machines to read and comprehend [5]. It is well-established that incorporating a suitable inductive (architectural) bias goes a long way in building NLU systems. As such, this thesis tackles the problem of designing effective and efficient neural network architectures for building natural language understanding systems.

1.1 Motivation

Language understanding is indeed an extremely challenging problem. Language is highly complex and nuanced. For example, understanding intricacies in user queries alone might already be extremely difficult, let alone understanding entire documents or chat logs. For many NLU applications, there is also a critical need to scaling beyond surface-level understanding [5, 6]. Past decades of NLU research has primarily focused on either feature-based machine learning [7, 8] or structure-based representations [9, 10]. However, both of which not only do not perform well, but

are also generally limited in their understanding capabilities. Today, many modern NLU systems (e.g., QA, personal assistants) are largely based on neural network approaches [11–15].

The beauty of the neural paradigm is multi-fold. Firstly, it is common to completely dispense with handcrafted features, relying on distributed representations pre-trained in an unsupervised manner from large corpora (e.g, word embeddings [16] or pretrained language models [15, 17]). Secondly, a multitude of neural components such as sequence encoders also provide a highly effective inductive bias for modeling the sequential nature of language [18–20]. Thirdly, recent advances such as attention modules [21–25] facilitate the encoding of reasoning-inspired inductive biases within networks. Lastly, these neural components are generally modular, universally applicable and general purpose. In short, they have the potential for impact beyond the originally intended application. To this end, the research climate today is often not constrained to a single application but actually considers the versatility of these neural components. Conversely, in the pre-neural era, many methods had to be specially tailored to a single application use case.

As such, the design of these core neural building blocks (encoders and/or reasoning modules) becomes fundamental to NLU research. The design of efficient, highly performant end-to-end neural architectures, blocks, components, and inductive biases for natural language understanding forms the central theme of this thesis.

A large driving force behind NLU research stems from industrial demands [26–28]. A wide spectrum of real-world applications enjoy benefits from improvements in NLU technology. It is easy to enumerate a few key applications, e.g., document search [29, 30], question answering [25] and/or dialogue systems [11, 31, 32].

As a prime example, the development of advanced NLU systems is central to the development of advanced search and information retrieval (IR) systems. Traditionally, many IR systems are based on surface-level features and as a result, face inherent difficulties with bridging the lexical gap [33]. At this juncture, NLU enhanced IR systems are able to better cope with both modeling user queries (or questions) and documents. This impact spans across a myriad of potential applications such as web search, question answering systems [5, 34], conversational systems [11, 35], social media search [36], etc.

At a deeper level, NLU has closed links to allied problem domains such as commonsense reasoning [37], semantics and knowledge representation [38–40]. To this end, many commonsense reasoning problems have been posed as inference for NLU problems [41, 42]. It is often difficult to distinguish between these problem areas given that they often require overlapping capabilities, i.e., in order to understand language, one requires some levels of basic commonsense. Any application that requires reasoning and understanding to text can benefit from NLU research.

From a philosophical point of view, the domain of NLU is also highly related to the core principle of intelligence, i.e., NLU is an intuitive prerequisite for Artificial General Intelligence (AGI) to manifest. Moreover, the ability for machines to provide explanations for predictions is also tightly coupled to NLU capabilities since a machine needs to sufficiently understand language in order to express itself. In short, language understanding is as fundamental to machines as it is to humans.

The use of neural architectures for NLU (i.e., deep learning for NLU) is a recent and emerging research area. Works in the recent years have seen progress in challenging machines to reading comprehension tests [1, 5], solving standardized tests [43] or commonsense inference [41, 42]. While still in its infancy, the study of these building blocks for reasoning with language is fundamental to progress in this field. Despite making reasonably exciting progress in recent years, we (the community) are still working things out. As such, this is an exciting time and motivating factor for pursuing NLU research.

1.2 Research Objectives

The main objectives of this thesis are outlined as follows:

- Advance the state-of-the-art across natural language understanding tasks such as natural language inference, question answering and machine reading comprehension, retrieval-based NLU and NLU-based recommender systems.
- Propose effective and efficient novel neural models and/or general universal building blocks for neural NLU systems.

1.3 Problem Overview and Research Scope

In this thesis, we consider several angles of attack against the challenging problems of natural language understanding (NLU).

In general, Natural language Inference (NLI) [44, 45], Machine Reading Comprehension (MRC) and Question Answering [1, 46, 47] are generally regarded as core NLU tasks since they *explicitly* probe for understanding and reasoning. As such, they are commonly adopted as test beds for evaluating the performance and progress of designing NLU models. In MRC systems, machines are required to read a paragraph and answer a question. This is akin to how the education system tests for a student’s understanding in comprehension examinations.

In lieu of the modular, universally applicable nature of the research climate today, this thesis also aims to study fundamental building blocks that are commonly used across many NLU and NLP systems (e.g, recurrent encoders). To this end, the design of neural building blocks (sequence encoders and/or attention modules) has immense potential to benefit many NLU-related tasks. Aside from building blocks, we also consider new inductive biases for training NLU systems, e.g., non-Euclidean (hyperbolic) spaces and/or Quaternion (hypercomplex) representations for efficient learning of NLU models. With this regard, to ascertain the effectiveness of these general-purpose components, we occasionally conduct additional experiments on allied problem domains such as sentiment analysis and/or machine translation.

Intuitively, the benefits and impact of NLU systems and modules are not limited to traditional NLU problems. In this thesis, we also consider NLU inspired applications for leveraging vast amounts of textual data on the web for building and enhancing recommender systems, bridging the field of NLP and data mining.

The following subsections delve into the focus areas of this thesis.

1.3.1 Natural Language Inference (NLI)

This task may also be referred to as recognizing textual entailment (RTE) [44] and aims to understand the relation between two sentences. The goal of this task is to determine the relationship between two sentences (premise and hypothesis). In most cases, this is framed as a multi-class classification problem, predicting either

entail, contradict or neutral relations between two sentences. For example, three examples of these relations are described as follows:

- **Entailment:**

- Premise: The man is swimming in the pool with his friends.
 - Hypothesis: The man is wet.

- **Contradiction:**

- Premise: The man is swimming in the pool with his friends.
 - Hypothesis: The man is reading a book.

- **Neutral:**

- Premise: The man is swimming in the pool with his friends.
 - Hypothesis: It is a sunny day.

In the first example, it is clear that if the man is swimming in the pool, he *must* be wet. This is why the hypothesis entails the premise. In the second example, it is generally impossible to be reading a book while swimming. Therefore, both statements contradict each other. In the third case, while it might be plausible that the man is swimming *because* it is sunny, we are not able to infer this directly. These examples are often classified as neutral because they are not directly entailed or contradicted. While examples may be subjective, NLI corpora is often curated with inter-annotator agreement in mind and by taking the majority vote.

While NLI may seem not to have any *direct* applications, it is generally used as natural language understanding (NLU) benchmark, acting as a test bed for complex language understanding. Furthermore, trained NLI models can be typically deployed to support search systems or fact verification systems, for example, determining if an extracted answer makes sense. Nevertheless, one major key goal of NLI is to push for models that are capable of understanding the rich nuances of language.

1.3.2 Question Answering and Machine Reading Comprehension

Question Answering (QA) is a well-established line of research, owing largely to its wide adoption across many applications. QA is often linked to language understanding in the sense that machines have to understand in order to answer questions. This is also commonly known as *machine comprehension* or *reading comprehension*, and is used interchangeably throughout this thesis. In this task, the input format is a query and a context document (given). The goal is to find the correct answer within the given context document. The task is coined as *comprehension* due to the fact that it allows evaluation of a machine’s ability to read and understand documents.

In many standard MRC problems [1, 47, 48], this task is concerned with extracting the relevant answer from the provided document. However, there has been recent work that pushes for generative QA systems [49], requiring answers to be generated instead. MRC systems are highly sought after in industrial search applications, providing fine-grained language understanding capabilities to standard information retrieval applications.

1.3.3 Retrieval-based Natural Language Understanding

In reality, the entire QA/MRC application pipeline requires documents to be *retrieved* and then read by NLU systems. Within the context of MRC systems, this is often referred to as open domain QA. To this end, open domain question answering and several challenging benchmarks such as SearchQA [48] and TriviaQA [47] are focused on these settings.

Prior to the development of MRC-based QA systems, retrieval-based QA systems are commonplace [50–52]. In fact, the MRC paradigm is a recent advance made possible by pointer network-based neural architectures [53, 54]. Past years of neural NLU research have typically considered the retrieval setting of ranking answers given questions [50] or ranking responses given messages [55]. Many applications fall under the retrieval-based NLU framework, e.g., question answering (question-answer), dialogue prediction (message-reply) [56] and social media search (tweet-reply) [36].

The study of powerful retrieval models is also essential to the entire NLU vision and pipeline.

1.3.4 Natural Language Understanding for Recommender Systems

While QA is an obvious use case of NLU systems, this thesis also considers the compelling use case of using NLU systems (on the web) for recommending items to users. Across a wide number of web applications (e.g., Yelp, Amazon, etc.), there exists reviews and user-generated content that users may contribute. By leveraging neural modules built for general NLU (NLI/QA) applications, this taps into the potential of vast amounts of textual data found on the web.

1.3.5 Neural Building Blocks for Natural Language Understanding

Neural architectures for NLU are typically composed of sub-modules that perform a range of operations such as reasoning, sequence encoding, or prediction. For example, recurrent [18, 19], convolutional [57, 58] or attention modules [24] are indispensable in the deep learning for NLP today. As such, research and innovation at this level can potentially have a positive impact across many NLU applications. To this end, the development of reasoning and/or encoding modules is also a focal point of this thesis.

1.4 Major Contributions

This section describes the key contributions of this thesis.

1.4.1 Factorized Neural Attention Models for Natural Language Inference

We study the natural language inference (NLI) problem, a core task in natural language understanding. We proposed a new efficient attention layer based on alignment factorization and an overall lightweight model which we call CAFE (ComProp Alignment Factorized Encoders [59]). CAFE achieves state-of-the-art performance on the Stanford Natural Language Inference (SNLI) dataset with far fewer parameters compared to competitors. The key idea here is the development of an efficient model that models the low-rank structure of alignment between sentence-pairs (e.g., alignment factorization). In addition, we also propose a deeper (hierarchical) extension of CAFE, CSRAN (Co-stack Residual Affinity Networks [60]). CSRAN introduces a new co-stacking affinity (CSRA) mechanism along with a multi-level attention refinement (MAR) mechanism. The key idea of CSRA and MAR is to leverage CAFE modules across stacked BiRNN encoders. We show that the extension, CSRAN outperforms CAFE on standard benchmarks.

1.4.2 Multi-Cast Attention Networks for Retrieval-based Natural Language Understanding

We study the retrieval-based NLU problem. Inspired by the previous work on natural language inference, we extend the premise-hypothesis setting to retrieval based, two tower problems. Moreover, retrieval-based NLU also is essential to the entire NLU pipeline, given that it is often used as a candidate filtering step for many MRC or advanced NLU models. We propose a Multi-Cast Attention and an overall model framework (i.e., MCAN) [61]. This work builds upon ideas from the previous contribution (i.e., CAFE model), but adapts it to retrieval tasks, e.g., answer retrieval and dialogue prediction. More concretely, we integrate various max, mean, alignment and intra based pooling attentions into a form of *multi-headed* architecture, achieving state-of-the-art results on all retrieval-based NLU tasks.

1.4.3 Densely Connected Attention Propagation for Machine Reading Comprehension

We study the problem of web-based machine reading comprehension (MRC) and propose a new model that advances the state-of-the-art. Web-based MRC can be considered as a harder problem to NLI and retrieval-based NLU. Hence, we are interested to design powerful architectures for this task. To this end, we propose a new dense attention network and the first usage of attention as a residual connector. The proposed Densely Connected Attention Propagation (DecaProp) [62], comprise the novel DecaEnc (DecaEncoder) and DecaCore modules. DecaProp achieves state-of-the-art performance on multiple well-established machine reading comprehension benchmarks (NewsQA [46], SearchQA [48], Quasar [63] and NarrativeQA[64]), covering a broad range of domains (News, Web Documents, Stories, etc.)

1.4.4 Introspective Curriculum Pointer-Generator for Machine Reading Comprehension over Long Narratives

We study the challenging problem of MRC over narratives (full stories and novels). This problem is extremely hard, given that full stories easily comprises ten of thousands of words. While previous contributions are focused on achieving stellar performance on standard documents, this contribution aims to push the reading comprehension paradigm further. We propose a new Introspective Alignment Reader (IAL Reader), along with a Curriculum Learning-based Pointer-Generator network (IAL-CPG) [65]. The IAL Reader is characterized by block-based local self-attention, which enables scalability benefits for extremely long documents. The key idea is a form of curriculum-based data augmentation, facilitated by generative pretraining on the fly. We achieve state-of-the-art results on the full story setting of the NarrativeQA challenge by Google Deepmind [49].

1.4.5 Natural Language Understanding for Recommender Systems

We study the problem of learning to recommend on web platforms such as Amazon or Yelp. While previous contributions have demonstrated the promise of neural and attention models on NLU tasks, the key idea of this chapter is to leverage NLU-based ideas for recommender systems. We propose an NLU-based framework based on hierarchical co-attention pointers (i.e., Multi-Pointer Co-Attention Networks (MPCN)) [66] for selecting appropriate reviews to learn representations of users and items. Overall, we use NLU-based reasoning techniques for reasoning with user-generated reviews. All in all, the proposed method bridges recommender systems with natural language understanding (NLU), leveraging the general-purpose tools and ideas from language inference to the recommender system community.

1.4.6 Multi-Granular Sequence Encoders for Natural Language Understanding

Aside from attention modules, sequence encoding is a crucial component in NLU. We have proposed reasoning architectures for a suite of NLU tasks. This contribution makes orthogonal improvements to the overall problem of NLU, focusing on the encoding module. We introduce a fast and expressive sequence encoder for reading comprehension. To this end, we propose Dilated Composition Units (DCU) [67], which parameterizes the gating units of a recurrent cell with multi-granular composition functions. The DCU unit comprises a novel fold-unfold operation along with a multi-granular reasoning module. The proposed DCU unit is evaluated on several MRC benchmarks, achieving greater efficiency and performance compared to LSTMs and GRUs. Moreover, we propose a Bi-Attention framework that incorporates DCU units, achieving state-of-the-art results on the RACE [43] benchmark.

1.4.7 Recurrently Controlled Recurrent Networks for Natural Language Understanding

Sequence encoders such as convolutional neural networks and recurrent neural networks are fundamental and pivotal components in many neural models for NLU. We propose Recurrently Controlled Recurrent Networks (RCRN) [68], a new recurrent unit that learns its recurrent gating functions via another recurrent model. We show that RCRN achieves very promising results on several NLU and NLP tasks. Compared to the DCU and other encoding modules, RCRN is designed to be very expressive and powerful.

1.4.8 Hyperbolic Representations for Natural Language Understanding

We propose a new, simple and efficient retrieval-based NLU model based on hyperbolic representations (HyperQA) [2]. While many of the contributions of this thesis are mainly targeted at good performance, there may be occasions that call for mobile-friendly, lightweight and efficient models. Moreover, aside from neural architectures, we postulate that alternative representation learning methods can be useful. To this end, we propose learning neural NLU models in hyperbolic space, imbuing word and sentence representations with hierarchical inductive biases, i.e., hyperbolic space is a geometrically conducive embedding space for learning tree-structured representations, owing to mainly non-uniform notion of the distance across the vector space. We show that HyperQA achieves competitive results in many attention-based and highly parameterized models in the literature. While HyperQA does not achieve state-of-the-art, it certainly serves as a good option for efficient deployment.

1.4.9 Quaternion Representations for Natural Language Understanding

We have previously established that model efficiency is critical to real-world applications. We explore learning efficient representations by training Quaternion models of language. Quaternions are Hypercomplex numbers (one real component

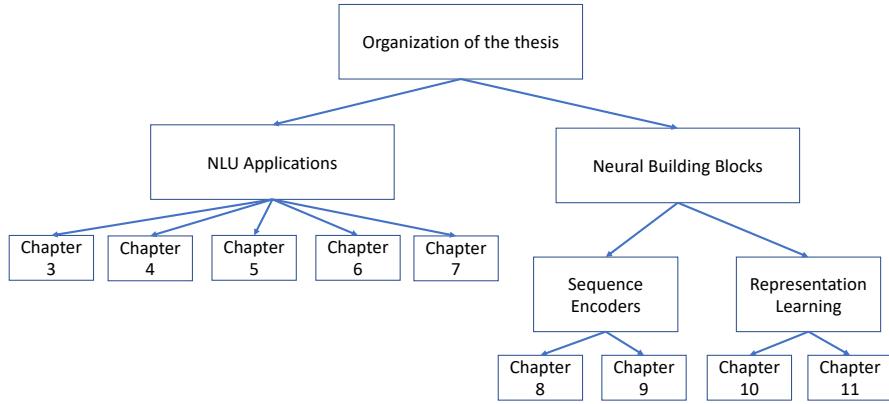


FIGURE 1.1: Hierarchy of Proposed Contributions.

Chapter	Model	Publication
3	CAFE [59], CSRAN [60]	EMNLP'18
4	MCAN [61]	KDD'18
5	DecaProp [62]	NeurIPS'18
6	IAL-CPG [65]	ACL'19
7	MPCN [66]	KDD'18
8	DCU [67]	EMNLP'18
9	RCRN [68]	NeurIPS'18
10	HyperQA [2]	WSDM'18
11	Quaternion NLP [69]	ACL'19

TABLE 1.1: Overview of contributions.

and three imaginary components). The Hamilton product encourages inter-latent components, along with a neat form of weight sharing property. This enables up to four times of parameter savings. We propose Quaternion Attention Models and Quaternion Transformers [69]. We run experiments on a suite of NLU (and allied/related problems) tasks. Overall, we show that Quaternion NLU models are capable of comparable performance while achieving 4 times parameter savings.

Finally, the major contributions of this thesis are summarized in Table 1.1. A hierarchical representation of the proposed contributions is depicted at Figure 1.1.

1.5 Organization of the Thesis

Chapter 1 presents the scope and overview of this thesis.

Chapter 2 reviews the related work of deep learning for natural language understanding.

Chapter 3 introduces CompProp Alignment-Factorized Encoders (CAFE) for natural language inference that advances the existing state-of-the-art. This model utilizes factorized attention in a ComProp (compare-propagate) architecture. This chapter also introduces Co-stack Residual Affinity Aetworks (CSRAN), an extended multi-layered adaptation of CAFE.

Chapter 4 introduces Multi-Cast Attention Networks (MCAN) for a suite of retrieval-based NLU problems. The key idea is to combine various multiple attention flavors and variants (multi-casting). MCAN achieves state-of-the-art on all retrieval-based NLU tasks.

Chapter 5 introduces Densely Connected Attention Propagation (DecaProp), a new model that achieves state-of-the-art on highly competitive machine reading comprehension benchmarks. The key idea is to propagate and densely connect an entire MRC model using attention connectors.

Chapter 6 introduces Introspective Alignment Reader and Curriculum Pointer Generator (IAL-CPG) for machine reading comprehension over extremely long narratives (stories and full novels). The key idea is to use a curriculum learning scheme in conjunction with a side generative model. This simulates generative pre-training and data augmentation. IAL-CPG achieves state-of-the-art on the Google Deepmind Reading Comprehension challenge.

Chapter 7 demonstrates the effectiveness of NLU models on the recommender system. This chapter introduces Multi-Pointer Co-Attention Networks, a state-of-the-art model for utilizing user-generated reviews for recommendation on the web.

Chapter 8 introduces Dilated Composition Units (DCU), a fast, efficient, and highly expressive sequence encoder. We show that DCU outperforms staple LSTM/GRU units on several NLU benchmarks.

Chapter 9 introduces Recurrently Controlled Recurrent Networks (RCRN), a novel RNN encoder. In this model, we parameterize the recurrent gating functions with another RNN in a controller-listener architecture. RCRN outperforms stacked BiLSTMS on 26 NLP/NLU datasets.

Chapter 10 introduces hyperbolic representations for natural language understanding. We show that an efficient model with a low number of parameters can achieve good performance, comparable to many advanced models in the literature. The

key idea is to encode hyperbolic inductive bias through the means of the distance function.

Chapter 11 introduces Quaternion representations for natural language understanding. The key idea is that Quaternion representations enable four times of parameter savings by weight sharing between imaginary components. We propose Quaternion Attention and Quaternion Transformers, demonstrating that competitive performance can be attained even at much lower parameterization.

Chapter 12 gives a conclusion of this thesis, and presents new challenges and directions for future work.

Chapter 2

Literature Review

Deep learning (i.e., gradient-based learning) [70, 71] has made insurmountable progress in recent years. The field of natural language has indeed a lot to gain from the deep learning revolution [17, 24, 34]. To this end, almost, if not all, language-based tasks from question answering to semantic parsing are dominated by neural architectures today [13, 14, 24]. Neural architectures are a composition of differentiable functions. As a whole, they serve as an inductive bias for learning from data. This section discusses the general architectures for many NLU models and tasks. It is also good to note that the scope of this thesis will be mainly based on *supervised learning*.

In this chapter, we review the core NLP tasks (e.g., Natural language inference and Machine Reading Comprehension). Then, we review the fundamental neural architectures for NLU. Note that the first four chapters are pertaining to the NLU tasks while the fifth chapter provides the necessary background for the thesis. A summary of the used benchmarks and datasets is reported at Table .

2.1 Natural Language Inference

The task of NLI is to determine the relation between two sentences, i.e., whether they entail or contradict each other. Natural language inference (or textual entailment recognition) is a long standing problem in NLP research, typically carried out on smaller datasets using traditional methods [72–75].

Dataset	Train	Dev	Test
SNLI	549,367	9,842	9,824
MNLI	392,702	9815 / 9,796	9832 / 9,847
SciTail	23,596	1,304	2,126
Quora	400,000	50,000	50,000
Squad	87,599	10,570	-
NewsQA	92,549	5116	5126
NarrativeQA	32,747	3,461	10,557
SearchQA	99,820	13,393	27,248
Quasar-T	28496	3000	3000
WikiQA	8672	1130	2351
TrecQA	53417	1148	1517
Ubuntu Dialogue Corpus	1M	50,000	50,000

TABLE 2.1: Statistics of datasets.

2.1.1 Related Work

The relatively recent creation of 570K human annotated sentence pairs [45] have spurred on many recent works that use neural networks for NLI. Many advanced neural architectures have been proposed for the NLI task, with most exploiting some variants of neural attention which learn to pay attention to important segments in a sentence [21, 22, 76–78].

Amongst the myriad of neural architectures proposed for NLI, the Enhanced Sequential Inference Model (ESIM) [76] is one of the best performing models. The ESIM, primarily motivated by soft subphrase alignment [21], learns alignments between BiLSTM encoded representations and aggregates them with another BiLSTM layer. The authors also proposed the usage of subtractive composition, claiming that this helps model contradictions amongst alignments.

Compare-Aggregate models are also highly popular in NLI tasks. While this term was coined by [79], many prior NLI models follow this design [21, 76, 80, 81]. The key idea is to aggregate matching features and pass them through a dense layer for prediction. Wang et el. [80] proposed BiMPM, which adopts multi-perspective cosine matching across sequence pairs. Wang et al. [79] proposed a one-way attention and convolutional aggregation layer. Gong et al. [81] learns representations with highway layers and adopts ResNet for learning features over an interaction matrix.

There are several other notable models for NLI. For instance, models that leverage directional self-attention [82] or Gumbel-Softmax [83]. DGEM is a graph-based attention model which was proposed together with a new entailment challenge dataset, SciTail [84]. Pretraining has been known to also be highly useful in the NLI task. For instance, contextualized vectors learned from machine translation (CoVe) [85] or language modeling (ELMo) [17] have shown to be able to improve performance when integrated with existing NLI models.

2.1.2 Benchmarks

The major benchmarks include the Stanford Natural Language Inference (SNLI) [86] benchmark. The SNLI is the first large scale dataset constructed for the purpose of NLI research. In this dataset, annotators were asked to create premise-hypothesis pairs based on reference images. Subsequently, MultiNLI [87] was constructed, introducing domain-specific sentence pairs. SNLI and MultiNLI are key benchmarks for this task for an extended period of time and fierce competition took part on the SNLI leaderboard¹ across a span of one or two years. While SNLI relied on reported test scores from papers, MultiNLI incorporated a test server in which users could only submit predictions. This prevented overfitting on the released test set. Subsequently, following the popularity of SNLI and MultiNLI, more NLI datasets were constructed, such as SciTail [84] which is based on Science question answering examples. It is also popular to cast Quora² question pairs as a NLI problem (e.g., the task of determining if two questions on Quora are paraphrases).

2.2 Retrieval-based Natural Language Understanding

Retrieval-based NLU has formed the bedrock of many NLU applications. For example, before the emerging capabilities of pointer-based or generative question answering, question answering was mainly considered as a retrieval problem (given questions, retrieve answers). A wide spectrum of NLU applications today still

¹<https://nlp.stanford.edu/projects/snli/>

²<https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs>

fall under the retrieval paradigm (e.g., dialogue prediction, open domain question answering, social media search, etc.). Notably, retrieval-based NLU has been a key component in facilitating machine reading comprehension models [40, 88, 89].

2.2.1 Related Work

The dominant state-of-the-art models for retrieval-based NLU today are mostly neural network based models. Neural network models such as convolutional neural networks (CNN) [30, 90–92], recurrent neural networks (RNN) [55, 93, 94] or recursive neural networks [54] are used for learning document representations. A parameterized function such as multi-layered perceptrons [50], tensor layers [95] or holographic layers [91] then learns a similarity score between document pairs.

Recent advances in neural ranking models go beyond independent representation learning. There are several main architectural paradigms that invoke interactions between document pairs which intuitively improve performance due to matching at a deeper and finer granularity. The first can be thought of as extracting features from a constructed word-by-word similarity matrix [96, 97]. The second involves matching across multiple views and perspectives [80, 98, 99]. The third method involves learning pairwise attention weights (i.e., co-attention). In these models, the similarity matrix is used to learn attention weights, learning to attend to each document based on its partner. Attentive Pooling Networks [100] and Attentive Interactive Networks [101] are models that are grounded in this paradigm, utilizing *extractive max-pooling* to learn the relative importance of a word based on its maximum importance to all words in the other document. The Compare-Aggregate model [79] used a co-attention model for matching and then a convolutional feature extractor for aggregating features.

There are several other notable and novel classes of model architectures which have been proposed for document search. Examples include knowledge-enhanced models [55, 102], lexical decomposition [103], fused temporal gates [104] and coupled LSTMs [105]. Novel metric learning techniques such as hyperbolic spaces have also been proposed [106]. Zhang et al. [107] proposed a quantum-like model for matching QA pairs.

The retriever is a key component in NLU. There have been recent works to combine retriever-reader interactions. Wang et al. [108] proposed Reinforced Reader Ranker (R_3) which combines reader and retriever with reinforcement learning. The reader's success is used as a reward for the retriever. Multi-step reasoning and interaction between reader and retriever were also proposed by Das et al. [40]. Lin et al. [89] proposed Denoising QA which jointly learns the retriever and reader.

2.2.2 Benchmarks

Many application domains fall under the retrieval-based NLU paradigm. Answer retrieval benchmarks such as WikiQA [109] and TrecQA [7] have been core benchmarks for answer retrieval. Community based QA datasets such as Yahoo Answers or QatarLiving forums are also actively used [52, 101]. Moreover, there have been several datasets that are concerned with predicting the next dialogue response or reply. The Ubuntu dialogue corpus [55, 56] is a well-established benchmark for dialogue prediction.

2.3 Machine Reading Comprehension

The ability for machines to read documents and retrieve answers has been an extremely recent progress in the NLP research community.

2.3.1 Related Work

Spurred on by the availability of data, many neural models have also been proposed to tackle these challenges. The main technical innovation behind these models can be characterized as:

- **Bidirectional Attention Models** - These models include BiDAF [110], Match-LSTM [34], DCN/DCN+ [111, 112], R-NET [113], DrQA [114], AoA Reader [115], Reinforced Mnemonic Reader [116], ReasoNet [25], AMANDA [117]
- **Multi-step Reasoning** - Reinforced Mnemonic Reader [116], ReasoNet [25]

- **Reinforcement Learning** - Reinforced Mnemonic Reader [116], ReasoNet [25], DCN+ [112], , R³ Reinforced Reader Ranker [88].
- **Self-Attention** DCN+ [112], QANet [118], AMANDA [117]

Many of these works innovate the attention module. The bidirectional attention, first inception by [110] provides a strong foundation. Moreover, the answer pointer layer or span extraction layer [34] also provides the appropriate loss function and training objective for the task at hand.

2.3.2 Benchmarks

Early datasets, e.g., CNN/Dailymail [5] were largely based on cloze style prediction. Subsequently, SQuAD (Stanford Question Answering Dataset) [1], became staple for MRC researchers. SQuAD upped the game, by requiring span selection instead of simply token-based predictions. While CNN/Dailymail were constructed from news articles, SQuAD was constructed from Wikipedia articles, averaging at about 200–300 tokens per document. SQuAD popularized the concept of answer pointers [34] which enable modern deep learning architectures to train on spans of answer text. In general, the current models are able to do reasonably well on this task. Consider the following excerpt from a Wikipedia article about Dr Who.

'Doctor Who is a British science-fiction television programme produced by the BBC since 1963. The programme depicts the adventures of the Doctor, a Time Lord a space and time-travelling humanoid alien. He explores the universe in his TARDIS, a sentient time-travelling space ship. Its exterior appears as a blue British police box, which was a common sight in Britain in 1963 when the series first aired. Accompanied by companions, the Doctor combats a variety of foes, while working to save civilisations and help people in need.'

Question from SQuAD consists of questions such as (1) *What year did Doctor Who first show on TV?* or (2) *What type/genre of TV show is Doctor Who?*. While current models worked quite well on this task, it was soon deemed to be not as difficult as the community had imagined. For example, if the question involved a *date* answer, the model could simply learn to return the only date (1963) in the document without really performing any form of reasoning.

A multitude of new datasets and benchmarks were released shortly. NewsQA [46] focused yet again on MRC on news articles, claiming a greater extent of difficulty as opposed to SQuAD. RACE [43] proposed MCQ-based QA datasets that were constructed from real-world examinations in China. TriviaQA [47] proposed the open-domain setting, in which the context requires some extent of pre-retrieval before any fine-grained reading can take place. In the similar vein, SearchQA [48] tries to emulate the full QA pipeline by using documents retrieved by search engines as MRC input. Quasar [63] is yet another open domain dataset released for this same purpose. Many of the above mentioned datasets involve either extracting the answer or selecting the correct answer from a list of choices. NarrativeQA [64] proposed an extremely challenging benchmark of requiring machines to read entire novels and stories. In its dataset, exact answers may or may not be found in the story and hence, a generative approach becomes mandatory.

2.4 Text-based Recommender Systems

While recommender system research has often little to do with language understanding, this thesis demonstrates how NLU may be used to improve recommender systems.

2.4.1 Related Work

The utility of exploiting reviews for recommendations have been extensively discussed and justified in many works [119–123]. This not only enables a mitigation of cold-start issues but also provides a richer semantic modeling of user and item characteristics. While relatively earlier works have mainly concentrated efforts on topic modeling and language modeling approaches [122, 124, 125], the recent shift towards deep learning models is prominent. The advantages of neural architectures are clear, i.e., not only do these models dispense with laborious feature engineering altogether, they are often highly competitive. In many recent works, Convolutional Neural Networks (CNN) act as automatic feature extractors, encoding a user (item) into a low-dimensional vector representation. User and item embeddings are then compared with a matching function.

An earlier neural model, the Deep Co-operative Neural Networks (DeepCoNN) [119] represents a user as all the reviews that he (she) has written. Likewise, an item is represented as all the reviews ever written (by other users) for it. User and item documents are then encoded with CNNs and passed into a Factorization Machine (FM) [126] for matching. It was later argued that DeepCoNN’s competitive performance exploits the fact that test reviews were leaked (into the training set) [120]. As such, this reduces the recommendation problem to resemble a noisy adaptation of standard document-level sentiment analysis. To this end, Catherine and Cohen [120] proposed TransNets, augmenting a DeepCoNN-like neural network with an additional multi-task learning scheme. More specifically, it learns to transform the penultimate hidden layer of DeepCoNN into a CNN-encoded representation of the test review. This signal was found to be useful, improving the performance on multiple benchmarks.

DeepCoNN and TransNet are relatively simple model architectures. Another recently proposed model, the Dual Attention CNN model (D-ATT) [123] proposed augmenting CNNs with neural attention. The key idea of neural attention [23] is to emphasize important segments of documents by weighting each word by a learned *attention vector*. The final representation comprises a weighted linear combination of all input embeddings. Two variants of attention mechanism are proposed, i.e., local and global, both modeling different views of user-item review documents.

In many ways, the review-based recommender system involves modeling interaction between user and item reviews. It is intuitive and easy to see that this can model many NLU problems (e.g., premise-hypothesis and query-document based NLU). Hence, we postulate that review-based recommender systems can be significantly improved using NLU techniques.

2.4.2 Benchmarks

Experiments on text-based recommendation have been mainly conducted on Amazon, Yelp or BeerAdvocate reviews [119, 120]. Yelp is an online review platform for businesses such as restaurants, bars, spas, etc. The dataset from the Yelp dataset challenge³ is frequently used. On the other hand, Amazon reviews [127, 128] are

³<https://www.yelp.com/dataset/challenge>

mainly concerned with user interaction behaviour and the reviews they have written on the Amazon commerce platform.

2.5 Neural Building Blocks for NLU

This section provides an overview of the core building blocks in NLU research.

2.5.1 Recurrent Neural Networks

This section briefly discusses Recurrent Neural Networks (RNNs). In particular, we focus on Long Short-Term Memory Networks (LSTM) [18] and Gated Recurrent Units (GRU) [19] which are the most popular encoding units for NLU applications.

Long Short-Term Memory (LSTM) LSTM augments vanilla RNNs with gating functions. LSTMs operate via a recurrent loop, processing sequences one time-step at a time. The overall LSTM function can be described as follows:

$$h_t, c_t = \text{LSTM}(x_t, h_{t-1}, c_{t-1}) \quad (2.1)$$

where h_t is the hidden state at time step t and c_t is the cell state at time step t . More specifically, the internal loop of the LSTM unit is defined as:

$$\begin{aligned} i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ c_t &= f_t c_{t-1} + i_t \tanh(W_c x_t + U_c h_{t-1} + b_c) \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\ h_t &= o_t \tanh(c_t) \end{aligned}$$

where $\sigma(\cdot)$ is the sigmoid activation function. i_t, f_t and o_t are the input, forget and output gates of the LSTM cell. W_*, U_*, b_* are the parameters of the LSTM unit where $* \in \{i, f, o, c\}$. The incorporation of gating functions are targetted at combating the vanishing/exploding gradient problem, as well as enabling expressive modeling of dependencies over time.

Gated Recurrent Units (GRUs) GRUs are a variant of RNNs that are considered lightweight variants of LSTM units.

$$h_t = GRU(x_t, h_{t-1}) \quad (2.2)$$

GRUs dispense away with the cell state. Instead, it only relies on an update gate and reset gate. The internal loop of the GRU cell is written as:

$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\ r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \\ \hat{h}_t &= \tanh(W_h x_t + U_h(r_t h_{t-1}) + b_h) \\ h_t &= z_t h_{t-1} + (1 - z_t) \hat{h}_t \end{aligned}$$

where h_t is the hidden state at time step t , and z_t and r_t are the update gate and reset gate at time step t respectively. σ is the sigmoid function. x_t is the input to the GRU unit at time step t .

Bidirectional Variants LSTM and GRU units are often used in a *bidirectional* fashion. In this case, the input sequence is parsed from both ends and then concatenated to form the output representation. It is common to refer to these networks as BiLSTMs and/or BiGRUs.

Recent Progress in Recurrent Network Research In lieu of the fact that LSTM units date back to 1970s [18], it would be interesting to investigate the recent advances in recurrent neural network research. There have been interesting uses of parameterizing recurrent gates with convolution [129]. Sparse Attentive Backtracking [130] exploits attention in recurrent units for fast access to previous hidden states. Relational Recurrent Networks [131] leverages self-attention for relational reasoning across memory cells within recurrent units. The usage of convolution [132] and/or dilation [133] within recurrent cells is also notable [134]. Exploring novel inductive biases such as non-Euclidean Hyperbolic RNNs [135], complex RNNs [136] and/or Hypercomplex Quaternion RNNs [137] have also been a promising direction.

2.5.2 Neural Attention

Learning relative importance amongst sequence inputs is the core principle in neural attention models. Attention, alone, has been one of the major driving forces of modern deep learning research. Though simple, this simple paradigm has demonstrated to not only be pervasive but also extremely powerful [4, 22–24, 110]. This was first inceptioned as a form of *alignment* mechanism [23] which was used to align source-target pairs in machine translation. This section discusses several common variants of attention networks.

Vanilla Attention This is a standard attention module in which the input sequence is compared against a context vector. Let $H \in \mathbb{R}^{\ell \times d}$ be the input sequence.

$$Y = \text{Softmax}((QC)^\top)H \quad (2.3)$$

where $Q = F(H)$ and $F(\cdot)$ is a parameterized function such as $F(X) = WX + b$. $C \in \mathbb{R}^{d \times 1}$ is a trainable context vector. The output vector $Y \in \mathbb{R}^d$ is a learned weighted sum of the input sequence H . This can be interpreted as (1) transforming each input vector, followed by (2) performing a vector-wise dot product with context vector C and finally (3) using the output (with softmax) to weight the original input H .

Cross Attention In vanilla attention, we compute similarity scores with the context vector. Now let us consider the case where there are two input sequences and each word in sequence A attends to all words in B (and vice versa). This can be written as:

$$\begin{aligned} C &= \text{Softmax}(AB^\top)B \\ D &= \text{Softmax}((AB^\top)^\top A) \end{aligned}$$

where $AB^\top \in \mathbb{R}^{\ell_A \times \ell_B}$, $C \in \mathbb{R}^{\ell_B \times d}$ and $D \in \mathbb{R}^{\ell_A \times d}$. This formulation is also the Decomposable Attention [21] proposed for NLI. This formulation learns alignment between sequences. In this case, the next layer, i.e., $F(C, B)$ is often used to compare aligned representations. $F(\cdot)$ is a parameterized function and can be a feed-forward neural network or recurrent neural network. There have been many

variations of cross attention which come in various names. For example, Bidirectional Attention [110], Co-attention [111, 112] which have become staple techniques in MRC. Notably, this form of attention actually performs alignment in similar spirit to the alignment modules in neural machine translation. It is also good to note that a separate class of attention modules perform extraction by utilizing max or mean pooling operations [4, 100, 138, 139]. Although very different in nature, we may still consider them as cross attention methods due to the fact that they learn attention weights conditioned on another sequence.

Self-Attention A dominant method in recent literature is to utilize self-attention for single sequence representation learning [24].

$$A = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V, \quad (2.4)$$

where Q, K, V are linear transforms from the input sequence H . d is the dimensionality of the input vectors. Note that self-attention is essentially applying cross attention to the identical input sequence, albeit projected with another set of parameters. Self-attention was popularized by Transformers [24] although it can be traced back to earlier work and sometimes referred to as intra-attention [21] or self-matching [113]. Self-attention layers have been utilized frequently as sequence encoders (as in machine translation) and are also frequently used in NLU (e.g., MRC [118] or NLI [21]).

Scoring Function The three above-mentioned attention modules operate on a dot-product (or scaled dot-product) similarity. Across the rich history of attention research, other forms of scoring functions have also been adopted. For example, Cosine Similarity [140], Additive [23], Location-based Attention [141] or parameterizing the scoring function with a bilinear scoring function [141] are representative examples of scoring function variants. In recent years, novel methods such as Hyperbolic Attention [142] or Hermitian Attention [143] have also been proposed. However, to date, dot-product attention remains the most popular. A plausible reason is due to GPU-efficient matrix multiplication. Owing to the quadratic computation and inelegant requirement for tensor tiling, the other scoring functions may not be preferred.

2.6 Summary

In this chapter, we provided an overview of the key NLU tasks, namely Natural Language Inference, Retrieval-based NLU, Machine Reading Comprehension and Text-based Recommender Systems. We also provided a detailed overview of the common neural building blocks for NLU such as recurrent neural networks and attentional models. We also discussed RNN and attention variants. Additionally, we also discussed recent advances in recurrent neural network research.

Chapter 3

Factorized Neural Attention Models for Natural Language Inference

Natural Language Inference (NLI) is a pivotal and fundamental task in language understanding and artificial intelligence. More concretely, given a premise and hypothesis, NLI aims to detect whether the latter *entails* or *contradicts* the former. As such, NLI is also commonly known as Recognizing Textual Entailment (RTE). NLI is known to be a significantly challenging task for machines whose success often depends on a wide repertoire of reasoning techniques. In this chapter¹, we discuss our proposed models for NLI.

3.1 Introduction

In recent years, there has been a steep improvement in NLI systems, largely contributed by the release of the largest publicly available corpus for NLI - the Stanford Natural Language Inference (SNLI) corpus [45] which comprises 570K hand labeled

¹This chapter is published as *Compare, Compress and Propagate: Enhancing Neural Architectures with Alignment Factorization for Natural Language Inference*, Proceedings of EMNLP 2018 [59] and *Co-Stack Residual Affinity Networks with Multi-level Attention Refinement for Matching Text Sequences*, Proceedings of EMNLP 2018 [60].

sentence pairs. This has improved the feasibility of training complex neural models, given the fact that neural models often require a relatively large amount of training data.

Highly competitive neural models for NLI are mostly based on soft-attention alignments, popularized by [21, 22]. The key idea is to learn an alignment of sub-phrases in both sentences and learn to compare the relationship between them. Standard feed-forward neural networks are commonly used to model similarity between aligned (decomposed) sub-phrases and then aggregated into the final prediction layers.

Alignment between sentences has become a staple technique in NLI research and many recent state-of-the-art models such as the Enhanced Sequential Inference Model (ESIM) [76] also incorporate the alignment strategy. The difference here is that ESIM considers a non-parameterized comparison scheme, i.e., *concatenating* the subtraction and element-wise product of aligned sub-phrases, along with two original sub-phrases, into the final comparison vector. A bidirectional LSTM is then used to aggregate the compared alignment vectors.

In this chapter, we propose a *compare, compress and propagate* (ComProp) architecture where compressed alignment features are propagated to upper layers (such as an RNN-based encoder) for enhancing representation learning. Then, in order to achieve an efficient propagation of alignment features, we propose alignment factorization layers to reduce each alignment vector to a single scalar valued feature. Each scalar valued feature is used to augment the base word representation, allowing the subsequent RNN encoder layers to benefit from not only global but also cross sentence information.

There are several major advantages to our proposed architecture. Firstly, our model is relatively compact, i.e., we compress alignment feature vectors and augment them to word representations instead. This is to avoid large alignment (or match) vectors being propagated across the network. As a result, our model is more parameter efficient compared to ESIM since the width of the middle layers of the network is now much smaller. To the best of our knowledge, this is the first work that explicitly employs such a paradigm.

Secondly, the explicit usage of compression enables improved interpretability since each alignment pair is compressed to a scalar and hence, can be easily visualized.

Previous models such as ESIM use subtractive operations on alignment vectors, edging on the intuition that these vectors represent a contradiction. Our model is capable of visually demonstrating this phenomenon. As such, this design choice enables a new way of deriving insight from neural NLI models.

Thirdly, the alignment factorization layer is expressive and powerful, combining ideas from standard machine learning literature [144] with modern neural NLI models. The factorization layer tries to decompose the alignment vector (constructed from the variations of $a - b$, $a \odot b$ and $[a; b]$), learning higher-order feature interactions between each compared alignment. In other words, it models the second-order (pairwise) interactions between *each* feature in *every* alignment vector using factorized parameters, allowing more expressive comparison to be made over traditional fully-connected layers (FC). Moreover, factorization-based models are also known to be able to model low-rank structure and reduce risks of overfitting. The effectiveness of the factorization alignment over alternative baselines such as feed-forward neural networks is confirmed by early experiments.

The major contributions of this chapter are summarized as follows:

- We introduce a *Compare, Compress and Propagate* (ComProp) architecture for NLI. The key idea is to use the myriad of generated comparison vectors for augmentation of the base word representation instead of simply aggregating them for prediction. Subsequently, a standard compositional encoder can then be used to learn representations from the augmented word representations. We show that we are able to derive meaningful insight from visualizing these augmented features.
- For the first time, we adopt expressive factorization layers to model the relationships between soft-aligned sub-phrases of sentence pairs. Empirical experiments confirm the effectiveness of this new layer over standard fully connected layers.
- Overall, we propose a new neural model - CAFE (**C**omProp **A**lignment-**F**actorized **E**ncoders) for NLI. The model achieves state-of-the-art performance on SNLI, MultiNLI and the newly released SciTail dataset, outperforming existing state-of-the-art models such as ESIM. Ablation studies confirm the effectiveness of each proposed component in our model.

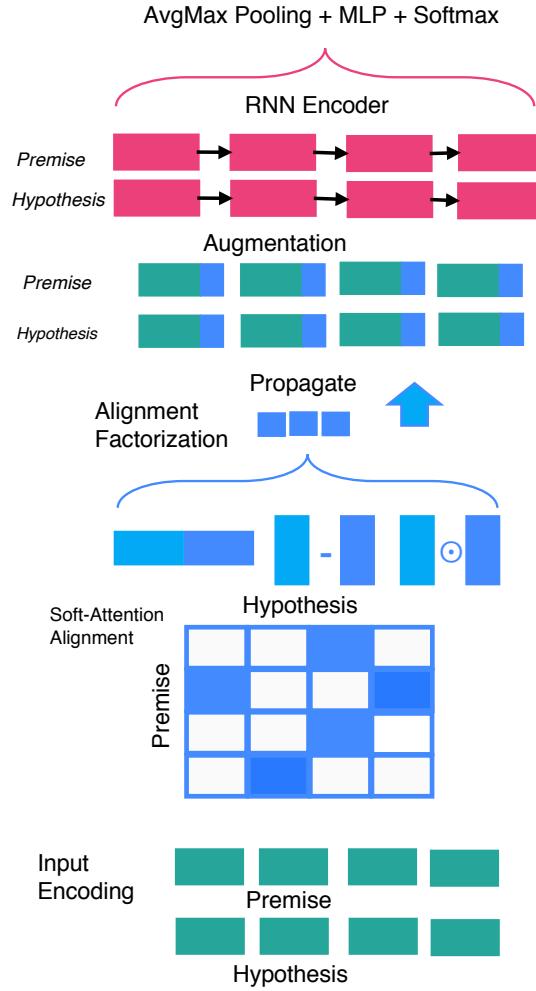


FIGURE 3.1: Architecture of the CAFE model.

- Additionally, we propose a new co-stacking mechanism, leveraging extended techniques for utilizing CAFE layers. This model, CSRAN (Co-stack Residual Affinity Networks), outperforms the original CAFE model on NLI tasks.

In the subsequent sections, we provide a layer-by-layer description of our proposed model architectures, CAFE and CSRAN.

3.2 Proposed Method (CAFE)

The proposed CAFE model accepts two sentences as an input, i.e., P (premise) and H (hypothesis). Figure 3.1 illustrates a high-level overview of the proposed model.

3.2.1 Input Encoding Layer

This layer aims to learn a k -dimensional representation for each word. Following [81], we learn feature-rich word representations by concatenating word embeddings, character embeddings and syntactic (part-of-speech tag) embeddings (provided in the datasets). Character representations are learned using a convolutional encoder with max pooling function which is commonly used in many relevant literature [80, 145].

Highway Encoder Subsequently, we pass each concatenated word vector into a two-layer highway network [146] in order to learn a k -dimensional representation. Highway networks are gated projection layers which learn to adaptively control how much information is being carried to the next layer. The strategy is similar to [21] which trains the projection layer in place of tuning the embedding matrix. The usage of highway layers over standard projection layers is empirically motivated. An intuition would be that the gates in this layer adapt to learn the relative importance of each word to the NLI task. Let $H(\cdot)$ and $T(\cdot)$ be single layered affine transforms with ReLU and sigmoid activation functions respectively. A single highway network layer is defined as:

$$y = H(x, W_H) \cdot T(x, W_T) + C \cdot x \quad (3.1)$$

where $C = (1 - T(x, W_T))$ and $W_H, W_T \in \mathbb{R}^{r \times d}$. Notably, the dimensions of the affine transform might be different from the size of the input vector. In this case, an additional nonlinear transform is used to project x to the same dimensionality. The output of this layer is $\bar{P} \in \mathbb{R}^{k \times \ell_P}$ (premise) and $\bar{H} \in \mathbb{R}^{k \times \ell_H}$ (hypothesis), with each word converted to a r -dimensional vector.

3.2.2 Soft-Attention Alignment Layer

This layer describes two soft-attention alignment techniques that are used in the model.

Inter-Attention Alignment Layer This layer learns an alignment of sub-phrases between \bar{P} and \bar{H} . Let $F(\cdot)$ be a standard projection layer with ReLU

activation function. The alignment matrix of two sequences is defined as follows:

$$e_{ij} = F(\bar{p}_i)^\top \cdot F(\bar{h}_j) \quad (3.2)$$

where $E \in \mathbb{R}^{\ell_p \times \ell_h}$ and \bar{p}_i, \bar{h}_j are the i -th and j -th word in the premise and hypothesis respectively.

$$\beta_i = \sum_{j=1}^{\ell_p} \frac{\exp(e_{ij})}{\sum_{k=1}^{\ell_p} \exp(e_{ik})} \bar{p}_j \quad (3.3)$$

$$\alpha_j = \sum_{i=1}^{\ell_h} \frac{\exp(e_{ij})}{\sum_{k=1}^{\ell_h} \exp(e_{kj})} \bar{h}_i \quad (3.4)$$

where β_i is the sub-phrase in \bar{P} that is softly aligned to h_i . Intuitively, β_i is a weighted sum across $\{p_j\}_{j=1}^{\ell_p}$, selecting the most relevant parts of \bar{P} to represent h_i .

3.2.3 Intra-Attention Alignment Layer

This layer learns a *self-alignment* of sentences and is applied to both \bar{P} and \bar{H} independently. For the sake of brevity, let \bar{S} represent either \bar{P} or \bar{H} , the intra-attention alignment is computed as:

$$s'_i = \sum_{j=1}^{\ell_p} \frac{\exp(f_{ij})}{\sum_{k=1}^{\ell_p} \exp(f_{ik})} \bar{s}_j \quad (3.5)$$

where $f_{ij} = G(\bar{s}_i)^\top \cdot G(\bar{s}_j)$ and $G(\cdot)$ is a nonlinear projection layer with ReLU activation function. The intra-attention layer models the similarity of each word with respect to the entire sentence, capturing long distance dependencies and ‘global’ context of the entire sentence.

3.2.4 Alignment Factorization Layer

This layer aims to learn a scalar valued feature for each comparison between aligned sub-phrases. Firstly, we introduce our factorization operation, which lives at the core of our neural model.

Factorization Operation Given an input vector x , the factorization operation [144] is defined as:

$$Z(x) = w_0 + \sum_{i=1}^n w_i x_i + \sum_{i=1}^n \sum_{j=i+1}^n \langle v_i, v_j \rangle x_i x_j \quad (3.6)$$

where $Z(x)$ is a scalar valued output, $\langle \cdot, \cdot \rangle$ is the dot product between two vectors and w_0 is the global bias. Factorization machines model low-rank structure within the matching vector producing a scalar feature. The parameters of this layer are $w_0 \in \mathbb{R}$, $w \in \mathbb{R}^r$ and $v \in \mathbb{R}^{r \times k}$. The first term $\sum_{i=1}^n w_i x_i$ is simply a linear term. The second term $\sum_{i=1}^n \sum_{j=i+1}^n \langle v_i, v_j \rangle x_i x_j$ captures all pairwise interactions in x (the input vector) using the factorization of matrix v .

3.2.5 Inter-Alignment Factorization

This operation compares the alignment between inter-attention aligned representations, i.e., (β_i, h_i) and (α_j, p_j) . Let (a, b) represent an alignment pair, we apply the following operations:

$$y_c = Z([a; b]) ; y_s = Z(a - b) ; y_m = Z(a \odot b) \quad (3.7)$$

where $y_c, y_s, y_m \in \mathbb{R}$, $Z(\cdot)$ is the factorization operation, $[.;.]$ is the concatenation operator and \odot is the element-wise multiplication. The intuition of modeling subtraction is targeted at capturing contradiction. However, instead of simply concatenating the extra comparison vectors, we compress them using the factorization operation. Finally, for each alignment pair, we obtain three scalar-valued features which map precisely to a word in the sequence.

3.2.6 Intra-Alignment Factorization

Next, for each sequence, we also apply alignment factorization on the intra-aligned sentences. Let (s, s') represent an *intra-aligned* pair from either the premise or hypothesis, we compute the following operations:

$$v_c = Z([s; s']) ; v_s = Z(s - s') ; v_m = Z(s \odot s') \quad (3.8)$$

where $v_c, v_s, v_m \in \mathbb{R}$ and $Z(\cdot)$ is the factorization operation. Applying alignment factorization to intra-aligned representations produces another three scalar-valued features which are mapped to each word in the sequence. Note that each of the *six* factorization operations has its own parameters but shares them amongst all words in the sentences.

3.2.7 Propagation and Augmentation

Finally, the *six* factorized features are then aggregated via concatenation to form a final feature vector that is propagated to upper representation learning layers via augmentation of the word representation \bar{P} or \bar{H} .

$$u_i = [s_i; f_{intra}^i; f_{inter}^i] \quad (3.9)$$

where s_i is i -th word in \bar{P} or \bar{H} , and f_{intra}^i and f_{inter}^i are the intra-aligned $[v_c; v_s; v_m]$ and inter-aligned $[y_c; y_s; y_m]$ features for the i -th word in the sequence respectively. Intuitively, f_{intra}^i augments each word with global knowledge of the sentence and f_{inter}^i augments each word with cross-sentence knowledge via inter-attention.

3.2.8 Sequential Encoder Layer

For each sentence, the augmented word representations u_1, u_2, \dots, u_ℓ are then passed into a sequential encoder layer. We adopt a standard vanilla LSTM encoder.

$$h_i = LSTM(u, i), \forall i \in [1, \dots, \ell] \quad (3.10)$$

where ℓ represents the maximum length of the sequence. Notably, the parameters of the LSTM are *siamese* in nature, sharing weights between both premise and hypothesis. We do not use a bidirectional LSTM encoder, as we found that it did not lead to any improvements on the held-out set. A logical explanation would be because our word representations are already augmented with global (intra-attention) information. As such, modeling in the reverse direction is unnecessary, resulting in some computational savings.

3.2.9 Pooling Layer

Next, to learn an overall representation of each sentence, we apply a pooling function across all hidden outputs of the sequential encoder. The pooling function is a concatenation of temporal max and average (avg) pooling.

$$x = [\max([h_1, \dots, h_\ell]); \text{avg}([h_1, \dots, h_\ell])] \quad (3.11)$$

where x is a final $2k$ -dimensional representation of the sentence (premise or hypothesis). We also experimented with *sum* and *avg* standalone poolings and found *sum* pooling to be relatively competitive.

3.2.10 Prediction Layer

Finally, given a fixed dimensional representation of the premise x_p and hypothesis x_h , we pass their concatenation into a two-layer h -dimensional highway network. Since the highway network has been already defined earlier, we omit the technical details here. The final prediction layer of our model is computed as follows:

$$y_{out} = H_2(H_1([x_p; x_h; x_p \odot x_h; x_p - x_h])) \quad (3.12)$$

where $H_1(\cdot), H_2(\cdot)$ are highway network layers with ReLU activation. The output is then passed into a final linear softmax layer.

$$y_{pred} = \text{softmax}(W_F \cdot y_{out} + b_F) \quad (3.13)$$

where $W_F \in \mathbb{R}^{h \times 3}$ and $b_F \in \mathbb{R}^3$. The network is then trained using standard multi-class cross entropy loss with L2 regularization.

3.2.11 Additional Discussion

Our proposed method compares and compresses alignment pairs using factorization layers which leverages the rich history of standard machine learning literature. The factorization layers incorporate highly expressive factorization machines (FMs) [144] into neural NLI models. In standard machine learning tasks, FMs remain a

very competitive choice for learning feature interactions [147] for both standard classification and regression problems. Intuitively, FMs are adept at handling data sparsity (typically interactions) by using factorized parameters to approximate a feature matching matrix. This makes it suitable in our model architecture since feature interaction between sub-phrase alignment pairs is typically very sparse as well.

A recent work [148] reports an interesting empirical study pertaining to the ability of standard FC layers and their ability to model ‘cross features’ (or multiplicative features). Their overall finding suggests that while standard ReLU FC layers are able to approximate 2-way or 3-way features, they are extremely inefficient in doing so (requiring either very wide or deep layers). This further motivates the usage of FMs in this work and is well aligned with our empirical results, i.e., strong competitive performance with reasonably small parameterization.

3.3 Proposed Method (CSRAN)

This section describes an extended model using CAFE as a base neural building block which further improves the performance of CAFE. We call this CSRAN (Co-stacked Residual Affinity Networks) in which the key idea is to present an extended technique for training a deeper version of CAFE. The key differences between CAFE and CSRAN are as follows:

- CSRAN utilizes multiple RNN encoder layers, i.e., deep stacked LSTMS. Conversely, CAFE is a shallow model.
- CSRAN presents a new Multi-level Attention Refinement Model to use CAFE to refine representations for the stacked BiLSTM layer.
- CSRAN presents a new Co-Stack Residual Affinity module that computes attention weights by considering all stacked layers instead of only the last layer. This helps to improve the gradient flow as well as the expressiveness of the interaction between premise and hypothesis.

Figure 3.2 describes the architecture of CSRAN.

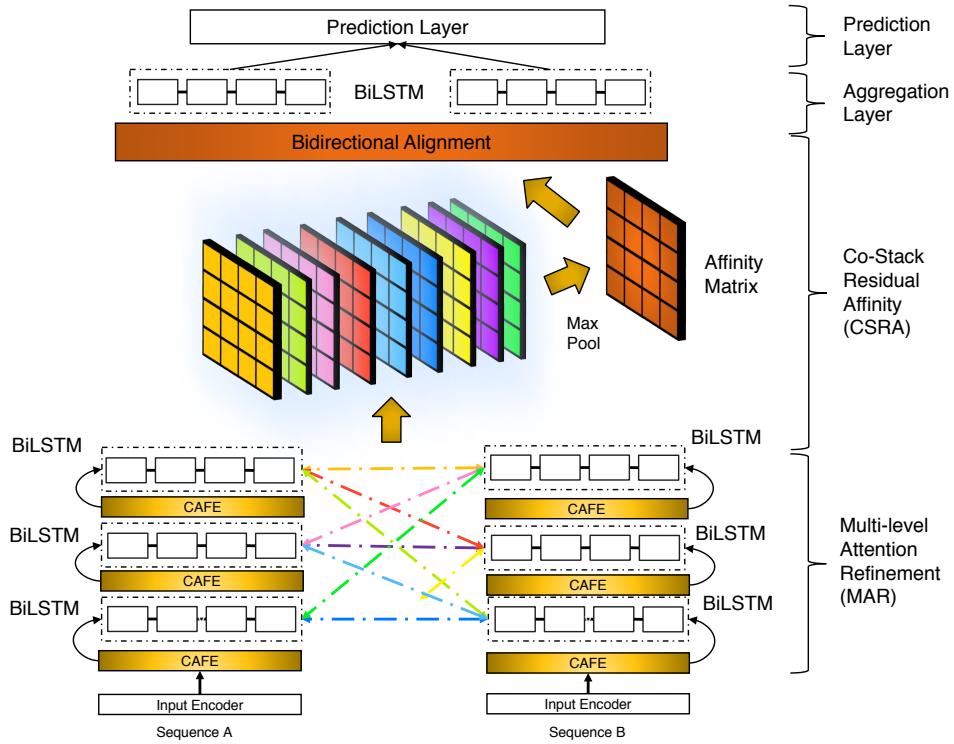


FIGURE 3.2: Architecture of CSRAN.

3.3.1 Input Encoder

The inputs to our model are standard sequences of words A and B which represent sequence a and sequence b respectively. In the context of different applications, a and b take different roles such as premise/hypothesis or question/answer. Both sequences are converted into word representations (via pretrained word embeddings) and character-based representations. Character embeddings are trainable parameters and a final character-based word representation of d dimensions is learned by passing all characters into a Bidirectional LSTM encoder. This is standard, following many works such as [149]. Word embeddings and character-based word representations are then concatenated to form the final word representation. Then, the word representation is passed through a (optional and tuned as a hyperparameter) 2-layered highway network of d dimensions.

3.3.2 Stacked Recurrent Encoders

Next, word representations are passed into a stacked recurrent encoder layer. Specifically, we use Bidirectional LSTM encoders at this layer. Let k be the number

of layers of the stacked recurrent encoder layer.

$$h_t^i = \text{BiLSTM}_i(h_{t-1}) \quad \forall t \in [1, 2 \dots \ell] \quad (3.14)$$

where BiLSTM_i represents the i -th BiLSTM layer and h_t^i represents the t -th hidden state of the i -th BiLSTM layer. ℓ is the sequence length. Note that the parameters are shared for both a and b .

3.3.3 Multi-level Attention Refinement (MAR)

Additionally, we may utilize CAFE blocks between the BiLSTM layers. Each CAFE block returns *six* features, which are generated by a factorization operation using factorization machines (FM). We utilize this in a multi-layered fashion which we found to have worked well. This constitutes our multi-level attention refinement mechanism. More concretely, we apply the CAFE operation to the outputs of each BiLSTM layer, allowing the next BiLSTM layer to process the ‘augmented’ representations. The next layer retains its dimensionality by projecting the augmented representation back to its original size using the BiLSTM encoder. This can be interpreted as repeatedly refining representations via attention. As such, adding CAFE blocks is a very natural fit to the stacked recurrent architecture.

This layer is the cornerstone of our proposed approach and is represented as the middle segment of Figure 3.2 (the colorful matrices).

3.3.4 Co-Stacking

Co-stacking refers to the fusion of a and b across multiple hierarchies. Recall that the affinity score between two words is typically computed by $s_{ij} = a^\top b$. We extend this to a residual formulation. More concretely, the affinity score between both words is now computed as the *maximum* influence it has over all layers.

$$s_{ij} = \max \sum_p \sum_q a_{pi}^\top b_{qj} \quad (3.15)$$

where a_{pi} is the i -th word for the p -th stacked layer for a and b_{qj} is the j -th word for the q -th stacked layer for b . The choice of the maximum operator is intuitive

and is strongly motivated by the fact that we would like to give a high affinity for each word pair that shows a strong match at any of different hierarchical stages of learning representations. Note that this layer can be interpreted as constructing a matching tensor based on multi-hierarchical information and selecting the most informative match across all representation hierarchies.

3.3.5 Bidirectional Alignment

In order to learn (bidirectionally) attentive representations, we first concatenate all stacked outputs to form a $\ell \times kd$ vector. Next, we apply the following operations to $A \in \mathbb{R}^{\ell_a \times kd}$ and $B \in \mathbb{R}^{\ell_b \times kd}$.

$$\bar{A} = \text{Softmax}(S^\top)B \text{ and } \bar{B} = A \text{ Softmax}(S^\top) \quad (3.16)$$

where $\bar{A} \in \mathbb{R}^{\ell_b \times kd}$, $\bar{B} \in \mathbb{R}^{\ell_a \times kd}$ are the attentive (aligned) representations.

3.3.6 Matching and Aggregation Layer

Next, we match the attentive (aligned) representations using the subtraction, element-wise multiplication and concatenation of each aligned word. Subsequently, we pass this matching vector into a k -layered BiLSTM layer.

$$a'_i = \text{BiLSTM}_k([\bar{b}_i - a_i, \bar{b}_i \odot a_i, \bar{b}_i, a_i]) \quad (3.17)$$

$$b'_i = \text{BiLSTM}_k([\bar{a}_i - b_i, \bar{a}_i \odot b_i, \bar{a}_i, b_i]) \quad (3.18)$$

The final feature representation is learned via the summation across the temporal dimension as follows:

$$z = [\sum_{i=1}^{\ell_a} a'_i ; \sum_{i=1}^{\ell_b} b'_i] \quad (3.19)$$

where $[.;.]$ is the concatenation operator.

3.3.7 Output and Prediction Layer

Next, the model predicts using the feature vector z for every given sequence pair. At this layer, we utilize standard fully-connected layers. The number of output layers is typically 2-3 and is a tuned hyperparameter. Softmax is applied onto the final layer. The final layer is application specific, e.g., k classes for classification tasks and a two-class softmax for pointwise ranking. For all datasets, we optimize the cross entropy loss.

3.4 Experiments (CAFE)

In this section, we describe our experimental setup and report the experimental results for the CAFE model.

3.4.1 Experimental Setup

To ascertain the effectiveness of our models, we use the SNLI [45] and MultiNLI [87] benchmarks which are standard and highly competitive benchmarks for the NLI task. We also include the newly released SciTail dataset [84] which is a binary entailment classification task constructed from science questions. Notably, SciTail is known to be a difficult dataset for NLI, made evident by the low accuracy scores even though it is binary in nature.

SNLI The state-of-the-art competitors on this dataset are the BiMPM [80], ESIM [76] and DIIN [81]. We compare against competitors across three settings. The first setting disallows cross sentence attention. In the second setting, cross sentence is allowed. The last (third) setting is a comparison between model ensembles while the first two settings only comprise single models. Note that we consider the 1st setting to be relatively less important (since our focus is not on the encoder itself) but still report the results for completeness.

MultiNLI We compare on two test sets (*matched* and *mismatched*) which represent in-domain and out-domain performance. The main competitor on this dataset

is the ESIM model, a powerful state-of-the-art SNLI baseline. We also compare with ESIM + Read [150].

SciTail This dataset only has one official setting. We compare against the reported results of ESIM [76] and DecompAtt [21] in the original paper. We also compare with DGEM, the new model proposed in [84].

Across all experiments and in the spirit of fair comparison, we only compare with works that (1) do not use extra training data and (2) do not use external resources (such as external knowledge bases, etc.). However, for the sake of completeness, we still report their scores [17, 85, 151].

3.4.2 Implementation Details

We implement our model in TensorFlow [152] and train them on Nvidia P100 GPUs. We use the Adam optimizer [153] with an initial learning rate of 0.0003. L2 regularization is set to 10^{-6} . Dropout with a keep probability of 0.8 is applied after each fully-connected, recurrent or highway layer. The batch size is tuned amongst {128, 256, 512}. The number of latent factors k for the factorization layer is tuned amongst {5, 10, 50, 100, 150}. The size of the hidden layers of the highway network layers is set to 300. All parameters are initialized with xavier initialization. Word embeddings are pre-loaded with 300d GloVe embeddings [16] and fixed during training. Sequence lengths are padded to batch-wise maximum. The batch order is (randomly) sorted within buckets following [21].

3.4.3 Experimental Results

Table 3.1 reports our results on the SNLI benchmark. On the cross sentence (single model setting), the performance of our proposed CAFE model is extremely competitive. We report the test accuracy of CAFE at different extents of parameterization, i.e., varying the size of the LSTM encoder, width of the pre-softmax hidden layers and final pooling layer. CAFE obtains 88.5% accuracy on the SNLI test set, an extremely competitive score on the extremely popular benchmark. Notably, competitive results can be also achieved with a much smaller parameterization. For example, CAFE also achieves 88.3% and 88.1% test accuracy with

TABLE 3.1: Results (CAFE) on the SNLI dataset.

Model	Params	Train	Test
Single Models (w/o Cross Sentence Attention)			
300D Gumbel TreeLSTM [83]	2.9M	91.2	85.6
300D DISAN [82]	2.4M	91.1	85.6
300D Residual Stacked Encoders [154]	9.7M	89.8	85.7
600D Gumbel TreeLSTM [83]	10M	93.1	86.0
300D CAFE (w/o CA)	3.7M	87.3	<u>85.9</u>
Single Models			
100D LSTM with attention [22]	250K	85.3	83.5
300D mLSTM [77]	1.9M	92.0	86.1
450D LSTMN + deep att. fusion [155]	3.4M	88.5	86.3
200D DecompAtt + Intra-Att [21]	580K	90.5	86.8
300D NTI-SLSTM-LSTM [156]	3.2M	88.5	87.3
300D re-read LSTM [157]	2.0M	90.7	87.5
BiMPM [80]	1.6M	90.9	87.5
448D DIIN [81]	4.4M	91.2	88.0
600D ESIM [76]	4.3M	92.6	88.0
150D CAFE (SUM+2x200D MLP)	750K	88.2	87.7
200D CAFE (SUM+2x400D MLP)	1.4M	89.4	88.1
300D CAFE (SUM+2x600D MLP)	3.5M	89.2	<u>88.3</u>
300D CAFE (AVGMAX+300D HN)	4.7M	89.8	88.5
Ensemble Models			
600D ESIM + 300D Tree-LSTM [76]	7.7M	93.5	88.6
BiMPM [80]	6.4M	93.2	88.8
448D DIIN [81]	17.0M	92.3	88.9
300D CAFE (Ensemble)	17.5M	92.5	89.3
External Resource Models			
BiAttentive Classification + CoVe + Char [85]	22M	88.5	88.1
KIM [151]	4.3M	94.1	88.6
ESIM + ELMo [17]	8.0M	91.6	88.7
200D CAFE (AVGMAX + 200D MLP) + ELMo	1.4M	89.5	89.0

only $3.5M$ and $1.5M$ parameters respectively. This outperforms the state-of-the-art ESIM and DIIN models with only a fraction of the parameter cost. At 88.1% accuracy, our model has about three times less parameters than ESIM/DIIN (i.e., $1.4M$ versus $4.3M/4.4M$). Moreover, our lightweight adaptation achieves 87.7% accuracy with only $750K$ parameters, which makes it extremely performant amongst models having the same amount of parameters such as the decomposable attention model (86.8%).

Finally, an ensemble of 5 CAFE models achieves 89.3% test accuracy, the best test

TABLE 3.2: Results (CAFE) on MultiNLI and SciTail datasets.

Model	MultiNLI		SciTail
	Match	Mismatch	-
Majority	36.5	35.6	60.3
NGRAM [#]	-	-	70.6
CBOW ^þ	65.2	64.8	-
BiLSTM ^þ	69.8	69.4	-
ESIM ^{#,þ}	72.4	72.1	70.6
DecompAtt [#] -	-	-	72.3
DGEM [#]	-	-	70.8
DGEM + Edge [#]	-	-	77.3
ESIM [†]	76.3	75.8	-
ESIM + Read [†]	77.8	77.0	-
CAFE	78.7	77.9	83.3
CAFE Ensemble	80.2	79.0	-

scores on the SNLI benchmark². Overall, we believe that the good performance of our CAFE can be attributed to (1) the effectiveness of the ComProp architecture (i.e., providing word representations with global and local knowledge for better representation learning) and (2) the expressiveness of alignment factorization layers that are used to decompose and compare word alignments. More details are given at the ablation study. Finally, we emphasize that CAFE is also relatively lightweight, efficient and fast to train given its performance. A single run on SNLI takes approximately 5 minutes per epoch with a batch size of 256. Overall, a single run takes ≈ 3 hours to get to convergence.

Table 3.2 reports our results on the MultiNLI and SciTail datasets. Models with \dagger , $\#$ and \flat are reported from [150], [84] and [87] respectively. On MultiNLI, CAFE significantly outperforms ESIM, a strong state-of-the-art model on both settings. We also outperform the ESIM + Read model [150]. An ensemble of CAFE models achieve competitive result on the MultiNLI dataset. On SciTail, our proposed CAFE model achieves state-of-the-art performance. The performance gain over strong baselines such as DecompAtt and ESIM are $\approx 10\% - 13\%$ in terms of accuracy. CAFE also outperforms DGEM, which uses a graph-based attention for improved performance, by a significant margin of 5%. As such, empirical results demonstrate the effectiveness of our proposed CAFE model on the challenging SciTail dataset.

²As of 22nd May 2018, the deadline of the EMNLP submission.

TABLE 3.3: Ablation study on MultiNLI development sets.

	Match	Mismatch
Original Model	79.0	78.9
(1a) Rm FM for 1L-FC	77.7	77.9
(1b) Rm FM for 1L-FC (ReLU)	77.3	77.5
(1c) Rm FM for 2L-FC (ReLU)	76.6	76.4
(2) Remove Char Embed	78.1	78.3
(3) Remove Syn Embed	78.3	78.4
(4) Remove Inter Att	75.2	75.6
(5) Replace HW Pred. with FC	77.7	77.9
(6) Replace HW Enc. with FC	78.7	78.7
(7) Remove Sub Feat	77.9	78.3
(8) Remove Mul Feat	78.7	78.6
(9) Remove Concat Feat	77.9	77.6
(10) Add Bi-directional	78.3	78.4

3.4.4 Ablation Study

Table 3.3 reports ablation studies on the MultiNLI development sets (HW stands for Highway). In (1), we replaced all FM functions with regular fully-connected (FC) layers in order to observe the effect of **FM versus FC**. More specifically, We experimented with several FC configurations as follows: (a) 1-layer linear, (b) 1-layer ReLU and (c) 2-layer ReLU. The 1-layer linear setting performs the best and is therefore reported in Table 3.3. Using ReLU seems to be worse than nonlinear FC layers. Overall, the best combination (option (a)) still experienced a decline in performance in both development sets.

In (2-3), we explored the utility of using character and syntactic embeddings, which we found to have helped CAFE marginally. In (4), we removed the inter-attention alignment features, which naturally impact the model performance significantly. In (5-6), we explored the effectiveness of the highway layers (in prediction layers and encoding layers) by replacing them to FC layers. We observe that both highway layers have marginally helped the overall performance. Finally, in (7-9), we removed the alignment features based on their composition type. We observe that the *Sub* and *Concat* compositions were more important than the *Mul* composition. However, removing any of the three will result in some performance degradation. Finally, in (10), we replaced the LSTM encoder with a BiLSTM, observing that adding bi-directionality did not improve performance for our model.

TABLE 3.4: Linguistic Error Analysis on MultiNLI dataset.

	Matched		Mismatched	
	ESIM	CAFE	ESIM	CAFE
Conditional	100	70	60	85
Word overlap	50	82	62	87
Negation	76	76	71	80
Antonym	67	82	58	80
Long Sentence	75	79	69	77
Tense Difference	73	82	79	89
Active/Passive	88	100	91	90
Paraphrase	89	88	84	95
Quantity/Time	33	53	54	62
Coreference	83	80	75	83
Quantifier	69	75	72	80
Modal	78	81	76	81
Belief	65	77	67	83

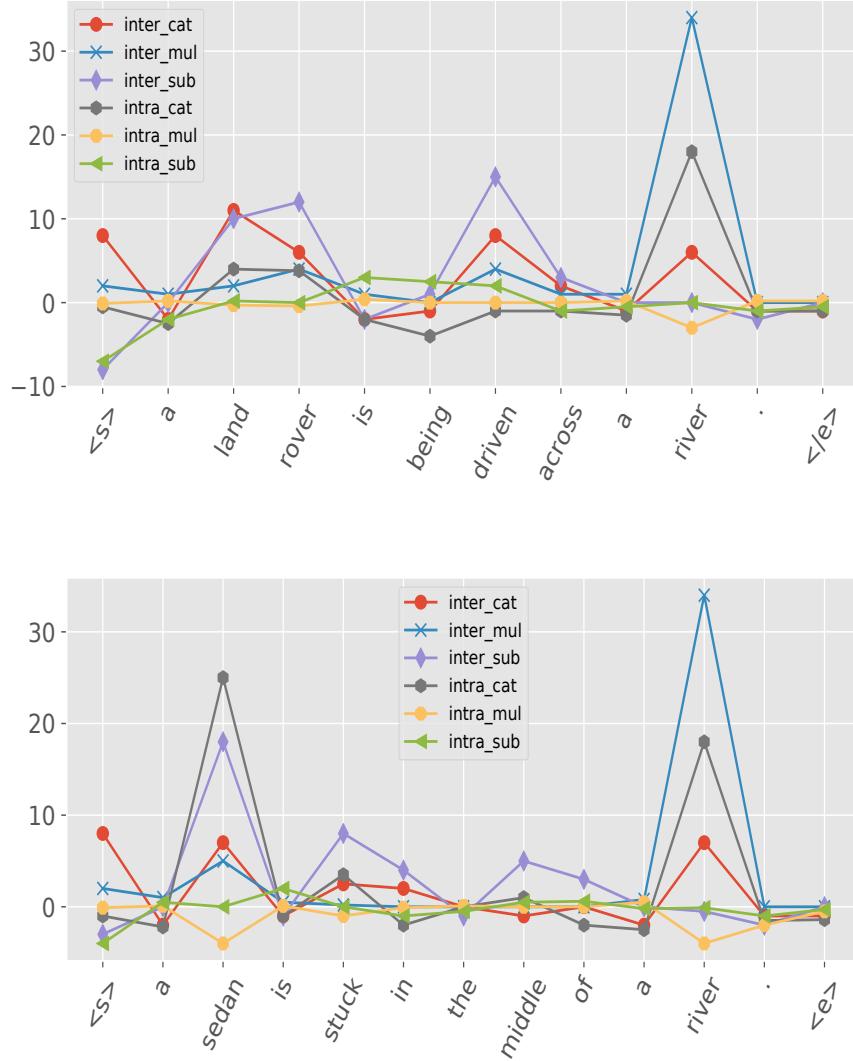
3.4.5 Linguistic Error Analysis

We perform a linguistic error analysis using the supplementary annotations provided by the MultiNLI dataset. We compare against the model outputs of the ESIM model across 13 categories of linguistic phenomena [87]. Table 3.4 reports the result of our error analysis. We observe that our CAFE model generally outperforms ESIM on *most categories*.

On the mismatched setting, CAFE outperforms ESIM in 12 out of 13 categories, losing only in one percentage point in *Active/Passive* category. On the matched setting, CAFE is outperformed by ESIM very marginally on coreference and paraphrase categories. Despite generally achieving much superior results, we noticed that CAFE performs poorly on the matched setting. However, this only accounts for 5% of samples. Measuring the absolute ability of CAFE, we find that CAFE performs extremely well in handling linguistic patterns of *paraphrase detection* and *active/passive*. This is likely to be attributed by the alignment strategy that both CAFE and ESIM exploit.

3.4.6 Interpreting and Visualizing with CAFE

Finally, we also observed that the propagated features are highly interpretable, giving insights to the inner workings of the CAFE model. Figure 3.3 shows a visu-

FIGURE 3.3: Visualization of *six* propagated features.

alization of the feature values from an example in the SNLI test set. The ground truth is *contradiction*. Based on the above example, we make several observations. Firstly, *inter_mul* features mostly capture identical words (or semantically similar words), i.e., *inter_mul* features for ‘*river*’ spike in both sentences. Secondly, *inter_sub* spikes on conflicting words that might cause contradiction, e.g., ‘*sedan*’ and ‘*land rover*’ are not the same vehicle. Another interesting observation is that we notice the *inter_sub* features for *driven* and *stuck* spiking. This also validates the observation of [76], which shows what the *sub* vector in the ESIM model is looking out for contradictory information. However, our architecture allows the inspection of these vectors since they are compressed via factorization, leading to

larger extents of explainability - a quality that neural models inherently lack. We also observe that intra-attention (e.g., intra_cat) features seem to capture the more important words in the sentence (i.e., ‘*river*’, ‘*sedan*’ and ‘*land rover*’).

3.5 Experiments (CSRAN)

3.5.1 Experimental Setup

In this section, we present extended experiments of CSRAN on SNLI and SciTail. For the sake of completeness, we also run additional experiments on Quora NLI dataset. Paraphrase detection problems are strongly related to Natural Language Inference and can be considered to be a ‘*child*’ problem of the NLI family.

Quora Question NLI It is a well-studied paraphrase identification dataset³. We use the splits provided by [149]. The task is to determine if two questions are paraphrases of each other. This task is formulated as a binary classification problem. We compare with L.D.C [103], BiMPM, the DecompAtt implementation by [158] (word and char level) and DIIN. All baselines are reported from the respective papers. All models are trained with the Adam optimizer [153] with learning rates tuned amongst {0.001, 0.0003, 0.0004}. Batch size is tuned amongst {32, 64, 128, 256}. The dimensions of the BiLSTM encoders are tuned amongst {64, 100, 200, 300} and the number of hidden dimensions of the prediction layers is tuned amongst {100, 200, 300, 600}. The number of stacked recurrent layers is tuned from [2, 5] and the number of aggregation BiLSTM layers is tuned amongst {1, 2}. The number of prediction layers is tuned from [1, 3]. Parameters are initialized using glorot uniform [159]. All unspecified activation functions are ReLU activations. Word embeddings are initialized with GloVe [16] and fixed during training. We implement our model in Tensorflow [152] and use the cuDNN implementation for all BiLSTM layers.

³<https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs>

TABLE 3.5: Results (CSRAN) on single model SNLI dataset.

Model	Acc
BiMPM [149]	87.5
ESIM [76]	88.0
DIIN [81]	88.0
DR-BiLSTM [160]	88.5
CAFE	88.5
CSRAN	88.7

TABLE 3.6: Results (CSRAN) on SciTail dataset.

Model	Acc
DecompAtt [21]	72.3
ESIM [76]	70.6
DGEM [84]	77.3
CAFE	83.3
CSRAN	86.7

TABLE 3.7: Results (CSRAN) on Quora NLI dataset.

Model	Acc
L.D.C [103]	87.5
Word DecompAtt [158]	87.5
BiMPM [149]	88.1
Char DecompAtt [158]	88.4
CAFE	88.5
DIIN [81]	89.0
CSRAN	89.2

3.5.2 Experimental Results

On SNLI (Table 3.5), CSRAN achieves the best single model performance on the well-established dataset. This demonstrates the effectiveness of CSRAN, taking into consideration of the inherent competitiveness of this well-known benchmark. On SciTail (Table 3.6), CSRAN similarly achieves the best performance to date on this dataset, outperforming the existing CAFE model by +3.4% absolute accuracy. We validate that CSRAN can indeed improve the performance of CAFE.

On Quora NLI (Table 3.7), CSRAN also achieves the best single model score, outperforming strong baselines such as BiMPM (+1.1%) and DIIN (+0.2%). Similarly, CSRAN can further improve the performance of CAFE.

3.6 Summary

In this chapter, we proposed a new neural architecture, CAFE for NLI. CAFE achieves very competitive performance on three benchmark datasets. CAFE was state-of-the-art on the well-contested SNLI leaderboard⁴ from December 2017 to May 2018. Extensive ablation studies confirm the effectiveness of FM layers over FC layers. Qualitatively, we show how different compositional operators (e.g., sub and mul) behave in NLI task and shed light on why subtractive composition helps in other models such as ESIM. Additionally, we proposed an extension of CAFE, i.e., the CSRAN model. CSRAN uses co-stacking for improved gradient flow and CAFE blocks between stacked recurrent encoders. Overall, both models helped push and advance the state-of-the-art in NLI research.

⁴<https://nlp.stanford.edu/projects/snli/>

Chapter 4

Multi-Cast Attention Networks for Retrieval-based Natural Language Understanding

Modeling textual relevance between document query pairs lives at the heart of IR-based NLU research. Intuitively, this enables a wide assortment of real life applications, ranging from standard web search to automated chatbots. In this chapter¹, we discuss our proposed Multi-Cast Attention Networks for retrieval-based NLU.

4.1 Introduction

The key idea of retrieval-based NLU systems is to learn a scoring function between document-query pairs, providing a ranked list of candidates as an output. A considerable fraction of such IR systems are focused on short textual documents, e.g., answering facts based questions or selecting the best response in the context of a chat-based system. The application of retrieval-based response and question answering (QA) systems is overall versatile, potentially serving as a powerful standalone domain-specific system or a crucial component in larger, general purpose chat systems such as Alexa. This chapter presents a universal neural ranking model for such tasks.

¹This chapter is published as *Multi-Cast Attention Networks*, Proceedings of KDD 2018 [61]

Neural networks (or deep learning) have garnered considerable attention for retrieval-based systems [30, 50, 161–163]. Notably, the dominant state-of-the-art systems for many benchmarks are now neural models, almost completely dispensing with traditional feature engineering techniques altogether. In these systems, convolutional or recurrent networks are empowered with recent techniques such as neural attention [22, 23, 161], achieving very competitive results on standard benchmarks. The key idea of attention is to extract only the most relevant information that is useful for prediction. In the context of textual data, attention learns to weight words and sub-phrases within documents based on how important they are. In the same vein, co-attention mechanisms [94, 100, 101, 111] are a form of attention mechanisms that learn joint pairwise attentions, with respect to both document and query.

Typically, attention is applied once to a sentence [22, 161]. A final representation is learned, and then passed to prediction layers. In the context of handling sequence pairs, co-attention is applied and a final representation for each sentence is learned [94, 100, 101]. An obvious drawback which applies to many existing models is that they are generally restricted to one attention variant. In the case where one or more attention calls are used (e.g., co-attention and intra-attention, etc.), concatenation is generally used to fuse representations [21, 94]. Unfortunately, this incurs cost in subsequent layers by doubling the representation size per call.

The rationale for desiring more than one attention call is intuitive. In [21, 94], Co-Attention and Intra-Attention are both used because each provides a different view of the document pair, learning high quality representations that could be used for prediction. Hence, this can significantly improve performance. Moreover, Co-Attention also comes in different flavors and can either be used with extractive max-mean pooling [100, 101] or alignment-based pooling [21, 76, 94]. Each co-attention type produces different document representations. In max-pooling, signals are extracted based on a word’s *largest* contribution to the other text sequence. Mean-pooling calculates its contribution to the overall sentence. Alignment-pooling is another flavor of co-attention, which aligns semantically similar sub-phrases together. As such, different pooling operators provide a different view of sentence pairs. This is often tuned as a hyperparameter, i.e., performing architectural engineering to find the best variation that works best on each problem domain and dataset.

Our approach is targeted at serving two important purposes: (1) It removes the need for architectural engineering of this component by enabling attention to be called for an arbitrary k times with hardly any consequence and (2) concurrently it improves performance by modeling multiple views via multiple attention calls. As such, our method is in a similar spirit to multi-headed attention, albeit efficient. To this end, we propose *Multi-Cast Attention Networks* (MCAN), a new deep learning architecture for a potpourri of tasks in the question answering and conversation modeling domains. In our approach, attention is **casted**, in contrast to the most other works that use it as a pooling operation. We cast co-attention multiple times, each time returning a compressed **scalar** feature that is re-attached to the original word representations. The key intuition is that compression enables scalable casting of multiple attention calls, aiming to provide subsequent layers with a *hint* of not only global knowledge but also cross sentence knowledge. Intuitively, when passing these enhanced embeddings into a compositional encoder (such as a long short-term memory encoder), the LSTM can then benefit from this hint and alter its representation learning process accordingly.

In summary, the prime contributions of this work are:

- For the first time, we propose a new paradigm of utilizing attentions not as a pooling operator but as a form of feature augmentation. We propose an overall architecture, Multi-Cast Attention Networks (MCAN) for generic sequence pair modeling.
- We evaluate our proposed model on four benchmark tasks, i.e., Dialogue Reply Prediction (Ubuntu dialogue corpus), Factoid Question Answering (TrecQA), Community Question Answering (QatarLiving forums from SemEval 2016) and Tweet Reply Prediction (customer support). On Ubuntu dialogue corpus, MCAN outperforms the existing state-of-the-art models by 9%. MCAN also achieves the best performing score of 0.838 MAP and 0.904 MRR on the well-studied TrecQA dataset.
- We provide a comprehensive and in-depth analysis of the inner workings of our proposed MCAN model. We show that the casted attention features are interpretable and are capable of learning (1) a neural adaptation of word overlap and (2) a differentiation of evidence and anti-evidence words/patterns.

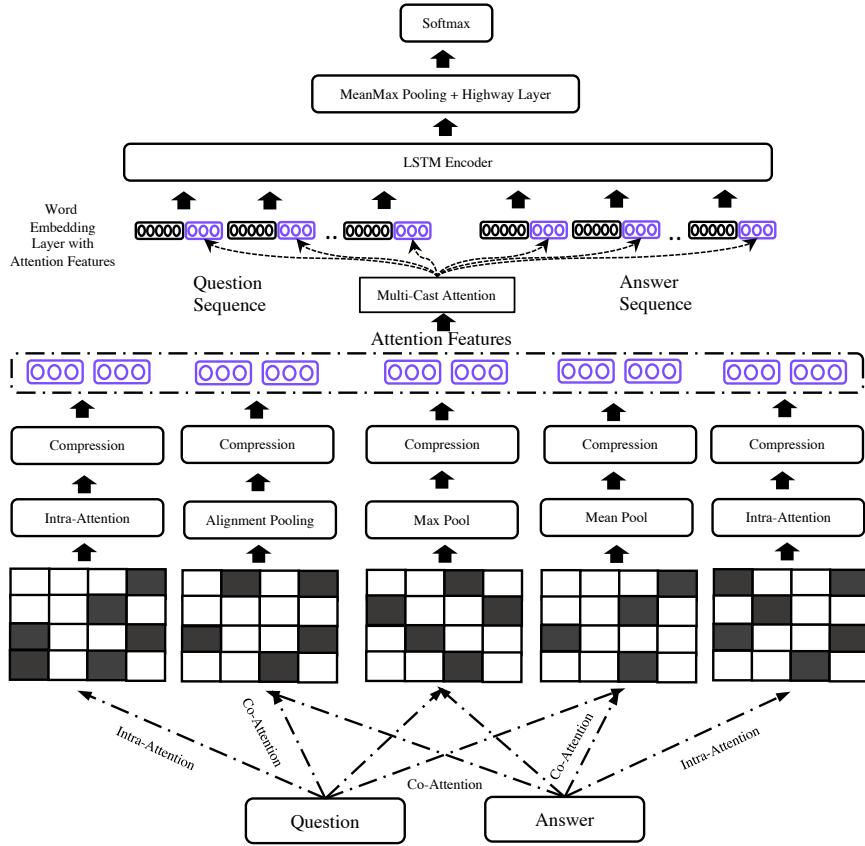


FIGURE 4.1: Architecture of Multi-Cast Attention Networks.

4.2 Proposed Method

In this section, we describe our proposed MCAN model. The inputs to our model are two text sequences which we denote as query q and document d . In our problem, query-document can be generalizable to different problem domains such as question-answering or message-response prediction. Figure 4.1 illustrates the overall model architecture for question-answer retrieval. MCAN is a wide multi-headed attention architecture that utilizes compression functions and attention as features. The given example is for question-answer retrieval. Note that the Input Encoding layer is omitted for clarity.

4.2.1 Input Encoder

The document and query inputs are passed in as one-hot encoded vectors. A word embedding layer parameterized by $W_e \in \mathbb{R}^{d \times |V|}$ converts each word to a dense word

representation $w \in \mathbb{R}^d$. V is the set of all words in the vocabulary.

Highway Encoder Each word embedding is passed through a highway encoder layer. Highway networks [146] are gated nonlinear transform layers which control information flow to subsequent layers. Many works adopt a projection layer that is trained in place of the raw word embeddings. Not only does this save computation cost but also reduces the number of trainable parameters. Our work extends this projection layer to use a highway encoder. The intuition for doing so is simple, i.e., highway encoders can be interpreted as data-driven word filters. As such, we can imagine them to parametrically learn which words have an inclination to be important and not important to the task at hand, for example, filtering stop words and words that usually do not contribute much to the prediction. Similar to recurrent models that are gated in nature, this highway encoder layer controls how much information (of each word) is flowed to the subsequent layers.

Let $H(\cdot)$ and $T(\cdot)$ be single layered affine transforms with ReLU and sigmoid activation functions respectively. A single highway network layer is defined as:

$$y = H(x, W_H) \cdot T(x, W_T) + (1 - T(x, W_T)) \cdot x \quad (4.1)$$

where $W_H, W_T \in \mathbb{R}^{r \times d}$. Notably, the dimensions of the affine transform might be different from the size of the input vector. In this case, an additional nonlinear transform is used to project x to the same dimensionality.

4.2.2 Co-Attention

Co-Attention [111] is a pairwise attention mechanism that enables attending to text sequence pairs jointly. In this section, we introduce four variants of attention, i.e., (1) max-pooling, (2) mean-pooling, (3) alignment-pooling, and finally (4) intra-attention (or self-attention). The first step in co-attention is to learn an affinity (or similarity) matrix between each word across both sequences. Following Parikh et al. [21], we adopt the following formulation for learning the affinity matrix.

$$s_{ij} = F(q_i)^T F(d_j) \quad (4.2)$$

where $F(\cdot)$ is a function such as a multi-layered perceptron (MLP). Alternate forms of co-attention are also possible such as $s_{ij} = q_i^\top M d_j$ and $s_{ij} = F([q_i; d_j])$.

Extractive Pooling The most common variant of extractive co-attention is the *max-pooling* co-attention, which attends to each word based on its maximum influence it has on the other text sequence.

$$q' = \text{Soft}(\max_{\text{col}}(s))^\top q \quad \text{and} \quad d' = \text{Soft}(\max_{\text{row}}(s))^\top d \quad (4.3)$$

where q' , d' are the co-attentive representations of q and d respectively. $\text{Soft}(\cdot)$ is the Softmax operator. Alternatively, the mean row and column-wise pooling of matrix s can be also used:

$$q' = \text{Soft}(\text{mean}_{\text{col}}(s))^\top q \quad \text{and} \quad d' = \text{Soft}(\text{mean}_{\text{row}}(s))^\top d \quad (4.4)$$

However, each pooling operator has different impacts and can be intuitively understood as follows: max-pooling selects each word based on its maximum importance of all words in the other text. Mean-pooling is a more *wholesome* comparison, paying attention to a word based on its overall influence on the other text. This is usually dataset-dependent, regarded as a hyperparameter and is tuned to see which performs best on the held out set.

Alignment-Pooling Soft alignment-based pooling has also been utilized for learning co-attentive representations [21]. However, the key difference with soft alignment is that it *realigns* sequence pairs while standard co-attention simply learns to weight and score important words. The co-attentive representations are then learned as follows:

$$d'_i := \sum_{j=1}^{\ell_q} \frac{\exp(s_{ij})}{\sum_{k=1}^{\ell_q} \exp(s_{ik})} q_j \quad \text{and} \quad q'_j := \sum_{i=1}^{\ell_d} \frac{\exp(s_{ij})}{\sum_{k=1}^{\ell_d} \exp(s_{kj})} d_i \quad (4.5)$$

where d'_i is the sub-phrase in q that is softly aligned to d_i . Intuitively, d'_i is a weighted sum across $\{q_j\}_{j=1}^{\ell_q}$, selecting the most relevant parts of q to represent d_i .

Intra-Attention Intra-Attention, or Self-Attention was recently proposed to learn representations that are aware of long-term dependencies. This is often formulated as a co-attention (or alignment) operation with respect to itself. In this case, we apply intra-attention to both document and query independently. For notational simplicity, we refer to them as x instead of q or d here. The Intra-Attention function is defined as:

$$x'_i := \sum_{j=1}^{\ell} \frac{\exp(s_{ij})}{\sum_{k=1}^{\ell} \exp(s_{ik})} x_j \quad (4.6)$$

where x'_i is the intra-attentional representation of x_j .

4.2.3 Multi-Cast Attention

At this point, it is easy to make several observations. Firstly, each attention mechanism provides a different flavor to the model. Secondly, attention is used to alter the original representation either by re-weighting or realigning. As such, most neural architectures only make use of one type of co-attention or alignment function [21, 100]. However, this requires the right model architecture to be tuned and potentially missing out from the benefits brought by using multiple variations of co-attention mechanism. As such, our work casts each attention operation as a **word-level** feature.

Casted Attention Let x be either q or d and \bar{x} is the representation² of x after applying co-attention or soft attention alignment. The attention features for the co-attention operators are:

$$f_c = F_c([\bar{x}; x]) \quad (4.7)$$

$$f_m = F_c(\bar{x} \odot x) \quad (4.8)$$

$$f_s = F_c(\bar{x} - x) \quad (4.9)$$

where \odot is the Hadamard product and $[.; .]$ is the concatenation operator. $F_c(.)$ is a compression function used to reduce features to a scalar. Intuitively, what is achieved here is that we are modeling the influence of co-attention by comparing

²We omit subscripts for clarity.

representations before and after co-attention. For soft-attention alignment, a critical note here is that x and \bar{x} (though of equal lengths) have ‘*exchanged*’ semantics. In other words, in the case of q , \bar{q} actually contains the aligned representation of d . Finally, the usage of multiple comparison operators (subtractive, concatenation and multiplicative operators) is to capture multiple perspectives and is inspired by the ESIM model [76].

Compression Function This section defines the compression function $F_c(\cdot)$ to be used. The rationale for compression is simple and intuitive - we do not want to *bloat* subsequent layers with a high dimensional vector which consequently incurs parameter costs in subsequent layers. We investigate the usage of three compression functions, which are capable of reducing a n -dimensional vector to a scalar.

- **Sum (SM)** Function is a non-parameterized function that sums the entire vector, returning a scalar as an output.

$$F(x) = \sum_i^n x_i , \quad \forall x_i \in x \quad (4.10)$$

- **Neural Network (NN)** is a fully-connected layer that converts each n -dimensional feature vector as follows:

$$F(x) = \text{ReLU}(W_c(x) + b_c). \quad (4.11)$$

where $W_c \times \mathbb{R}^{n \times 1}$ and $b_c \in \mathbb{R}$ are the parameters of the FC layer.

- **Factorization Machines (FM)** are general purpose machine learning techniques that accept a real-valued feature vector $x \in \mathbb{R}^n$ and return a scalar output.

$$F(x) = w_0 + \sum_{i=1}^n w_i x_i + \sum_{i=1}^n \sum_{j=i+1}^n \langle v_i, v_j \rangle x_i x_j \quad (4.12)$$

where $w_0 \in \mathbb{R}$, $w_i \in \mathbb{R}^n$ and $\{v_1, \dots, v_n\} \in \mathbb{R}^{n \times k}$ are the parameters of the FM model. FMs are expressive models that capture pairwise interactions between features using factorized parameters. k is the number of factors of the FM model. For more details, we refer interested readers to [144].

Note that we do not share parameters across multiple attention casts because each attention cast is aimed at modeling a different view. Our experiments report the above mentioned variants under the model name MCAN (SM), MCAN (NN) and MCAN (FM) respectively.

4.2.4 Multi-Cast

The key idea behind our architecture is the facilitation of k attention calls (or casts), with each cast augmenting raw word embeddings with a real-valued attentional hint. We formally describe the Multi-Cast Attention mechanism. For each query-document pair, we apply (1) Co-Attention with mean-pooling, (2) Co-Attention with max-pooling and (3) Co-Attention with alignment-pooling. Additionally, we apply Intra-Attention to both query and document individually. Each attention cast produces three scalars (per word) which are concatenated with the word embedding. The final casted feature vector is $z \in \mathbb{R}^{12}$. As such, for each word w_i , the new word representation becomes $\bar{w}_i = [w_i; z_i]$.

4.2.5 LSTM Encoder

Next, the word representations with casted attention $\bar{w}_1, \bar{w}_2, \dots, \bar{w}_\ell$ are then passed into a sequential encoder layer. We adopt a standard vanilla long short-term memory (LSTM) encoder:

$$h_i = LSTM(u, i), \forall i \in [1, \dots, \ell] \quad (4.13)$$

where ℓ represents the maximum length of the sequence. Notably, the parameters of the LSTM are ‘siamese’ in nature, sharing weights between document and query. The key idea is that the LSTM encoder learns representations that are aware of sequential dependencies by the usage of nonlinear transformations as gating functions. Since LSTMs are standard neural building blocks, we omit technical details in favor of brevity. As such, the key idea behind casting attention as features right before this layer is that it provides the LSTM encoder with hints that provide information such as (1) long-term and global sentence knowledge and (2) knowledge between sentence pairs (document and query).

4.2.6 Pooling Operation

Finally, a pooling function is applied across the hidden states $\{h_1 \dots h_\ell\}$ of each sentence, converting the sequence into a fixed dimensional representation.

$$h = \text{MeanMax}[h_1 \dots h_\ell] \quad (4.14)$$

We adopt the *MeanMax* pooling operator, which concatenates the result of the mean pooling and max pooling together. We found this to consistently perform better than using *max* or *mean* pooling in isolation.

4.2.7 Prediction Layer and Optimization

Finally, given a fixed dimensional representation of the document-query pair, we pass their concatenation into a two-layer h -dimensional highway network. The final prediction layer of our model is computed as follows:

$$y_{out} = H_2(H_1([x_q; x_d; x_q \odot x_d; x_q - x_d])) \quad (4.15)$$

where $H_1(\cdot), H_2(\cdot)$ are highway network layers with ReLU activation. The output is then passed into a final linear softmax layer.

$$y_{pred} = \text{softmax}(W_F \cdot y_{out} + b_F) \quad (4.16)$$

where $W_F \in \mathbb{R}^{h \times 2}$ and $b_F \in \mathbb{R}^2$. The network is then trained using standard multi-class cross entropy loss with L2 regularization.

$$J(\theta) = - \sum_{i=1}^N [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)] + \lambda \|\theta\|_{L2} \quad (4.17)$$

where θ are the parameters of the network. \hat{y} is the output of the network. $\|\theta\|_{L2}$ is the L2 regularization and λ is the weight of the regularizer.

4.2.8 Relation to CAFE

Without the max and mean pooling attention layers, MCAN reduces to the CAFE model. However, the max and mean pooling are characteristic of many IR-based NLU tasks. Therefore, they are critical to the overall MCAN framework. Moreover, MCAN demonstrates the potential of ‘*multi-casting*’ and is essentially a multi-casted variation of CAFE introduced in Chapter 3. This chapter also showcases the effectiveness of a CAFE-like architecture on a suite of IR-based NLU tasks. MCAN equips with extractive pooling layers, which makes it critical for IR-based NLU.

4.3 Experiments

This section describes and reports our experimental results. The experiments are conducted based on four retrieval-based NLU tasks, i.e., Dialogue Prediction, Factoid Question Answering, Community Question Answering and Tweet Reply Prediction.

4.3.1 Dialogue Prediction

In this first task, we evaluate our model on its ability to successfully predict the next reply in conversations.

Dataset and Evaluation Metric For this experiment, we utilize the large and well-known large-scale Ubuntu Dialogue Corpus (UDC) [56]. We use the same testing splits provided by Xu et al. [164]. In this task, the goal is to match a sentence with its reply. Following [55], the task mainly utilizes the last two utterances in each conversation, predicting if the latter follows the former. The training set consists of **one million** message-response pairs with a 1 : 1 positive-negative ratio. The development and testing sets have a 9 : 1 ratio. Following [55, 164], we use the evaluation metric of recall@ k ($R_n@K$) which indicates whether the ground truth exists in the top k results from n candidates. The four evaluation metrics used are $R_2@1$, $R_{10}@1$, $R_{10}@2$ and $R_{10}@5$.

TABLE 4.1: Results on Ubuntu Dialogue Corpus.

	$R_2@1$	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$
MLP	0.651	0.256	0.380	0.703
DeepMatch	0.593	0.345	0.376	0.693
ARC-I	0.665	0.221	0.360	0.684
ARC-II	0.736	0.380	0.534	0.777
CNTN	0.743	0.349	0.512	0.797
MatchPyramid	0.743	0.420	0.554	0.786
LSTM	0.725	0.361	0.494	0.801
AP-LSTM	0.758	0.381	0.545	0.801
MV-LSTM	0.767	0.410	0.565	0.800
KEHNN	0.786	0.460	0.591	0.819
MCAN (SM)	0.831	0.548	0.682	0.873
MCAN (NN)	<u>0.833</u>	<u>0.549</u>	0.686	0.875
MCAN (FM)	0.834	0.551	<u>0.684</u>	0.875

Implementation Details We compare against a large number of competitive baselines, e.g., MLP, DeepMatch [165], ARC-I / ARC-II [90], CNTN [95], Match-Pyramid [97], vanilla LSTM, Attentive Pooling LSTM [100], MV-LSTM [96] and finally the state-of-the-art Knowledge Enhanced Hybrid Neural Network (KEHNN) [55]. A detailed description of baselines can be found at [55]. Since testing splits are the same, we report the results directly from [55]. For fair comparison, we set the LSTM encoder size in MCAN to $d = 100$ which makes it equal to the models in [55]. We optimize MCAN with Adam optimizer [153] with an initial learning rate of 3×10^{-4} . A dropout rate of 0.2 is applied to all layers except the word embedding layer. The sequences are dynamically truncated or padded to their batch-wise maximums (with a hard limit of 50 tokens). We initialize the word embedding layer with pretrained GloVe embeddings.

Results Table 4.1 reports the results of our experiments. Clearly, we observe that all MCAN models achieve a huge performance gain over existing state-of-the-art models. More specifically, the improvement across all metrics are $\approx 5\% - 9\%$ better than KEHNN. The performance improvement over strong baselines such as AP-LSTM and MV-LSTM are even greater, hitting an improvement of 15% in terms of $R_{10}@1$. This ascertains the effectiveness of the MCAN model. Overall, MCAN (FM) and MCAN (NN) are comparable in terms of performance. MCAN (SM) is marginally lower than both MCAN (FM) and MCAN (NN). However, its performance is still considerably higher than the existing state-of-the-art models.

4.3.2 Factoid Question Answering

Factoid question answering is the task of answering factual based questions. In this task, the goal is to provide a ranked list of answers to a given question.

Dataset and Evaluation Metric We utilize the QA dataset from TREC (Text Retrieval Conference). TrecQA is one of the most widely evaluated datasets, competitive and long standing benchmark for QA. This dataset was prepared by Wang et al. [166] and contains 53K QA pairs for training and 1100/1500 pairs for development and testing respectively. Following the recent works, we evaluate on the clean setting as noted by [167]. The evaluation metrics for this task are the MAP (mean average precision) and MRR (mean reciprocal rank) scores which are well-known IR metrics.

Implementation Details We compare against all previously published works on this dataset. The competitive baselines for this task are QA-LSTM/AP-CNN [100], LDC model [103], MPCNN [98], MPCNN+NCE [167], HyperQA [106], BiMPM [80] and IWAN [94]. For our model, the size of the LSTM used is 300. The dimensions of the highway prediction layer is 200. We use the Adam optimizer with a 3×10^{-4} learning rate. The L2 regularization is set to 10^{-6} . A dropout rate of 0.2 is applied to all layers except the embedding layer. We use pretrained 300d GloVe embeddings and fix the embeddings during training. For MCAN (FM), we use a FM model with 10 factors. We pad all sequences to the maximum sequence length and truncate them to the batch-wise maximums.

Results Table 4.2 reports our results on TrecQA. All MCAN variations outperform all existing state-of-the-art models. Notably, MCAN (FM) is currently the best performing model on this extensively studied dataset. MCAN (NN) comes in second which marginally outperforms the highly competitive and recent IWAN model. Finally, MCAN (SM) remains competitive to IWAN, despite naively summing over casted attention features.

TABLE 4.2: Results on TrecQA (*clean*) dataset.

Model	MAP	MRR
QA-LSTM (dos Santos et al.)	0.728	0.832
AP-CNN (dos Santos et al.)	0.753	0.851
LDC Model (Wang et al.)	0.771	0.845
MPCNN (He et al.)	0.777	0.836
HyperQA (Tay et al.)	0.784	0.865
MPCNN + NCE (Rao et al.)	0.801	0.877
BiMPM (Wang et al.)	0.802	0.899
IWAN (Shen et al.)	0.822	0.889
MCAN (SM)	0.827	0.880
MCAN (NN)	<u>0.827</u>	<u>0.890</u>
MCAN (FM)	0.838	0.904

4.3.3 Community Question Answering (cQA)

This task is concerned with ranking answers in community forums. Different from factoid QA, answers are generally subjective instead of factual. Moreover, answer lengths are also much longer.

Dataset and Evaluation We use the QatarLiving dataset, a well-studied benchmark dataset from SemEval-2016 Task 3 Subtask A (cQA) and has been used extensively as a benchmark for recent state-of-the-art neural network models for cQA [101, 104]. This is a real-world dataset obtained from Qatar Living Forums and comprises $36K$ training pairs, $2.4K$ development pairs and $3.6K$ testing pairs. In this dataset, there are ten answers in each question ‘thread’ which are marked as ‘Good’, ‘Potentially Useful’ or ‘Bad’. Following [101], ‘Good’ is regarded as positive and anything else is regarded as negative labels. We evaluate on two metrics, namely the Precision@1 (P@1) and Mean Average Precision (MAP) metric.

Implementation Details The key competitors of this dataset are the CNN-based ARC-I/II architecture by Hu et al. [90], the Attentive Pooling CNN [100], Kelp [168] a feature engineering based SVM method, ConvKN [169] a combination of convolutional tree kernels with CNN and finally AI-CNN (Attentive Interactive CNN) [101], a tensor-based attentive pooling neural model. We also compare with the Cross Temporal Recurrent Networks (CTRN) [104], a recently proposed model for ranking QA pairs which has achieved very competitive performance on this

TABLE 4.3: Results on QatarLiving dataset.

Model	P@1	MAP
ARC-I (Hu et al.)	0.741	0.771
ARC-II (Hu et al.)	0.753	0.780
AP-CNN (dos Santos et al.)	0.755	0.771
Kelp (Filice et al.)	0.751	0.792
ConvKN (Barron Cedeno et al.)	0.755	0.777
AI-CNN (Zhang et al.)	0.763	0.792
CTRN (Tay et al.)	0.788	<u>0.794</u>
MCAN (SM)	<u>0.803</u>	0.787
MCAN (NN)	0.802	0.784
MCAN (FM)	0.804	0.803

dataset. Following [104], we initialize MCAN with domain-specific 200 dimensional word embeddings using the unannotated QatarLiving corpus. Word embeddings are not updated during training. The size of the highway projection layer, LSTM encoder and highway prediction layer are all set to 200. The model is optimized with Adam optimizer with learning rate of 3×10^{-4} .

Results Table 4.3 reports the performance comparison on the QatarLiving dataset. Our best performing MCAN model achieves state-of-the-art performance on this dataset. Performance improvement over recent, competitive neural network baselines is significant. Notably, the improvement of MCAN (FM) over AI-CNN on the $P@1$ metric is 4.1% and 1.1% in terms of MAP. MCAN (FM) also achieves competitive results relative to the CTRN model. The performance of MCAN (NN) and MCAN (SM) is lower than MCAN (FM) but still remains competitive on this benchmark.

4.3.4 Tweet Reply Prediction

This experiment is concerned with predicting an appropriate reply given a tweet.

Dataset and Evaluation Metrics We utilize a customer support dataset obtained from Kaggle³. This dataset contains tweet-response pairs of tweets to famous brands and their replies. For each Tweet-Reply pair, we randomly select four

³<https://www.kaggle.com/soaxelbrooke/customer-support-on-twitter>

TABLE 4.4: Results on Tweets dataset.

Model	MRR	P@1
CBOW + MLP	0.658	0.442
LSTM	0.652	0.431
CNN	0.657	0.441
AP-CNN (dos Santos et al.)	0.643	0.426
AI-CNN (Zhang et al.)	0.675	0.465
AP-BiLSTM (dos Santos et al.)	0.725	0.540
MCAN (SM)	0.722	0.548
MCAN (NN)	<u>0.747</u>	<u>0.585</u>
MCAN (FM)	0.759	0.593

tweets as negative samples that originate from the same brand. The dataset is split into 8:1:1 train-dev-test split. The evaluation metrics for this task are MRR (Mean reciprocal rank) and Precision@1 (accuracy). Unlike previous datasets, there are no published works on this dataset. As such, we implement the baselines ourselves. We implement standard baselines such as (1) CBOW (sum embeddings) passed into a 2 layer MLP with ReLU activations, (2) standard vanilla LSTM and CNN models and (3) BiLSTM and CNN with standard Co-Attention (AP-BiLSTM, AP-CNN) following [100]. All models minimize the binary cross entropy loss (pointwise) since we found performance to be much better than using ranking loss. We also include the recent AI-CNN (Attentive Interactive CNN) which uses multi-dimensional co-attention. We set all LSTM dimensions to $d = 100$ and the number of CNN filters is 100. The CNN filter width is set to 3. We train all models with Adam optimizer with 3×10^{-4} learning rate. Word embeddings are initialized with GloVe and fixed during training. A dropout of 0.2 is applied to all layers except the word embedding layer.

Results Table 4.4 reports our results on the Tweets dataset. MCAN (FM) achieves the top performance by a significant margin. The performance of MCAN (NN) falls short of MCAN (FM), but is still highly competitive. Our best MCAN model outperforms AP-BiLSTM by 3.4% in terms of MRR and 5.3% in terms of P@1. The performance improvement of AI-CNN is even greater, i.e., 8.4% in terms of MRR and 12.5% in terms of P@1. The strongest baseline is AP-BiLSTM which significantly outperforms AI-CNN and AP-CNN.

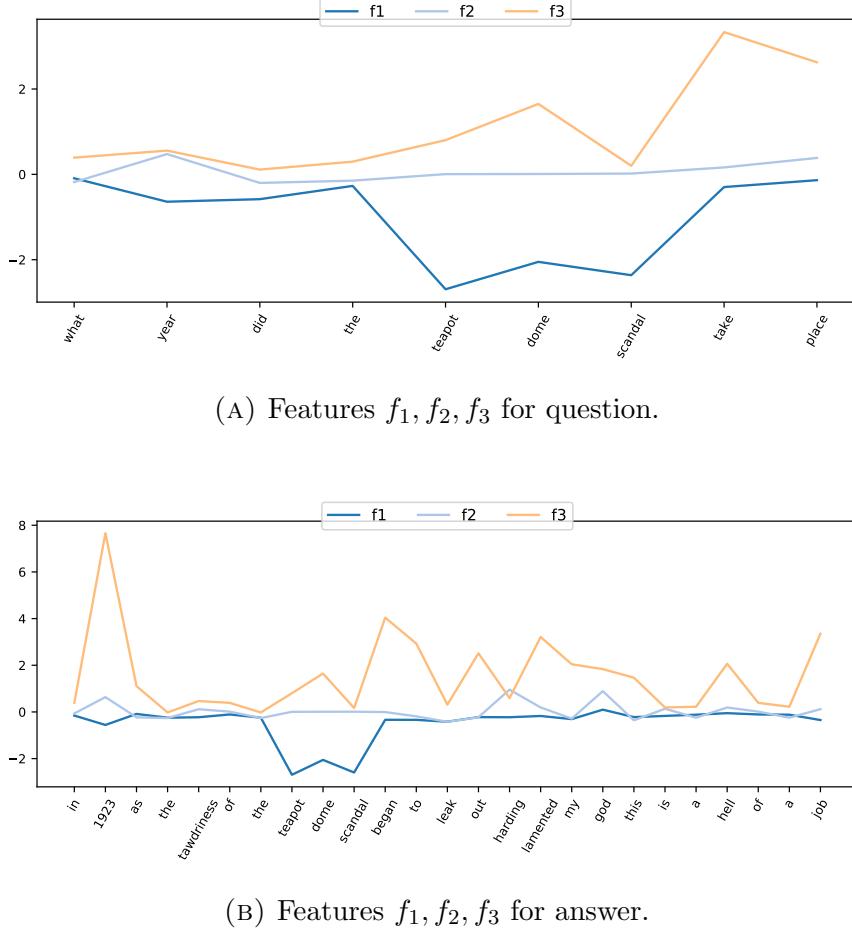
TABLE 4.5: Ablation analysis (validation set) on TrecQA dataset.

Setting	MAP	MRR
Original	0.866	0.922
(1) Remove Highway	0.825	0.863
(2) Remove LSTM	0.765	0.809
(3) Remove MCA	0.670	0.749
(4) Remove Intra	0.834	0.910
(5) Remove Align	0.682	0.726
(6) Remove Mean	0.858	0.906
(7) Remove Max	0.862	0.915

4.3.5 Ablation Analysis

This section aims to demonstrate the relative effectiveness of different components of our proposed MCAN model. Table 4.5 reports the results on the validation set of the TrecQA dataset. We report the scores of seven different configurations. In (1), we replaced all highway layers with regular feed-forward neural networks. In (2), we removed the LSTM encoder before the prediction layer. In (3), we removed the entire multi-cast attention mechanism. This is equivalent to removing the twelve attention features. In (4-7), we removed different attention casts, aiming to showcase that removing either one results in some performance drop.

From our ablation analysis, we can easily observe the crucial components to our model. Firstly, we observe that removing MCA entirely significantly decreases the performance (ablation (3)). In this case, validation MAP drops from 0.866 to 0.670. As such, our casted attention features contribute a lot to the performance of the model. Secondly, we also observe that the LSTM encoder is necessary. This is intuitive because the goal of MCAN is to provide features as hints for a compositional encoder. As such, removing the LSTM encoder allows our attention hints to go unused. While the upper prediction might still manage to learn from these features, it is still sub-optimal compared to using an LSTM encoder. Thirdly, we observe that removing Max or Mean Co-Attention decreases performance marginally. However, removing the Alignment Co-Attention decreases the performance significantly. As such, it is clear that the alignment-based attention is most important for our model. However, Max, Mean and Intra attention all contribute to the performance of MCAN. Hence, using multiple attention casts can improve performance. Finally, we also note that the highway layers also contribute slightly to performance.

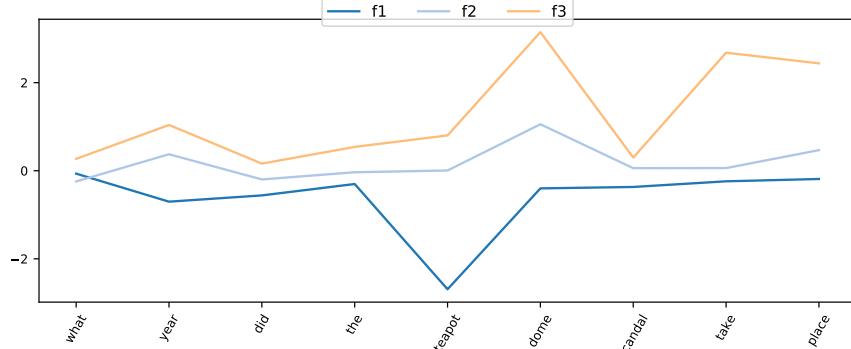
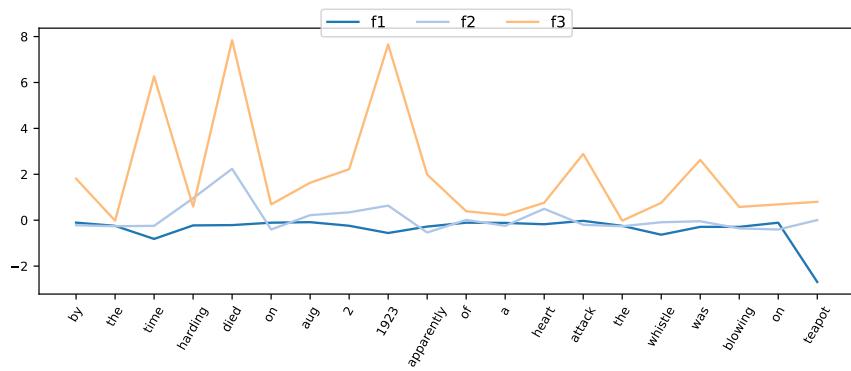
FIGURE 4.2: Casted Attention Features on a *positive* sample.

4.3.6 In-depth Model Analysis

In this section, we aim to provide insights pertaining to the inner workings of our model. More specifically, we list several observations by visual inspection of the casted attention features. We trained a MCAN model with FM compression and extracted the word-level casted attention features. The features are referred to as f_i where $i \in [1, 12]$. f_1, f_2, f_3 are generated from alignment-pooling. f_4, f_5, f_6 and f_7, f_8, f_9 are generated from max and mean pooled co-attention respectively.

Observation 1: Features learn a Neural Adaptation of Word Overlap
Figure 4.2 and Figure 4.3 show a positive and negative QA pair from the TrecQA test set. Firstly, we analyze⁴ the first three features f_1 , f_2 and f_3 . These features

⁴This is done primarily for clear visualization, lest the diagram becomes too cluttered.

(A) Features f_1, f_2, f_3 for question.(B) Features f_1, f_2, f_3 for answer.FIGURE 4.3: Casted Attention Features on a *negative* sample.

correspond to the alignment attention and multiply, subtract and concat composition respectively. From the figures, we observe that f_1 spikes (in the negative direction) when there is a word overlap across sentences, e.g., ‘teapot’ in Figure 4.3 and ‘teapot dome scandal’ in Figure 4.2. Hence, f_1 (dark blue line) behaves as a neural adaptation of the conventional overlap feature. Moreover, in contrary to traditional binary overlap features, we also notice that the value of the neural word overlap feature is dependent on the word itself, i.e., ‘teapot’ and ‘dome’ have different values. As such, it encodes more information over the traditional binary feature.

Observation 2: Features React to Evidence and Anti-Evidence While f_1 is primarily aimed at modeling overlap, we observe that f_3 tries to gather supporting evidence for the given QA pair. In Figure 4.2, the words ‘year’ and ‘1923’ have

spiked. It also tries to extract key verbs such as ‘*take place*’ (question) and ‘*began*’ (answer) which are related verbs generally used to describe events. Finally, we observe that f_2 (subtractive composition) seems to be searching for anti-evidence, i.e., a contradictory or irrelevant information. However, this appears to be more subtle as compared to f_1 and f_3 . In Figure 4.3, we note that the words ‘*died*’ and ‘*attack*’ (answer) have spiked. We find this example particularly interesting because the correct answer ‘1923’ is in fact found in the answer. However, the pair is **wrong** because the text sample refers to the ‘*death of Harding*’ and does not answer the question correctly. In the negative answer, we found that the word ‘*died*’ has the highest f_2 value. As such, we believe that f_2 is actively finding anti-evidence to why this QA pair should be negative. Additionally, irrelevant words such as ‘*attack*’ and ‘*god*’ experience nudges by f_2 . Finally, it is good to note that MCAN classifies these two samples correctly while a standard Bidirectional LSTM does not.

Observation 3: Diversity of Multiple Casts One of the key motivators for a multi-casted attention is that each attention cast produces features from different views of the sentence pair. While we have shown in our ablation study that all attention casts contributed to the overall performance, this section qualitatively analyzes the output features. Figure 4.4 shows the casted attention features (answer text) for max-pooled attention (f_4, f_5, f_6) and mean-pooled attention (f_7, f_8, f_9). Note that the values on f_7 are not constant. They appear to be since the max-min range is much smaller than f_8 and f_9 . Also, the corresponding question is the same as Figure 4.2 and Figure 4.3 which allows a direct comparison with the alignment-based attention. We observe that both attention casts produce extremely diverse features. More specifically, not only the spikes are all at different words but the overall sequential pattern is very different. We also note that the feature patterns differ a lot from alignment-based attention (Figure 4.2). While we were aiming to capture more diverse patterns, we also acknowledge that these features are much less interpretable than f_1, f_2 and f_3 . Even so, some patterns can still be interpreted, e.g., the value of f_5 is high for important words and low (negative) whenever the words are generic and unimportant such as ‘*to*’, ‘*the*’, ‘*a*’. Nevertheless, the main objective here is to ensure that these features are not learning identical patterns.

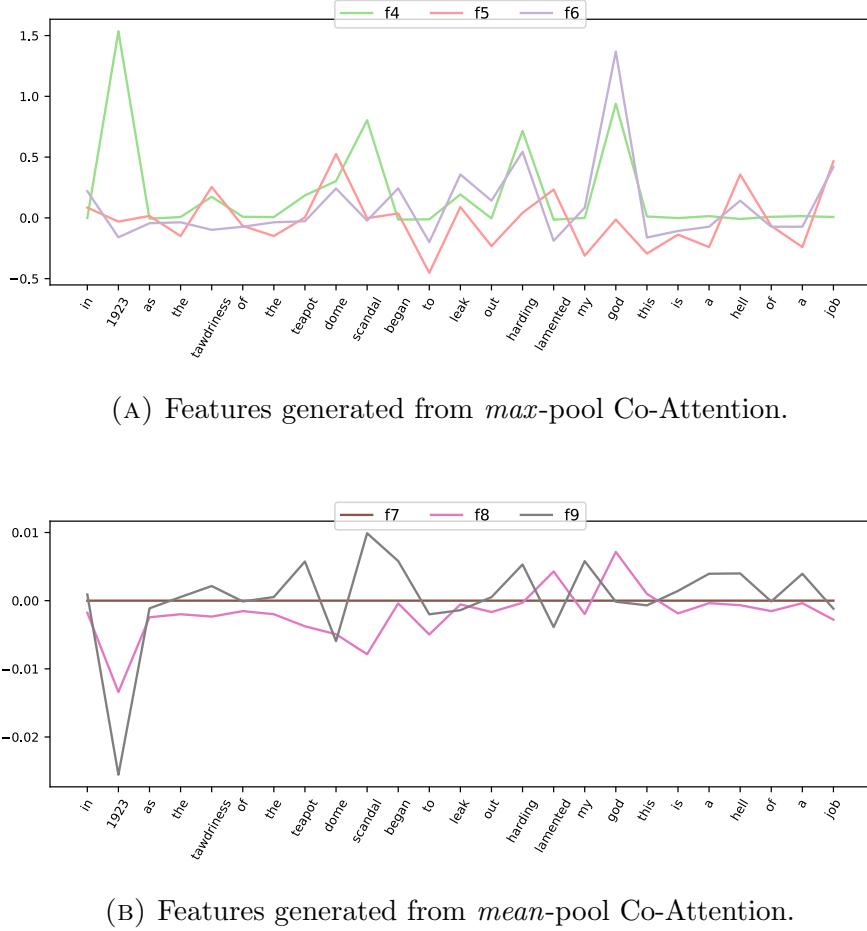


FIGURE 4.4: Differences between Max and Mean-pooled Casted Attention.

4.4 Summary

We proposed a new state-of-the-art neural model for a myriad of retrieval and matching tasks in the domain of question answering and conversation modeling. Our proposed model is based on a re-imagination of the standard and widely applied neural attention. For the first time, we utilize attention not as a pooling operator but as a form of feature augmentation. We propose three methods to compress attentional matrices into scalar features. Via visualization and qualitative analysis, we show that these casted features can be interpreted and understood. Our proposed model achieves highly competitive results on four benchmark tasks and datasets. The achievements of our proposed model are as follows: (1) Our model obtains the highest performing result on the well-studied TrecQA dataset, (2) our model achieves 9% improvement on Ubuntu dialogue corpus relative to

the best existing model, and (3) our model achieves strong results on Community Question Answering and Tweet Reply Prediction.

Chapter 5

Densely Connected Attention Propagation for Machine Reading Comprehension

Machine Reading comprehension (MRC) is one of the core natural language understanding (NLU) tasks. Different from natural language inference, this task evaluates a model’s capability to answer questions, which implicitly tests for understanding and reasoning capabilities. This chapter studies NLU from the perspective of MRC systems. To this end, we propose a new state-of-the-art system for question answering and reading comprehension. This chapter¹ discusses our proposed Densely Connected Attention Propagation (DecaProp) model for machine reading comprehension.

5.1 Introduction

The dominant neural architectures for machine reading comprehension (MRC) typically follow a standard ‘encode-interact-point’ design [34, 110, 111, 113, 117]. Following the embedding layer, a compositional encoder typically encodes Q (query) and P (passage) individually. Subsequently, an (bidirectional) attention layer is

¹This chapter is published as *Densely Connected Attention Propagation for Reading Comprehension*, Proceedings of NeurIPS 2018 [62]

then used to model interactions between P/Q . Finally, these attended representations are then reasoned over to find (point to) the best answer span. While there might be slight variants of this architecture, this overall architectural design remains consistent across many MRC models.

Intuitively, the design of MRC models often possesses some depth, i.e., every stage of the network easily comprises several layers. For example, the R-NET [113] architecture adopts three BiRNN layers as the encoder and two additional BiRNN layers at the interaction layer. BiDAF [110] uses two BiLSTM layers at the pointer layer, etc. As such, MRC models are often relatively deep, at the very least within the context of NLP.

Unfortunately, the depth of a model is not without implications. It is well-established fact that increasing the depth may impair gradient flow and feature propagation, making networks harder to train [146, 170, 171]. This problem is prevalent in computer vision, where mitigation strategies that rely on shortcut connections such as Residual networks [170], GoogLeNet [172] and DenseNets [171] were incepted. Naturally, many of the existing MRC models already have some built-in designs to workaround this issue by shortening the signal path in the network. Examples include attention flow [110], residual connections [112, 118] or simply the usage of highway encoders [146]. As such, we hypothesize that explicitly improving information flow can lead to further and considerable improvements in MRC models.

Another observation is that the flow of P/Q representations across the network are often well-aligned and ‘synchronous’, i.e., P is often only matched with Q at the same hierarchical stage (e.g., only after they have passed through a fixed number of encoder layers). To this end, we hypothesize that increasing the number of interaction interfaces, i.e., matching in an asynchronous, cross-hierarchical fashion, can also lead to an improvement in performance.

Based on the above mentioned intuitions, this chapter proposes a new architecture with two distinct characteristics. Firstly, the proposed network is densely connected, connecting every layer of P with every layer of Q . This not only facilitates information flow but also increases the number of interaction interfaces between P/Q . Secondly, our network is densely connected by *attention*, making it vastly different from any residual mitigation strategy in the literature. To the best of our

knowledge, this is the first work that explicitly considers attention as a form of skip-connector.

Notably, models such as BiDAF incorporates a form of attention propagation (flow). However, this is inherently unsuitable for forging dense connections throughout the network since this would incur a massive increase in the representation size in subsequent layers. To this end, we propose efficient *Bidirectional Attention Connectors* (BAC) as a base building block to connect two sequences at arbitrary layers. The key idea is to compress the attention outputs so that they can be small enough to propagate, yet enabling a connection between two sequences. The propagated features are collectively passed into prediction layers, which effectively connect shallow layers to deeper layers. Therefore, this enables multiple bidirectional attention calls to be executed without much concern, allowing us to efficiently connect multiple layers together.

Our work is concerned with densely connected networks aimed at improving information flow [146, 171, 172]. While most works are concerned with computer vision tasks or general machine learning, there are several notable works in the NLP domain. Ding et al. [173] proposed Densely Connected BiLSTMs for standard text classification tasks. In the MRC domain, DCN+ [112] used Residual Co-Attention encoders. QANet [118] used residual self-attentive convolution encoders. While the usage of highway/residual networks is not an uncommon sight in NLP, the usage of bidirectional attention as a skip-connector is new. Moreover, our work introduces new cross-hierarchical connections, which help to increase the number of interaction interfaces between P/Q .

Overall, we propose DecaProp (Densely Connected Attention Propagation), a novel architecture for machine reading comprehension. DecaProp achieves a significant gain of 2.6% – 14.2% absolute improvement in F1 score over the existing state-of-the-art on four challenging MRC datasets, namely NewsQA [46], Quasar-T [63], SearchQA [48] and NarrativeQA [64].

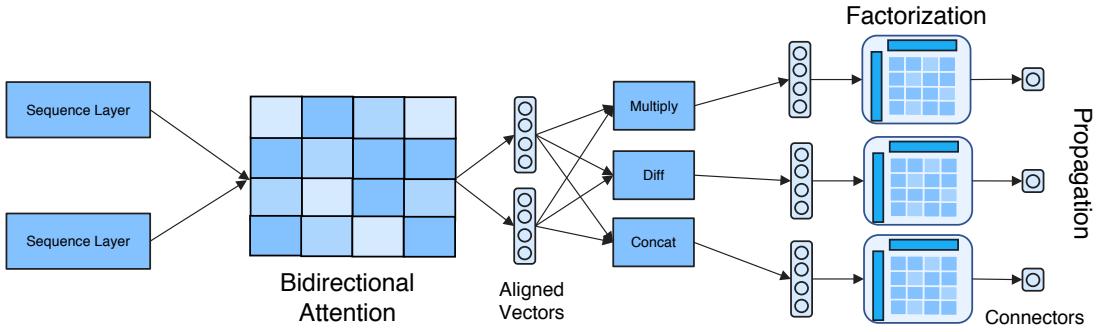


FIGURE 5.1: Architecture of BAC.

5.2 Proposed Method

5.2.1 Bidirectional Attention Connectors (BAC)

This section introduces the Bidirectional Attention Connectors (BAC) module which is central to our overall architecture. The BAC module can be thought of as a *connector* component that connects two sequences/layers. Figure 5.1 shows the BAC module.

The key goals of this module are to (1) connect any two layers of P/Q in the network, returning a residual feature that can be propagated to deeper layers, (2) model cross-hierarchical interactions between P/Q and (3) minimize any costs incurred to other network components such that this component may be executed multiple times across all layers. Notably, signals still have to back-propagate through the BAC parameters. However, this still enjoys the benefits when connecting far away layers and also by increasing the number of pathways.

Let $P \in \mathbb{R}^{\ell_p \times d}$ and $Q \in \mathbb{R}^{\ell_q \times d}$ be inputs to the BAC module. The initial steps in this module remain identical to standard bi-attention in which an affinity matrix is constructed between P/Q . In our bi-attention module, the affinity matrix is computed via:

$$E_{ij} = \frac{1}{\sqrt{d}} \mathbf{F}(p_i)^\top \mathbf{F}(q_j) \quad (5.1)$$

where $\mathbf{F}(\cdot)$ is a standard dense layer with ReLU activations and d is the dimensionality of the vectors. Note that this is the scaled dot-product attention from

[24]. Next, we learn an alignment between P/Q as follows:

$$A = \text{Softmax}(E^\top)P \quad \text{and} \quad B = \text{Softmax}(E)Q \quad (5.2)$$

where A, B are the aligned representations of the query/passage respectively. In many standard neural QA models, it is common to pass an augmented matching vector of this attentional representation to subsequent layers. This often refers to common element-wise operations such as the subtraction or multiplication. For this purpose, functions such as $f = W([b_i; p_i; b_i \odot p_i, b_i - p_i]) + b$ have been used [34]. However, simple/naive augmentation would not suffice in our use case. Even without augmentation, every call of bi-attention returns a new d -dimensional vector for each element in the sequence. If the network has l layers, then connecting all pairwise layers would require l^2 connectors and therefore an output dimension of $l^2 \times d$. This is not only computationally undesirable but also require a large network at the end to reduce this vector. With augmentation, this problem is aggravated. Hence, standard bidirectional attention is not suitable here.

To overcome this limitation, we utilize a parameterized function $G(\cdot)$ to compress the bi-attention vectors down to scalar.

$$g_i^p = [G([b_i; p_i]); G(b_i - p_i); G(b_i \odot p_i)] \quad (5.3)$$

where $g_i^p \in \mathbb{R}^3$ is the output (for each element in P) of the BAC module. This is done in an identical fashion for a_i and q_i to form g_i^q for each element in Q . Intuitively, g_i^* where $* = \{p, q\}$ are the learned scalar attention that is propagated to upper layers. Since there are only three scalars, they will not cause any problems even when executed for multiple times. As such, the connection remains relatively lightweight. This compression layer can be considered as a defining trait of the BAC, differentiating it from standard bi-attention.

Naturally, there are many potential candidates for the function $G(\cdot)$. One natural choice is the standard dense layer (or multiple dense layers). However, dense layers are limited as they do not compute dyadic pairwise interactions between features that inhibit its expressiveness. On the other hand, factorization-based models are known to not only be expressive and efficient, but also able to model low-rank structure well.

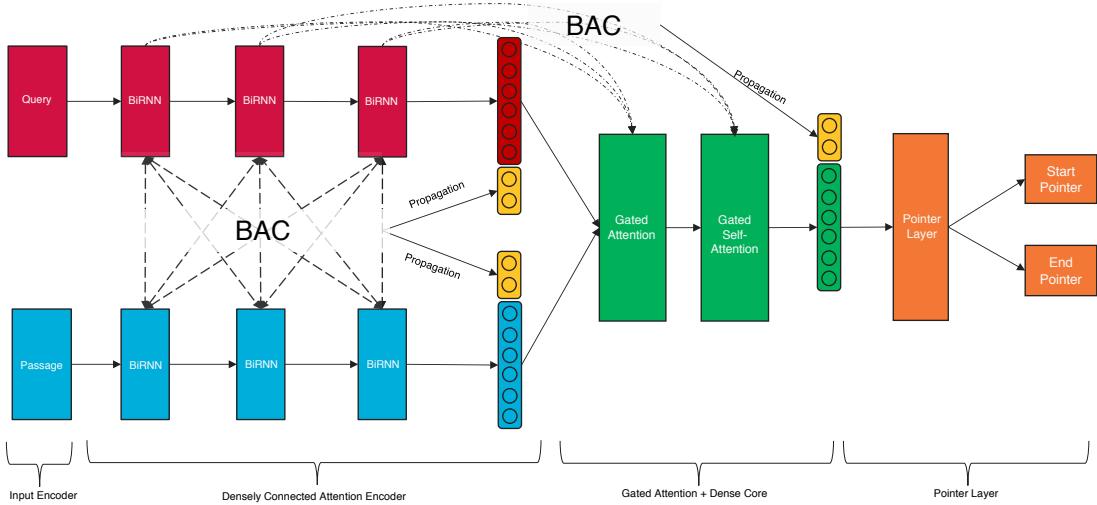


FIGURE 5.2: Architecture of DecaProp.

To this end, we adopt factorization machines (FM) [144] as $G(\cdot)$. The FM layer is defined as:

$$G(x) = w_0 + \sum_{i=1}^n w_i x_i + \sum_{i=1}^n \sum_{j=i+1}^n \langle v_i, v_j \rangle x_i x_j \quad (5.4)$$

where $v \in \mathbb{R}^{d \times k}$, $w_0 \in \mathbb{R}$ and $w_i \in \mathbb{R}^d$. The output $G(x)$ is a scalar. Intuitively, this layer tries to learn pairwise interactions between every x_i and x_j using factorized (vector) parameters v . In the context of our BAC module, the FM layer is trying to learn a low-rank structure from the ‘match’ vector (e.g., $b_i - p_i$, $b_i \odot p_i$ or $[b_i; p_i]$). Finally, we note that the BAC module takes inspiration from the main body of our CAFE model [174] for entailment classification and natural language inference. However, this work demonstrates the usage and potential of the BAC as a residual connector.

5.2.2 Densely Connected Attention Propagation (DecaProp)

In this section, we describe our proposed model in detail. Figure 5.2 depicts a high-level overview of our proposed architecture of DecaProp.

5.2.2.1 Contextualized Input Encoder

The inputs to our model are two sequences P and Q which represent passage and query respectively. Given Q , the task of the MRC model is to select a sequence of tokens in P as the answer. Following many MRC models, we enhance the input representations with (1) character embeddings (passed into a BiRNN encoder), (2) a binary match feature which denotes if a word in the query appears in the passage (and vice versa) and (3) a normalized frequency score denoting how many times a word appears in the passage. The Char BiRNN of h_c dimensions, along with two other binary features, is concatenated with the word embeddings $w_i \in \mathbb{R}^{d_w}$, to form the final representation of $d_w + h_c + 2$ dimensions.

5.2.2.2 Densely Connected Attention Encoder (DecaEnc)

The DecaEnc accepts the inputs P and Q from the input encoder. DecaEnc is a multi-layered encoder with k layers. For each layer, we pass P/Q into a bidirectional RNN layer of h dimensions. Next, we apply our attention connector (BAC) to $H^P/H^Q \in \mathbb{R}^\sim$ where H represents the hidden state outputs from the BiRNN encoder where the RNN cell can either be a GRU or LSTM encoder. Let d be the input dimensions of P and Q , then this encoder goes through a process of $d \rightarrow h \rightarrow h + 3 \rightarrow h$ in which the BiRNN at layer $l + 1$ consumes the propagated features from layer l .

Intuitively, this layer models P/Q whenever they are at the same network hierarchical level. At this point, we include ‘asynchronous’ (cross hierarchy) connections between P and Q . Let P^i, Q^i denote the representations of P, Q at layer i . We apply the Bidirectional Attention Connectors (BAC) as follows:

$$Z_p^{ij}, Z_q^{ij} = F_C(P^i, Q^j) \quad \forall i, j = 1, 2 \dots n \quad (5.5)$$

where F_C represents the BAC component. This densely connects all representations of P and Q across multiple layers. $Z_*^{ij} \in \mathbb{R}^{3 \times \ell}$ represents the generated features for each ij combination of P/Q . In total, we obtain $3n^2$ compressed attention features for each word. Intuitively, these features capture fine-grained relationships between P/Q at different stages of the network flow. The output of the encoder is

the concatenation of all the BiRNN hidden states $H^1, H^2 \dots H^k$ and Z^* which is a matrix of $(nh + 3n^2) \times \ell$ dimensions.

5.2.2.3 Densely Connected Core Architecture (DecaCore)

This section introduces the core architecture of our proposed model. This component corresponds to the interaction segment of standard MRC model architecture.

Gated Attention The outputs of the densely connected encoder are then passed into a standard gated attention layer. This corresponds to the ‘*interact*’ component in many other popular MRC models that model Q/P interactions with attention. While there are typically many choices of implementing this layer, we adopt the standard gated bi-attention layer following [113].

$$S = \frac{1}{\sqrt{d}} \mathbf{F}(P)^\top (\mathbf{F}(Q)) \quad (5.6)$$

$$\bar{P} = \text{Softmax}(S)Q \quad (5.7)$$

$$P' = \text{BiRNN}(\sigma(\mathbf{W}_g([P; \bar{P}]) + \mathbf{b}_g) \odot P) \quad (5.8)$$

where σ is the sigmoid function and $F(\cdot)$ are dense layers with ReLU activations. The output P' is the query-dependent passage representation.

Gated Self-Attention Next, we employ a self-attention layer, applying Equation (5.8) yet again on P' , matching P' against itself to form B , the output representation of the core layer. The key idea is that self-attention compares each word in the query-dependent passage representation against all other words, enabling each word to benefit from a wider global view of the context.

Dense Core At this point, we note that there are two intermediate representations of P , i.e., one after the gated bi-attention layer and one after the gated self-attention layer. We denote them as U^1, U^2 respectively. Unlike the Densely Connected Attention Encoder, we no longer have two representations at each hierarchical level since they have already been ‘fused’. Hence, we apply a one-sided BAC to all permutations of $[U^1, U^2]$ and $Q^i, \forall i = 1, 2 \dots k$. Note that the one-sided

BAC only outputs values for the left sequence, ignoring the right sequence.

$$R^{kj} = F'_C(U^j, Q^k) \quad \forall k = 1, 2 \dots n, \forall j = 1, 2 \quad (5.9)$$

where $R^{kj} \in \mathbb{R}^{3 \times \ell}$ represents the connection output and F'_C is the one-sided BAC function. All values of R^{kj} , $\forall j = 1, 2, \forall k = 1, 2 \dots n$ are concatenated to form a matrix R' of $(2n \times 6) \times \ell$, which is then concatenated with U^2 to form $M \in \mathbb{R}^{\ell_p \times (d+12n)}$. This final representation is then passed to the answer prediction layer.

5.2.2.4 Answer Pointer and Prediction Layer

Next, we pass M through a stacked BiRNN model with two layers and obtain two representations, H_1^\dagger and H_2^\dagger respectively.

$$H_1^\dagger = \text{BiRNN}(M) \text{ and } H_2^\dagger = \text{BiRNN}(H_1^\dagger) \quad (5.10)$$

The start and end pointers are then learned via:

$$p^1 = \text{Softmax}(w_1 H_1^\dagger) \text{ and } p^2 = \text{Softmax}(w_2 H_2^\dagger) \quad (5.11)$$

where $w_1, w_2 \in \mathbb{R}^d$ are parameters of this layer. To train the model, following prior work, we minimize the sum of negative log probabilities of the start and end indices:

$$L(\theta) = -\frac{1}{N} \sum_i^N \log(p_{y_i^1}^1) + \log(p_{y_i^2}^2) \quad (5.12)$$

where N is the number of samples, y_i^1, y_i^2 are the true start and end indices. p_k is the k -th value of the vector p . The test span is chosen by finding the maximum value of p_k^1, p_l^2 where $k \leq l$.

5.3 Experiments

This section describes our experimental setup and empirical results.

5.3.1 Datasets and Competitor Baselines

We conduct experiments on four challenging QA datasets which are described as follows:

NewsQA This challenging MRC dataset [46] comprises $100k$ QA pairs. Passages are relatively long at about 600 words on average. This dataset has also been extensively used in benchmarking MRC models. On this dataset, the key competitors are BiDAF [110], Match-LSTM [34], FastQA/FastQA-Ext [175], R2-BiLSTM [176], AMANDA [117].

Quasar-T This dataset [63] comprises $43k$ factoid-based QA pairs and is constructed using ClueWeb09 as its backbone corpus. The key competitors on this dataset are BiDAF and the Reinforced Ranker-Reader (R^3) [88]. Several variations of the ranker-reader model (e.g., SR, SR^2), which use the Match-LSTM underneath, are also compared against.

SearchQA This dataset [48] aims to emulate the search and retrieval process in question answering applications. The challenge involves reasoning over multiple documents. In this dataset, we concatenate all documents into a single passage context and perform MRC over the documents. The competitor baselines on this dataset are Attention Sum Reader (ASR) [177], Focused Hierarchical RNNs (FH-RNN) [178], AMANDA [117], BiDAF, AQA [179] and the Reinforced Ranker-Reader (R^3) [88].

NarrativeQA This dataset [64] is a recent QA dataset that involves comprehension over stories. We use the summaries setting which is closer to a standard QA or reading comprehension setting. We compare with the baselines in the original paper, namely Seq2Seq, Attention Sum Reader and BiDAF. We also compare with the recent BiAttention + MRU model [4].

As compared to the popular SQuAD dataset [1], these datasets are either (1) more challenging, involving more multi-sentence reasoning or (2) is concerned with searching across multiple documents in an ‘open domain’ setting (SearchQA/Quasar-T). Hence, these datasets accurately reflect real-world applications to a greater

extent. However, we regard the concatenated documents as a single context for performing reading comprehension. The evaluation metrics are the EM (exact match) and F1 score. Note that for all datasets, we compare all models solely on the MRC task. Therefore, for fair comparison we do not compare with algorithms that use a second-pass answer re-ranker [180]. Finally, to ensure that our model is not a failing case of SQuAD, we also include development set scores of our model on SQuAD.

5.3.2 Experimental Setup

Our model is implemented in Tensorflow [152]. The sequence lengths are capped at 800/700/1500/1100 for NewsQA, SearchQA, Quasar-T and NarrativeQA respectively. We use Adadelta [181] with $\alpha = 0.5$ for NewsQA, and Adam optimizer [153] with $\alpha = 0.001$ for SearchQA, Quasar-T and NarrativeQA. The choice of the RNN encoder is tuned between GRU and LSTM cells, and the hidden size is tuned amongst {32, 50, 64, 75}. We use the cuDNN implementation of the RNN encoder. Batch size is tuned amongst {16, 32, 64}. Dropout rate is tuned amongst {0.1, 0.2, 0.3} and applied to all RNN and fully-connected layers. We apply variational dropout [182] in-between RNN layers. We initialize the word embeddings with 300D GloVe embeddings [16] which are fixed during training. The size of the character embeddings is set to 8 and the character-level RNN is set to the same as the word-level RNN encoders. The maximum characters per word is set to 16. The number of layers in DecaEnc is set to 3 and the number of factors in the factorization kernel is set to 64. We use a learning rate decay factor of 2 and patience of 3 epochs whenever the EM (or ROUGE-L) score on the development set does not increase.

5.3.3 Performance Results

Overall, our results are optimistic and promising, with results indicating that DecaProp achieves state-of-the-art performance on all four datasets.

NewsQA Table 5.1 reports the results on NewsQA. On this dataset, DecaProp outperforms the existing state-of-the-art, i.e., the recent AMANDA model by

TABLE 5.1: Results on NewsQA dataset.

Model	Dev		Test	
	EM	F1	EM	F1
Match-LSTM	34.4	49.6	34.9	50.0
BARB	36.1	49.6	34.1	48.2
BiDAF	N/A	N/A	37.1	52.3
Neural BoW	25.8	37.6	24.1	36.6
FastQA	43.7	56.4	41.9	55.7
FastQAExt	43.7	56.1	42.8	56.1
R2-BiLSTM	N/A	N/A	43.7	56.7
AMANDA	48.4	63.3	48.4	63.7
DecaProp	52.5	65.7	53.1	66.3

TABLE 5.2: Results on Quasar-T dataset.

	Dev		Test	
	EM	F1	EM	F1
GA	25.6	25.6	26.4	26.4
BiDAF	25.7	28.9	25.9	28.5
SR	N/A	N/A	31.5	38.5
SR ²	N/A	N/A	31.9	38.8
R ³	N/A	N/A	34.2	40.9
DecaProp	39.7	48.1	38.6	46.9

(+4.7% EM / +2.6% F1). Notably, AMANDA is a strong neural baseline that also incorporates gated self-attention layers, along with question-aware pointer layers. Moreover, our proposed model also outperforms well-established baselines such as Match-LSTM (+18% EM / +16.3% F1) and BiDAF (+16% EM / +14% F1).

Quasar-T Table 5.2 reports the results on Quasar-T. Our model achieves state-of-the-art performance on this dataset, outperforming the state-of-the-art R³ (Reinforced Ranker Reader) by a considerable margin of +4.4% EM / +6% F1. Performance gain over standard baselines such as BiDAF and GA are even larger (> 15% F1).

SearchQA Table 5.3 and Table 5.4 report the results on SearchQA. On the original setting, our model outperforms AMANDA by +15.4% EM and +14.2% in terms of F1 score. On the overall setting, our model outperforms both AQA (+18.1% EM / +18% F1) and Reinforced Reader Ranker (+7.8% EM / +8.3%

TABLE 5.3: Results on SearchQA (original) dataset.

	Dev		Test	
	Acc	F1 ⁿ	Acc	F1 ⁿ
TF-IDF max	13.0	N/A	12.7	N/A
ASR	43.9	24.2	41.3	22.8
FH-RNN	49.6	56.7	46.8	53.4
AMANDA	48.6	57.7	46.8	56.6
DecaProp	64.5	71.9	62.2	70.8

TABLE 5.4: Results on SearchQA dataset.

	Dev		Test	
	EM	F1	EM	F1
BiDAF	31.7	37.9	28.6	34.6
AQA	40.5	47.4	38.7	45.6
R ³	N/A	N/A	49.0	55.3
DecaProp	58.8	65.5	56.8	63.6

TABLE 5.5: Results on NarrativeQA (Story Summaries) dataset.

	Test / Validation			
	BLEU-1	BLEU-4	METEOR	ROUGE-L
Seq2Seq	15.89 / 16.10	1.26 / 1.40	4.08 / 4.22	13.15 / 13.29
ASR	23.20 / 23.54	6.39 / 5.90	7.77 / 8.02	22.26 / 23.28
BiDAF	33.72 / 33.45	15.53 / 15.69	15.38 / 15.68	36.30 / 36.74
MRU	- / 36.55	- / 19.79	- / 17.87	- / 41.44
DecaProp	42.00 / 44.35	23.42 / 27.61	23.42 / 21.80	40.07 / 44.69

F1). Both models are reinforcement learning based extensions of existing strong baselines such as BiDAF and Match-LSTM.

NarrativeQA Table 5.5 reports the results on NarrativeQA. Our proposed model outperforms all baseline systems (Seq2Seq, ASR, BiDAF) in the original paper. On average, there is a $\approx +5\%$ improvement across all metrics.

SQuAD Table 5.6 reports dev scores of our model against several representative models on the popular SQuAD benchmark. While our model does not achieve state-of-the-art performance, our model can outperform the base R-NET (both our implementation as well as the published score). Our model achieves reasonably competitive performance.

TABLE 5.6: Results on SQuAD (Dev Set) dataset.

Model	EM	F1
DCN [111]	66.2	75.9
DCN + CoVE [85]	71.3	79.9
R-NET (Wang et al.) [113]	72.3	80.6
R-NET (Our re-implementation)	71.9	79.6
DecaProp (This work)	72.9	81.4
QANet [118]	73.6	82.7

5.3.4 Ablation Study

We conduct an ablation study on the NewsQA development set (Table 5.7). More specifically, we report the development scores of seven ablation baselines. In (1), we removed the entire DecaProp architecture, reverting it to an enhanced version of the original R-NET model. For a fairer comparison, we make several enhancements to the R-NET model as follows: (1) We replaced the additive attention with scaled dot-product attention similar to ours. (2) We added shortcut connections after the encoder layer. (3) We replaced the original Pointer networks with our BiRNN Pointer Layer. We found that these enhancements consistently lead to improved performance. The original R-NET performs at $\approx 2\%$ lower on NewsQA. In (2), we removed DecaCore and passed U^2 to the answer layer instead of M . In (3), we removed the DecaEnc layer and used a 3-layered BiRNN instead. In (4), we kept the DecaEnc but only compared layers of the same hierarchy and omitted cross hierarchical comparisons. In (5), we removed the Gated Bi-Attention and Gated Self-Attention layers. Removing these layers simply allow previous layers to pass through. In (6-7), we varied n , the number of layers of DecaEnc. Finally, in (8-9), we varied the FM with linear and nonlinear feed-forward layers.

From (1), we observe a significant gap in performance between DecaProp and R-NET. This demonstrates the effectiveness of our proposed architecture. Overall, the key insight is that all model components are crucial to DecaProp. Notably, the DecaEnc seems to contribute the most to the overall performance. Finally, Figure 5.3 shows the performance plot of the development EM metric (NewsQA) over training. We observe that the superiority of DecaProp over R-NET is consistent and relatively stable.

TABLE 5.7: Ablation study on NewsQA development set.

Ablation	EM	F1
(1) Remove All (R-NET)	48.1	61.2
(2) w/o DecaCore	51.5	64.5
(3) w/o DecaEnc	49.3	62.0
(4) w/o Cross Hierarchy	50.0	63.1
(5) w/o Gated Attention	49.4	62.8
(6) Set DecaEnc $n = 2$	50.5	63.4
(7) Set DecaEnc $n = 4$	50.7	63.3
(8) DecaProp (Linear)	50.9	63.0
(9) DecaProp (Nonlinear)	48.9	60.0
Full Architecture ($n = 3$)	52.5	65.7

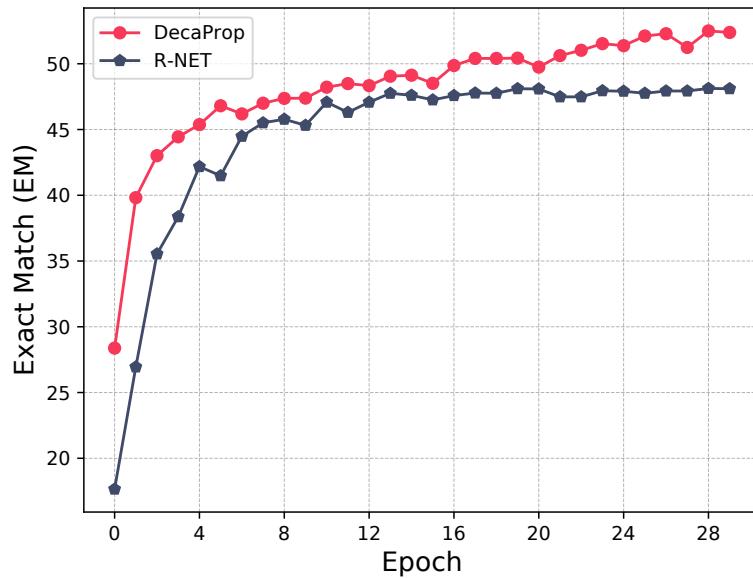


FIGURE 5.3: DecaProp versus R-NET on NewsQA.

5.4 Summary

We proposed a new *Densely Connected Attention Propagation* (DecaProp) mechanism. For the first time, we explore the possibilities of using bidirectional attention as a skip-connector. We proposed Bidirectional Attention Connectors (BAC) for efficient connection of any two arbitrary layers, producing connectors that can be propagated to deeper layers. This enables a shortened signal path, aiding information flow across the network. Additionally, the modularity of the BAC allows it to be easily equipped to other models and even other domains. Our proposed architecture achieves state-of-the-art performance on four challenging QA datasets, outperforming strong and competitive baselines such as Reinforced Reader Ranker

(R^3), AMANDA, BiDAF and R-NET. The source code for our model and experiments can be found at https://github.com/vanzytay/NIPS2018_DECAPROP.

Chapter 6

Introspective Curriculum Pointer-Generator for Machine Reading Comprehension over Long Narratives

Teaching machines to read and comprehend is a fundamentally interesting and challenging problem in AI research [1, 5, 46]. While there have been considerable and broad improvements in reading and understanding textual snippets, the ability for machines to read/understand complete stories and novels is still in infancy [49]. The challenge becomes insurmountable in lieu of not only the large context but also the intrinsic challenges of narrative text which arguably requires a larger extent of reasoning. As such, this motivates the inception of relevant, interesting benchmarks such as the NarrativeQA Reading Comprehension challenge. In this chapter¹, we tackle the full story setting instead of the summary setting which, inherently, is a much harder task [49].

¹This chapter is published as *Simple and Effective Curriculum Pointer-Generator Networks for Reading Comprehension over Long Narratives*, Proceedings of ACL 2019 [65].

6.1 Introduction

The challenges of having a long context have been traditionally mitigated by a two-step approach - retrieval first and then reading second [89, 108, 114]. This difficulty mirrors the same challenges of open domain question answering, albeit introducing additional difficulties due to the nature of narrative text (stories and retrieved excerpts need to be coherent). While some recent works have proposed going around by training retrieval and reading components end-to-end, this work follows the traditional paradigm with a slight twist. We train our models to be robust regardless of whatever is retrieved. This is in a similar spirit to domain randomization [183].

In order to do so, we propose a diverse curriculum learning scheme [184] based on two concepts of difficulty. The first, depends on whether the answer exists in the context (*answerability*), aims to bridge the gap between training time and inference time retrieval. On the other hand, and the second, depends on the size of the retrieved documents (coherence and *understandability*). While conceptually simple, we found that these heuristics help improve the performance of the QA model. To the best of our knowledge, we are the first to incorporate these notions of difficulty in QA reading models.

All in all, our model tries to learn to generate the answer even if the correct answer does not appear as evidence which acts as a form of *generative pretraining during training*. As such, this is akin to learning to guess, largely motivated by how humans are able to extrapolate/guess even when given access to a small fragment of a film/story. In this case, we train our model to generate answers, making do with whatever context it was given. To this end, a curriculum learning scheme controls the extent of difficulty of the context given to the model.

At this juncture, it would be easy to realize that standard pointer-based reading comprehension models would not adapt well to this scheme, as they fundamentally require the golden label to exist within the context [34, 110]. As such, our overall framework adopts a pointer-generator framework [185] that learns to point and generate, conditioned on not only the context but also the question. This relaxes this condition, enabling us to train our models with diverse views of the same story which is inspired by domain randomization [183]. For our particular task at hand,

the key idea is that, even if the answer is not found in the context, we learn to generate the answer despite the noisy context.

Finally, our method also incorporates a novel Introspective Alignment Reader (IAL Reader). The key idea of the IAL mechanism is to introspect over decomposed alignments using block-style local self-attention. This not only imbues our model with additional reasoning capabilities but enables a finer-grained (and local-globally aware) comparison between soft-aligned representations. All in all, our IAL mechanism can be interpreted as learning a matching over matches.

All in all, the prime contributions of this work are summarized as follows:

- We propose a curriculum learning based Pointer-Generator model for reading comprehension over narratives (long stories). For the first time, we propose two different notions of difficulty for constructing diverse views of long stories for training. We show that this approach achieves better results than existing models adapted for open-domain question answering.
- Our proposed model incorporates an Introspective Alignment Reader (IAL Reader) which uses block-based self-attentive reasoning over decomposed alignments. Ablative experiments show improvements of our IAL layer over the standard usage of vanilla self-attention.
- Our proposed framework (IAL-CPG) achieves state-of-the-art performance on the NarrativeQA reading comprehension challenge. On metrics such as BLEU-4 and Rouge-L, we achieve a 17% relative improvement over prior state-of-the-art and a **10** times improvement in terms of BLEU-4 score over BiDAF, a strong span prediction based model.

6.2 Proposed Method

This section outlines the components of our proposed architecture (IAL-CPG). Since our problem is mainly dealing with extremely long sequences, we employ an initial retrieval phrase by either using the answer or question as a cue (query for retrieving relevant chunks/excerpts). The initial retrieval is unavoidable since supporting up to 20K-30K words in computational graphs is still not manageable

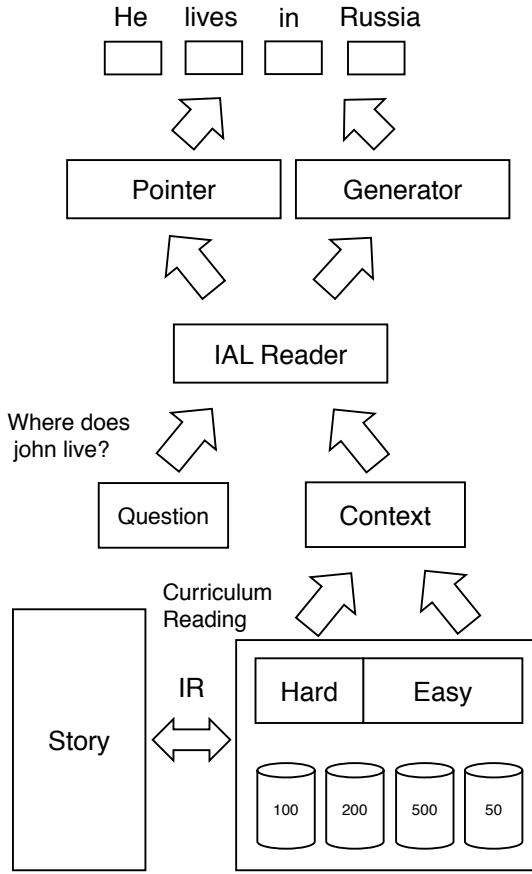


FIGURE 6.1: Architecture of IAL-CPG.

even with top-grade GPUs. The retrieval stage is controlled by our curriculum learning process in which the details are deferred to subsequent sections. The overall illustration of this framework is depicted in Figure 6.1.

6.2.1 Introspective Alignment Reader

This section introduces our proposed Introspective Alignment Reader (IAL Reader).

Input and Context Encoding Our model accepts two inputs, context C and question Q . Each input is a sequence of words. We pass each sequence into a shared Bidirectional LSTM layer.

$$H^c = \text{BiLSTM}(C) \text{ and } H^q = \text{BiLSTM}(Q)$$

where $H^c \in \mathbb{R}^{\ell_c \times d}$ and $H^q \in \mathbb{R}^{\ell_q \times d}$ are the hidden representations for C and Q respectively.

Introspective Alignment Next, we pass H^c, H^q into an alignment layer. Firstly, we compute a soft attention affinity matrix between H^c and H^q as follows:

$$E_{ij} = F(h_i^c)^\top F(h_j^q) \quad (6.1)$$

where h_i^c is the i -th word in the context and h_j^q is the j -th word in the question. $F(\cdot)$ is a standard nonlinear transformation function (i.e., $F(x) = \sigma(Wx + b)$, where σ indicates non-linearity function), and is shared between context and question. $E \in \mathbb{R}^{\ell_c \times \ell_q}$ is the soft matching matrix. To learn alignments between context and question, we compute:

$$A = \text{Softmax}(E) H^q$$

where $A \in \mathbb{R}^{\ell_c \times d}$ is the aligned representation of H^c .

Reasoning over Alignments Next, to reason over alignments, we compute a self-attentive reasoning over decomposed alignments:

$$\begin{aligned} G_{ij} &= F_s([A_i; H_i^c; A_i - H_i^c, A_i \odot H_i^c])^\top \cdot \\ &\quad F_s([A_j; H_j^c; A_j - H_j^c, A_j \odot H_j^c]) \end{aligned} \quad (6.2)$$

where square brackets $[\cdot; \cdot]$ denote vector concatenation, $F_s(\cdot)$ is another nonlinear transformation layer which projects onto $4d$ dimensions. i is the positional index of each word token. Intuitively, A_i comprises softly aligned question representations with respect to the context. The usage of the *Hadamard* and *Subtraction* operators helps to enhance the degree of comparison/matching. Hence, by including an additional local reasoning over these enhanced alignment vectors, our model can be interpreted as introspecting over alignment matches.

Local Block-based Self-Attention Since ℓ_c is large in our case (easily ≥ 2000), computing Equation (6.2) may become computationally prohibitive. As such, we compute the scoring function for all cases where $|i - j| \leq b$, in which, b is a

predefined hyperparameter and also the block size. Intuitively, the initial alignment layer (i.e., Equation (6.1)) already considers a global view. As such, this self-attention layer can be considered as a local-view perspective, confining the affinity matrix computation to a local window of b . Finally, to compute the introspective alignment representation, we compute:

$$B = \text{Softmax}(G) [A; H^c; A - H^c; A \odot H^c]$$

where $B^{\ell_c \times 4d}$ is the introspective aligned representation of A . Finally, we use another d -dimensional BiLSTM layer to aggregate the aligned representations:

$$Y = \text{BiLSTM}([B; A; H^c; A - H^c; A \odot H^c]) \quad (6.3)$$

where $Y \in \mathbb{R}^{\ell_c \times 2d}$ is the final contextual representation of context C .

6.2.2 Pointer-Generator Decoder

Motivated by recent, seminal work in neural summarization, our model adopts a pointer-generator architecture [185]. Given Y (the question infused contextual representation), we learn to either generate a word from vocabulary, or point to a word from the context. The decision to generate or point is controlled by an additive blend of several components such as the previous decoder state and/or question representation.

The pointer-generator decoder in our framework uses a LSTM decoder² with a cell state $c_t \in \mathbb{R}^n$ and hidden state vector $h_t \in \mathbb{R}^n$. At each decoding time step t , we compute an attention over Y as follows:

$$g_i = \tanh(F_a(y_i) + F_h(h_{t-1}) + F_q(H^q)), \quad (6.4)$$

$$a_i = g_i^\top w_a, \quad y_t = \sum_{i=0}^{\ell_c} a_i \cdot y_i \quad (6.5)$$

where $F_a(\cdot)$ and $F_h(\cdot)$ are nonlinear transformations projecting to n dimensions. i is the positional index of the input sequence. $F_q(\cdot)$ is an additional attentive pooling operator over the question representation H_q (after the context encoding

²To initialize the LSTM, we use an additional projection layer over the mean pooled representation of Y similar to [186].

layer). The semantics of the question may be lost after the alignment based encoding. As such, this enables us to revisit the question representation to control the decoder. $y_t \in \mathbb{R}^n$ is the context representation at decoding time step t and $a \in \mathbb{R}^{\ell_c}$ is an attention distribution over the context words which is analogous to the final probability distributions that exist in typical span prediction models. Next, we compute the next hidden state via:

$$h_t, c_t = \text{LSTM}([y_t; w_{t-1}], h_{t-1}, c_{t-1})$$

where w_{t-1} is the $(t-1)$ th token in the ground truth answer (teacher forcing). To learn to generate, we compute:

$$v_t = W_v(h_t) + b_v \quad (6.6)$$

where $v_t \in \mathbb{R}^{|V_g|}$, V_g is the global vocabulary size. The goal of the pointer-generator decoder is to choose between the abstractive distribution v_t over the vocabulary (see Equation (6.6)) and the extractive distribution a_t (see Equation (6.5)) over the context text tokens. To this end, we learn a scalar switch $p_t \in \mathbb{R}$:

$$p_t = \text{sigmoid}(F_{pc}(c_t) + F_{ph}(h_t) + F_{py}(y_t))$$

where $F_{pc}(\cdot)$, $F_{ph}(\cdot)$, $F_{py}(\cdot)$ are linear transformation layers (without bias) which project c_t , h_t and y_t into scalar values. To control the blend between the attention context and the generated words, we use a linear interpolation between a_t and v_t . The predicted word w_t at time step t is therefore:

$$w_t = \text{argmax}(p_t \cdot a_t + (1 - p_t)v_t)$$

Note that we scale (append and prepend) a_t and v_t with zeros to make them the same length (i.e., $\ell_c + |V_g|$). The LSTM decoder runs for a predefined fix answer length. During inference, we simply use greedy decoding to generate the output answer.

6.2.3 Curriculum Reading

A key advantage of the pointer-generator is that it allows us to generate answers even if the answers do not exist in the context. This also enables us to explore multiple (diverse) views of contexts to train our model. However, to this end, we must be able to identify effectively the most useful retrieved context evidences for the training. For that purpose, we propose to use a diverse curriculum learning scheme which is based on two intuitive notions of difficulty: answerability and understandability.

Answerability It is regarded as common practice to retrieve excerpts based by using the correct answer as a cue (during training). This establishes an additional gap between training and inference since during inference, correct answers are not available. This measure aims to bridge the gap between question and answer (as a query prompt for passage retrieval). In this case, we consider the set of documents retrieved based on questions as the *hard* setting, H . Conversely, the set of retrieved documents using answers is regarded as the *easy* setting, E .

Understandability This aspect controls how understandable the overall retrieved documents are as a whole. The key idea of this setting is to control the paragraph/chunk size. Intuitively, a small paragraph/chunk size would enable more relevant components to be retrieved from the document. However, its understandability might be affected if paragraph/chunk size is too small. Conversely, a larger chunk size would be easier to be understood. To control the level of understandability, we pre-define several options of chunk sizes (e.g., $\{50, 100, 200, 500\}$) which will be swapped and determined during training.

To combine the two measures described above, we comprise an easy-hard set pair for each chunk size, i.e., $\{E_k, H_k\}$, where:

$$\begin{aligned} k &\in \{50, 100, 200, 500\}, \\ E_n &\leftarrow F(\text{corpus, answer, } n), \\ H_n &\leftarrow F(\text{corpus, question, } n) \end{aligned} \tag{6.7}$$

$F(\cdot)$ is an arbitrary ranking function which may or may not be parameterized, and n is the size of each retrieved chunk.

Algorithm 6.1 Two-layer Curriculum Reading Algorithm

```

1: chunk_list  $\leftarrow \{50, 100, 200, 500\}$ 
2:  $n \leftarrow \text{sample } i \text{ in } \text{chunk\_list}$ 
3: chunk_list  $\leftarrow \text{chunk\_list} \setminus \{n\}$ 
4:  $E_n \leftarrow F(\text{Corpus}, \text{Answers}, n)$ 
5:  $H_n \leftarrow F(\text{Corpus}, \text{Questions}, n)$ 
6:  $D \leftarrow E_n$                                  $\triangleright$  initial training set
7:  $count \leftarrow 0$                            $\triangleright$  number of swaps within a chunk size
8: for  $i \leftarrow 1$  to numEpochs do
9:   Train( $D$ )
10:   $score \leftarrow \text{Evaluate}(\text{Dev\_set})$ 
11:  if  $score < bestDev$  then
12:    if  $count \leq 1/\delta$  then
13:       $D \leftarrow \text{Swap}(D, E_n, H_n, \delta)$   $\triangleright$  Swap  $\delta$  percent of easy set with the hard set
14:       $count \leftarrow count + 1$ 
15:    else
16:      Repeat step 3 to 8  $\triangleright$  Replace train set with new easy set of diff chunk size
17:    else
18:       $bestDev = score$ 

```

Two-layer Curriculum Reading Algorithm As our model utilizes two above measures of difficulty, there lies a question on whether we should swap one measure at a time or swap both whenever the model meets the failure criterion. In our case, we find that prioritizing answerability over understandability is a better choice. More concretely, at the beginning of the training, we start with an easy set E_k of a random chunk size k . When the failure criterion is met (e.g. the model score does not improve on the validation set), we randomly swap a small percent δ (e.g., 5% in our experiments) of the easy set E_k with the hard set H_k within its own chunk size group k to improve the *answerability*. In this case, after $\frac{1}{\delta}$ failures, the model runs out of easy set E_k and is completely based on the hard set H_k . At this junction, we swap the model for *understandability*, replacing the training set with a completely new easy set E_l of another chunk size l , and repeat the above process. The formal description of our proposed curriculum reading is introduced in Algorithm 6.1.

6.3 Experiments

We conduct our experiments on the NarrativeQA reading comprehension challenge. The NarrativeQA dataset was introduced earlier in Chapter 5. However, we use the full story setting in this chapter.

6.3.1 Experimental Setup

This section introduces our experimental setup.

Model Hyperparameters We implement our model in Tensorflow. Our model is trained with Adadelta [181]. The initial learning rate is tuned amongst $\{0.1, 0.2, 0.5\}$. The L2 regularization is tuned amongst $\{10^{-8}, 10^{-6}, 10^{-5}\}$. The size of the LSTM at the encoder layer is set to 128 and the decoder size is set to 256. The block size b for the Introspective Alignment Reader Layer is set to 200. We initialize our word embeddings with pre-trained GloVe vectors [187] which are not updated during training. In our early experiments, we also masked entities following the original work [49], however, we did not observe obvious difference in performance. This is probably because we do not update word embeddings during training.

Implementation Details Text is lowercased and tokenized with NLTK³. For the retrieval of paragraphs, we use the cosine similarity between TF-IDF vector representations. TF-IDF representations are vectorized by Scikit-Learn using an n -gram range of $[1, 3]$ with stopword filtering. The maximum context size is tuned amongst $\{2000, 4000\}$ and reported accordingly. The paragraph/chunk size is dynamic and configured amongst $\{50, 100, 200, 500\}$. The retrieved excerpts are retrieved based on similarity match between context chunks and answer or question depending on the curriculum learning scheme. We tune the maximum answer length amongst $\{6, 8, 12\}$ and the maximum question length is set to 30. Since two answers are provided for each question, we train on both sets of answers. During the construction of the golden labels, we first perform an n -gram search of the answer in the context. The largest n -gram match is allocated indices belonging to the context (i.e., $[1, \ell_c]$). For the remainder words, stopwords are automatically allocated indices in the global vocabulary and non-stopwords are assigned with context indices. If an answer word is not found, it is ignored. To construct the global vocabulary for the pointer generator decoder and avoid story-specific words, we use words that appear in at least 10 stories.

³<https://www.nltk.org/>

Evaluation During evaluation, we (1) remove the full stop at the end of answers and (2) lowercase both answers. We use the BLEU, Rouge and METEOR scorers provided at <https://github.com/tylin/coco-caption>.

Baselines As baselines, we compare the proposed model with reported results in [49]. Additionally, we include several baselines which we implement by ourselves. This is in the spirit of providing better (and fairer) comparisons. The compared baselines are listed below:

- **Attention Sum Reader (ASR)** [177] - This is a simple baseline for reading comprehension. Aside from the results on [49], we report our own implementation of the ASR model. Our implementation follows [49] closely.
- **Reinforced Reader Ranker (R³)** [108] - This is a state-of-the-art model for open domain question answering, utilizing reinforcement learning to select relevant passages to train the reading comprehension model. Our objective is to get a sense of how well do open-domain models work on understanding narratives.
- **RNET + PG / CPG** [113] - This is a strong, competitive model for paragraph level reading comprehension. We replace the span prediction layer in RNET with a pointer generator (PG) model with the exact setup as our model. We also investigate equipping RNET + PG with our curriculum learning mechanism (curriculum pointer generator).

6.3.2 Experimental Results

Table 6.1 reports the results of our approach on the NarrativeQA benchmark. Results are reported from [49]. The numbers besides the model name denote the total context size. Rel. Gain reports the relative improvement of our model and the best baseline reported in [49] on a specific context size setting.

Our approach achieves state-of-the-art results as compared to prior work [49]. When compared to the best ASR model in [49], the relative improvement across all metrics are generally high, ranging from +17% to +51%. The absolute improvements range from approximately +1% to +3%.

TABLE 6.1: Results on NarrativeQA (Full) reading comprehension dataset.

Model	ℓ	Dev Set				Test Set			
		BLEU-1	BLEU-4	Meteor	Rouge	BLEU-1	BLEU-4	Meteor	Rouge
IR (BLEU)	-	6.73	0.30	3.58	6.73	6.52	0.34	3.35	6.45
IR (ROUGE)	-	5.78	0.25	3.71	6.36	5.69	0.32	3.64	6.26
IR (Cosine)	-	6.40	0.28	3.54	6.50	6.33	0.29	3.28	6.43
BiDAF	-	5.82	0.22	3.84	6.33	5.68	0.25	3.72	6.22
ASR	200	16.95	1.26	3.84	1.12	16.08	1.08	3.56	11.94
ASR	400	18.54	0.00	4.2	13.5	17.76	1.10	4.01	12.83
ASR	1K	18.91	1.37	4.48	14.47	18.36	1.64	4.24	13.4
ASR	2K	20.00	2.23	4.45	14.47	19.09	1.81	4.29	14.03
ASR	4K	19.79	1.79	4.60	14.86	19.06	2.11	4.37	14.02
ASR (Ours)	4K	12.03	1.06	3.10	8.87	11.26	0.65	2.66	8.68
R^3	-	16.40	0.50	3.52	11.40	15.70	0.49	3.47	11.90
RNET-PG	4K	17.74	0.00	3.95	14.56	16.89	0.00	3.84	14.35
RNET-CPG	4K	19.71	2.05	4.91	15.05	19.27	1.45	4.87	15.50
IAL-CPG	4K	23.31	2.70	5.68	17.33	22.92	2.47	5.59	17.67
Rel. Gain	-	+31%	+51%	+23%	+17%	+20%	+17%	+28%	+26%

Pertaining to the models benchmarked by us, we found that our re-implementation of ASR (Ours) leaves a lot to be desired. Consequently, our proposed IAL-CPG model almost doubles the score on all metrics compared to ASR (Ours). The R^3 model, which was proposed primarily for open-domain question answering, does better than ASR (Ours) but still fall shorts. Our RNET-PG model performs slightly better than R^3 but fails to get a score on BLEU-4. Finally, RNET-CPG matches the state-of-the-art performance of [49]. However, we note that there might be distinct implementation differences⁴ with the primary retrieval mechanism and environment/preprocessing setup. A good fair comparison to observe the effect of our curriculum reading is the improvement between RNET-PG and RNET-CPG.

6.3.3 Ablation Study

In this section, we provide an extensive ablation study on all the major components and features of our proposed model. Table 6.2 reports results of our ablation study. (1-3) are architectural ablations. (4-8) are curriculum reading based ablations. (9) investigates RL-based generation. (10-16) explore the understandability/paragraph size heuristic. Note that (10) is the optimal scheme reported in the original setting. Moreover, more permutations were tested but only representative example are reported for brevity.

⁴This is made clear from how our ASR model performs much worse than [49]. We spend a good amount of time trying to reproduce the results of ASR on the original chapter.

TABLE 6.2: Ablation results on NarrativeQA development set.

Ablation	BLEU-1	BLEU-4	Meteor	Rouge
Original Full Setting	23.31	2.70	5.68	17.33
(1) Remove IAL Reader Layer	18.93	1.94	4.52	14.51
(2) Replace regular Self-Attention	19.61	0.96	4.38	15.24
(3) Remove Enhancement	20.25	1.76	4.92	15.14
(4) Remove PG + CR	15.30	0.91	3.85	11.36
(5) Remove CR (understandability)	20.13	2.30	4.94	16.96
(6) Remove CR (answerability)	20.13	1.82	4.92	15.77
(7) Train Easy Only	20.75	1.52	4.65	15.42
(8) Train Hard Only	19.18	1.49	4.60	14.19
(9) Add RL	21.85	2.70	5.31	16.73
(10) $50 \rightarrow 100 \rightarrow 200$	23.31	2.70	5.68	17.33
(11) $50 \rightarrow 100 \rightarrow 200 \rightarrow 500$	21.07	2.86	5.33	16.78
(12) $100 \rightarrow 200 \rightarrow 500 \rightarrow 50$	20.18	2.60	5.50	18.14
(13) $500 \rightarrow 50 \rightarrow 100 \rightarrow 200$	20.95	2.51	5.41	17.05
(14) $500 \rightarrow 200 \rightarrow 100 \rightarrow 50$	17.13	2.38	4.60	15.56
(15) 50 (static)	20.91	2.57	5.35	18.78
(16) 500 (static)	19.36	2.45	4.94	16.00

Attention Ablation In ablations (1-3), we investigated the effectiveness of the self-attention layer. In (1), we removed the entire IAL Reader layer, piping the context-query layer directly to the subsequent layer. In (2), we replaced block-based self-attention with the regular self-attention. Note that the batch size is kept extremely small (e.g., 2), to cope with the memory requirements. In (3), we removed the multiplicative and subtractive features in the IAL Reader layer. Results show that replacing the block-based self-attention with regular self-attention hurts performance the most. However, this may be due to the requirement of reducing the batch size significantly. Removing the IAL layer only sees a considerable drop while removing the enhancement also reduces performance considerably.

Curriculum ablation In ablations (4-8), we investigated various settings pertaining to curriculum learning. In (4), we removed the pointer generator (PG) completely. Consequently, there is also no curriculum reading in this setting. Performance drops significantly in this setting and demonstrates that the pointer generator is completely essential to good performance. In (5-6), we removed one component from our curriculum reading mechanism. Results show that the answerability heuristic is more important than the understandability heuristic. In (7-8), we focused on non-curriculum approaches training on the easy or hard set

only. It is surprising that training on the hard set alone gives considerably decent performance which is comparable to the easy set. However, varying them in a curriculum setting has significant benefits.

RL (Reinforcement Learning) Ablation In ablation (9), we investigated techniques that pass the BLEU-score back as a reward for the model and train the model jointly using reinforcement learning. We follow the setting of [188], using the mixed training objective and setting λ to 0.05. We investigated using BLEU-1, BLEU-4 and Rouge-L (and combinations of these) as a reward for our model along with varying λ rates. Results in Table 6.2 report the best result we obtained. We found that while RL does not significantly harm the performance of the model, there seem to be no significant benefit in using RL for generating answers, as opposed to other sequence transduction problems [188, 189].

Understandability Ablation From ablations (10-16), we study the effect of understandability and alternating paragraph sizes. We find that generally starting from a smaller paragraph and moving upwards perform better and moving the reverse direction may have adverse effects on performance. This is made evident by ablations (10-11). We also note that a curriculum approach beats a static approach often.

6.3.4 Qualitative Error Analysis

Table 6.3 provides some examples of the output of our best model. First, we discuss some unfortunate problems with the evaluation in generation based QA. In example (1), the model predicts a semantically correct answer but gets no credit due to a different form. In (2), no credit is given for word-level evaluation. In (3), the annotators provide a more general answer and therefore, a highly specific answer (e.g., moscow) does not get any credit.

Second, we observe that our model is occasionally able to get the correct (exact match) answer. This is shown in examples (4) and (7). However, there are frequent inability to generate phrases that make sense, even though it seems like the model is trudging along the right direction (e.g., “to wants to be a love of john” versus “because he wants her to have the baby” and “in the york school” versus “east

TABLE 6.3: Qualitative analysis on NarrativeQA development set.

Question	Model Answer	Ground Truth
(1) how many phases did the court complement competition have?	two	2
(2) who suffers from a crack addiction?	dick	dicky
(3) where did john and sophia go to from the airport?	moscow	russia
(4) what country did nadia’s cousin and friend visit her from?	russia	russia
(5) why is nadia kidnapped by alexei?	to wants be a love of john	because he now wants her to have the baby
(6) who does mary marry?	charles who is her	charles
(7) what instrument does roberta guaspari play?	violin	violin
(8) where is the school located where roberta takes a position as a substitute violin teacher?	in the york school	east harlem in new york city
(9) what is the profession of roberta’s husband?	she is a naval	he is in the us navy

harlem in new york”). In (9), we also note a partially correct answer, even though it fails to realize that the question is about a male and generates “she is a naval”.

Possible Directions In order to mitigate these errors and to provide more meaningful evaluations, a likely direction is to provide more semantically grounded evaluation, such as using another parameterized model for evaluation, i.e., having a paraphrase neural model or NLI model assign scores to answers. Another option is to provide human level judgements. However, the latter is expensive in terms of manual labour.

6.4 Summary

We proposed curriculum learning based pointer-generator networks for reading long narratives. Our proposed IAL-CPG model achieves state-of-the-art performance on the challenging NarrativeQA benchmark. We show that sub-sampling diverse views of a story and training them with a curriculum scheme is potentially more

effective than techniques designed for open-domain question answering. We conduct extensive ablation studies and qualitative analysis, shedding light on the task at hand.

Chapter 7

Natural Language Understanding for Recommender Systems

The potential impact of NLU systems is not restricted to QA or standard NLP domains. This chapter presents a compelling use case of using NLU-inspired neural models for improving recommender systems. While neural recommender system research has primarily been focused on interaction data [190–194], we find that NLU-based recommender systems have a lot of potentials. In this chapter¹, we discuss our proposed Multi-Pointer Co-Attention Network (MPCN), a novel NLU-based model for recommendation with reviews.

7.1 Introduction

On most e-commerce platforms today, the ability to write and share reviews is not only a central feature but is also a strongly encouraged act. Reviews are typically informative, pooling an extensive wealth of knowledge for prospective customers. However, the extensive utility of reviews does not only end at this point. Reviews are also powerful in capturing preferences of authors, given the rich semantic textual information that cannot be conveyed via implicit interaction data or purchase logs. As such, there have been immense interest in collaborative filtering systems that

¹This chapter is published as *Multi-Pointer Co-Attention Network for Recommendation*, Proceedings of KDD 2018 [66].

exploit review information for making better recommendations [119, 122, 124, 195–197].

Recent advances in deep learning has spurred on various innovative models that exploit reviews for recommendation [119, 120, 123]. The intuition is simple yet powerful, i.e., each user is represented as all reviews he (she) has written and an item is represented by all reviews that were written for it. All reviews are concatenated to form a single user (item) document. Subsequently, a convolutional encoder is employed to learn a single latent representation for the user (item). User and item embeddings are then matched using a parameterized function such as Factorization Machines [126]. This has shown to be highly performant [123], outperforming a wide range of traditionally strong baselines such as Matrix Factorization (MF). Models such as DeepCoNN [119], TransNets [120] and D-ATT [123] are recent state-of-the-art models that are heavily grounded in this paradigm.

Intuitively, this modeling paradigm leaves a lot to be desired. Firstly, the naive concatenation of reviews into a single document is unnatural, ad-hoc and noisy. In this formulation, reviews are treated indiscriminately irregardless of whether they are important or not. A user’s bad review about a coffee shop should be mostly irrelevant when deciding if a spa is a good match. Secondly, user and item representations are static irregardless of the target match. For example, when deciding if a coffee shop is a good match for a user, the user’s past reviews about other coffee shops (and eateries) should be highly relevant. Conversely, reviews about spas and gyms should not be counted. Hence, a certain level of dynamism is necessary. Finally, the only accessible interaction between user and item is through a fixed dimensional representation, which is learned via excessive compression of large user-item review banks into low-dimensional vector representations. For richer modeling of user and item reviews, deeper and highly accessible interaction interfaces between user-item pairs should be mandatory.

Recall that reviews were fundamentally written independently, at different times, and for different products (or by different people). There should be no reason to squash everything into one long document if they can be modeled independently and then combined later. More importantly, a user may write hundreds of reviews over an extended period of time while an item may effortlessly receive a thousand reviews. As such, existing modeling paradigms will eventually hit a dead-end. Overall, this work proposes four major overhauls that have to be made to existing

models, i.e., (1) reviews should be modeled independently, (2) not all reviews are equally important and should be weighted differently, (3) the importance of each review should be dynamic and dependent on the target match and finally, (4) user and item reviews should interact not only through compressed vector representations but also at a deeper granularity, i.e., word-level.

To this end, we propose a *Multi-Pointer Co-Attention Network* (MPCN), a novel deep learning architecture that elegantly satisfies our key desiderata. Our model is multi-hierarchical in nature, i.e., each user is represented by n reviews of ℓ words each. Subsequently, all user and item reviews (for this particular instance pair) are matched to determine the most informative reviews. In order to do so, we design a novel pointer-based co-attention mechanism. The pointer mechanism extracts the named reviews for direct review-to-review matching. At this stage, another co-attention mechanism learns a fixed dimensional representation, by modeling the word-level interaction between these matched reviews. This forms the crux of our *review-by-review* modeling paradigm. Finally, we introduce a multi-pointer learning scheme that can be executed an arbitrary k times, extracting multiple multi-hierarchical interactions between user and item reviews.

In summary, the prime contributions of this chapter are as follows:

- We propose a state-of-the-art NLU-based neural model for recommendation with reviews. Our proposed model exploits a novel pointer-based learning scheme. This enables not only noise-free but also deep word-level interaction between user and item.
- We conduct experiments on 24 benchmark datasets. Our proposed MPCN model outperforms all state-of-the-art baselines by a significant margin across all datasets. Our compared baselines are highly competitive, encompassing not only review-based models but also state-of-the-art interaction-only models. We outperform models such as Neural Matrix Factorization (NeuMF) [198], DeepCoNN [119], D-ATT [123] and TransNet [120]. Performance improvements over DeepCoNN, TransNets and D-ATT are up to 71%, 19% and 5% respectively.
- We investigate the inner workings of our proposed model and provide insight about how MPCN works under the hood. Additionally, analyzing the behavior of our pointer mechanism allows us to better understand the nature

of the problem. Through analysis of our pointer mechanism, we show that different problem domains have different patterns of ‘*evidence aggregation*’, which might require different treatments and special cares.

7.2 Proposed Method

In this section, we present a layer-by-layer description of our proposed model. The overall model architecture is illustrated in Figure 7.2. Note that this example illustrates a single pointer example. The review gating mechanism and multi-pointer learning are omitted for clarity.

7.2.1 Input Encoding

Our model accepts two input sequences, a (user) and b (item). Each input sequence is a list of reviews $\{r_1, r_2 \dots r_{\ell_d}\}$ where ℓ_d is the maximum number of reviews.

Embedding Layer Each review is a sequence of ℓ_w words which are represented as one-hot encoded vectors. For a and b , we pass all words into an embedding matrix $\mathbf{W}^{d \times |V|}$ where V is the vocabulary, retrieving a d dimensional vector for each word. Our network is hierarchical in nature, i.e., instead of representing all user (or item) reviews as one long document, we represent each user (or item) as a sequence of reviews. Each review is then hierarchically constructed from a sequence of words.

Review Gating Mechanism Each review is represented as a sum of its constituent word embeddings to form the vector $x \in \mathbb{R}^d$. Intuitively, not all reviews that a user writes and not all reviews written for a product is important. We design a first-stage filter, i.e., a review gating mechanism that accepts each review as an input and controls how much information passes through to the next level. Given the input $x \in \mathbb{R}^{\ell_r \times d}$, which represents either a or b .

$$\bar{x}_i = \sigma(\mathbf{W}_g x_i) + \mathbf{b}_g \odot \tanh(\mathbf{W}_u x_i + b_u) \quad (7.1)$$

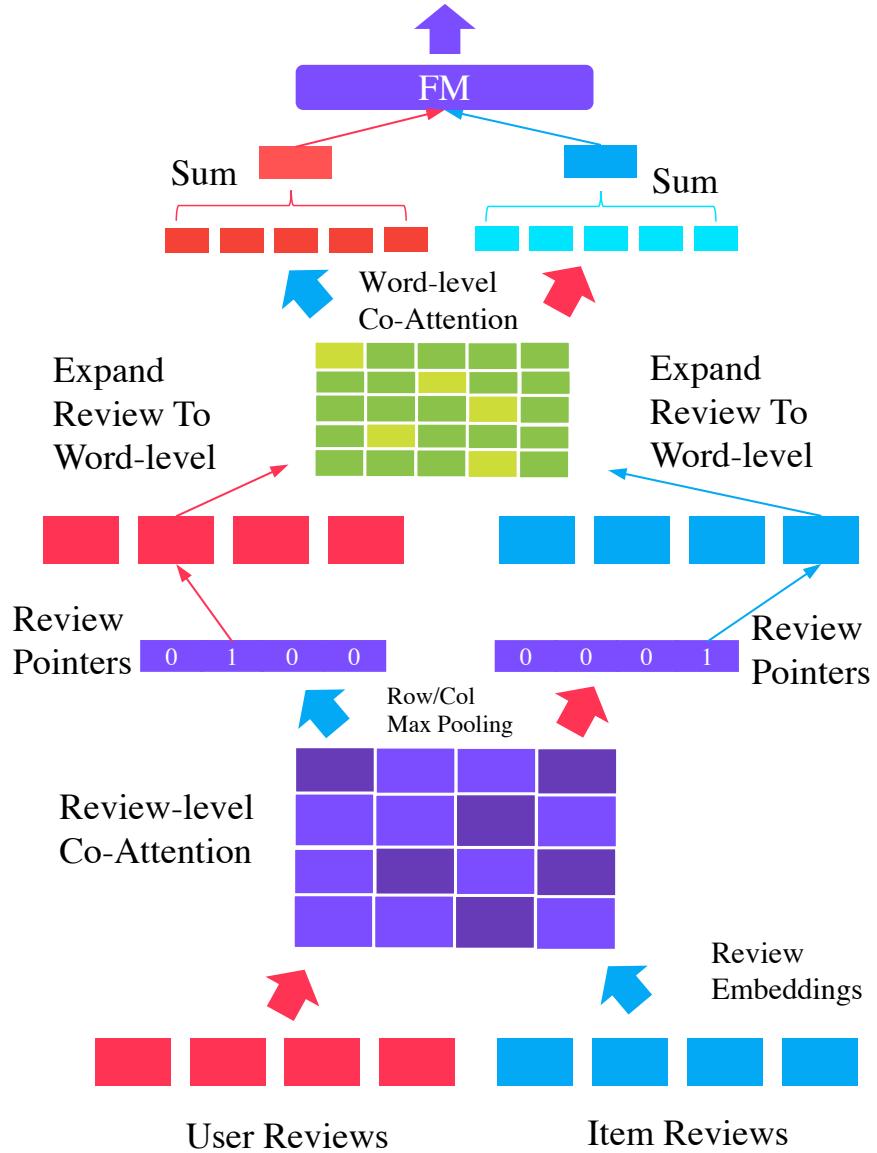


FIGURE 7.1: Architecture of MPCN.

where \odot is the Hadamard product and σ is the sigmoid activation function. x_i is the i -th review of sequence x . $\mathbf{W}_g, \mathbf{W}_u \in \mathbb{R}^{d \times d}$ and $b_g, b_u \in \mathbb{R}^n$ are parameters of this layer. While the purpose of the co-attention layer is to extract important reviews, we hypothesize that applying a pre-filter (gating mechanism) helps improve performance on certain datasets.

7.2.2 Review-level Co-Attention

In this layer, the aim is to select the most informative review from the review bank of each user and item respectively.

Affinity Matrix Given a list of review embeddings from each user ($a \in \mathbb{R}^{\ell_r \times d}$) and item ($b \in \mathbb{R}^{\ell_r \times d}$) banks, we calculate an affinity matrix between them. This is described as follows:

$$s_{ij} = F(a_i)^\top \mathbf{M} F(b_j) \quad (7.2)$$

where $\mathbf{M}^{d \times d}$ and $S \in \mathbb{R}^{\ell_r \times \ell_r}$. $F(\cdot)$ is a feed-forward neural network function with l layers. In practice, l is tuned amongst $[0, 2]$ where $l = 0$ reverts Equation (7.2) to the bilinear form.

Pooling Function By taking the row and column wise maximum of the matrix s and using them to weight the original list of reviews a and b , we are able to derive the standard co-attention mechanism. This is described as follows:

$$a' = (G(\max_{\text{col}}(s)))^\top a \quad \text{and} \quad b' = (G(\max_{\text{row}}(s)))^\top b \quad (7.3)$$

There are different choices for the pooling operation. Max pooling is used here because, intuitively it selects the review which has the maximum influence (or affinity) with all reviews from its partner. This type of co-attention is known to be extractive, characterized by its usage of max pooling.

Note that we apply the function $G(\cdot)$ to $\max_{\text{col}}(s)$ and $\max_{\text{row}}(s)$. In most applications, $G(\cdot)$ would correspond to the standard softmax function, which converts the input vector into a probability distribution. The vectors a', b' would then be the *co-attentional vector representations*. However, in our case, we desire further operations on the selected reviews and therefore do not make use of these representations. Instead, $G(\cdot)$ has to return a one-hot encoded vector, pointing to the selected reviews which forms the real objective behind this co-attentional layer. However, the Softmax function returns a continuous vector, which is unsuitable for our use case. The usage of discrete vectors in neural architectures is known to be difficult, as the arg max operation is non-differentiable. Hence, we leverage

a recent advance, the Gumbel-Max trick, for learning to point. The next section describes this mechanism.

7.2.3 Review Pointers

We leverage a recent advance, the Gumbel-Softmax [199], for incorporating discrete random variables in the network.

Gumbel-Max In this section, we describe Gumbel-Max [200], which facilitates the key mechanism of our MPCN model. Gumbel-Max enables discrete random variables (e.g., one-hot vectors) to be utilized within an end-to-end neural network architecture. Consider a k -dimensional categorical distribution where class probabilities p_1, \dots, p_k are defined in terms of unnormalized log probabilities π_1, \dots, π_k :

$$p_i = \frac{\exp(\log(\pi_i))}{\sum_{j=1}^k \exp(\log(\pi_j))} \quad (7.4)$$

A one-hot sample $z = (z_1, \dots, z_k) \in \mathbb{R}^k$ from the distribution can be drawn by using the following:

$$z_i = \begin{cases} 1, & i = \arg \max_j (\log(\pi_j) + g_j) \\ 0, & \text{otherwise} \end{cases} \quad (7.5)$$

$$g_i = -\log(-\log(u_i)) , \quad u_i \sim Uniform(0, 1) \quad (7.6)$$

where g_i is the *Gumbel noise* which perturbs each $\log(\pi_i)$ term such that the arg max operation is equivalent to drawing a sample weighted by p_1, \dots, p_k .

Straight-Through Gumbel-Softmax In the Gumbel-Softmax, the key difference is that the arg max function is replaced by the differentiable softmax function which is described as follows:

$$y_i = \frac{\exp(\frac{\log(\pi_i) + g_i}{\tau})}{\sum_{j=1}^k \exp(\frac{\log(\pi_j) + g_j}{\tau})} \quad (7.7)$$

where τ , the temperature parameter, controls the extend of how much the output approaches to a one hot vector. More concretely, as τ approaches to zero, the

output of the Gumbel-Softmax distribution becomes cold, i.e., becomes closer to a one-hot vector. In the straight-through (ST) adaptation, the forward-pass is discretized by converting the vector output to a one-hot vector via arg max:

$$y_i = \begin{cases} 1, & i = \arg \max_j(y_j) \\ 0, & \text{otherwise} \end{cases} \quad (7.8)$$

However, the backward pass maintains the flow of continuous gradients which allows the model to be trained end-to-end. This is useful as we only want important reviews to be selected (i.e., hard selection) for consideration in computation of the loss function. Notably, alternatives such as REINFORCE [201] exist. However, it is known to suffer from high variance and slow convergence [199].

Learning to Point In order to compute a review pointer (for user and item), we set $G(\cdot)$ in Equation (7.3) to use the Gumbel Softmax. However, since we are interested only in the pointer (to be used in subsequent layers), the pointer is then calculated as:

$$p_a = (Gumbel(\max_{\text{col}}(s))) \quad \text{and} \quad p_b = (Gumbel(\max_{\text{row}}(s))) \quad (7.9)$$

By applying p_a to a , we select the p_a -th review of user a and p_b -th review of item b . The selected reviews are then passed into the next layer where rich interactions are extracted between these reviews.

7.2.4 Word-level Co-Attention

The review-level co-attention smooths over word information as it compresses each review into a single embedding. However, the design of the model allows the most informative reviews to be extracted by the use of pointers. These reviews can then be compared and modeled at word-level. This allows a user-item comparison of finer granularity which facilitates richer interactions as compared to simply composing the two review embeddings. Let \bar{a}, \bar{b} be the selected reviews using the pointer learning scheme. Similar to the review-level co-attention, we compute a similarity matrix between \bar{a} and \bar{b} . The key difference is that the affinity matrix is computed

word-by-word and not review-by-review.

$$w_{ij} = F(\bar{a}_i)^\top \mathbf{M}_w F(\bar{b}_j) \quad (7.10)$$

where $\mathbf{M}_w \in \mathbb{R}^{d \times d}$ and $w \in \mathbb{R}^{\ell_w \times \ell_w}$. Next, to compute the *co-attentional* representation of reviews \bar{a}, \bar{b} , we take the mean pooling.

$$\bar{a}' = (S(\text{avg}_{\text{col}}(w)))^\top \bar{a} \quad \text{and} \quad \bar{b}' = (S(\text{avg}_{\text{row}}(w)))^\top \bar{b} \quad (7.11)$$

where $S(\cdot)$ is the standard Softmax function and $F(\cdot)$ is a standard feed-forward neural network with l layers. The rationale for using the average pooling operator here is also intuitive. At the review-level, a large maximum affinity score of a review with respect to all *opposite* reviews (even when a low average affinity score) warrants it being extracted, i.e., a strong signal needs to be further investigated. However, at a word-level, max-pooling may be biased towards identical words and may have a great inclination to act as a word matching operator. Hence, we want to maintain a stable co-attentive extractor. Our early empirical experiments also justify this design. Finally \bar{a}' and \bar{b}' are the output representations.

Note that, the implementation of co-attention layers (review-level and word-level) is equivalent to only two simple MATMUL operations (in Tensorflow). As such, scalability is not really a concern in our approach since this is quite efficiently optimized on GPUs.

7.2.5 Multi-Pointer Learning

While our objective is to eliminate noisy reviews by the usage of hard pointers, there might be insufficient information if we point to only a single pair of reviews. Hence, we devise a multi-pointer composition mechanism. The key idea is to use multiple pointers where the number of pointers n_p is a user-defined hyperparameter. Our model runs the Review-level Co-Attention n_p times, with each generating a unique pointer. Each of the n_p review pairs is then modeled with the Word-level Co-Attention mechanism. The overall output is a list of vectors $\{\bar{a}'_1, \dots, \bar{a}'_{n_p}\}$ and $\{\bar{b}'_1, \dots, \bar{b}'_{n_p}\}$. Additionally, we also found it useful to include the sum embedding of all words belonging to the user (item). This mainly helps in robustness, in case where user and item do not have any matching signals that were found by our

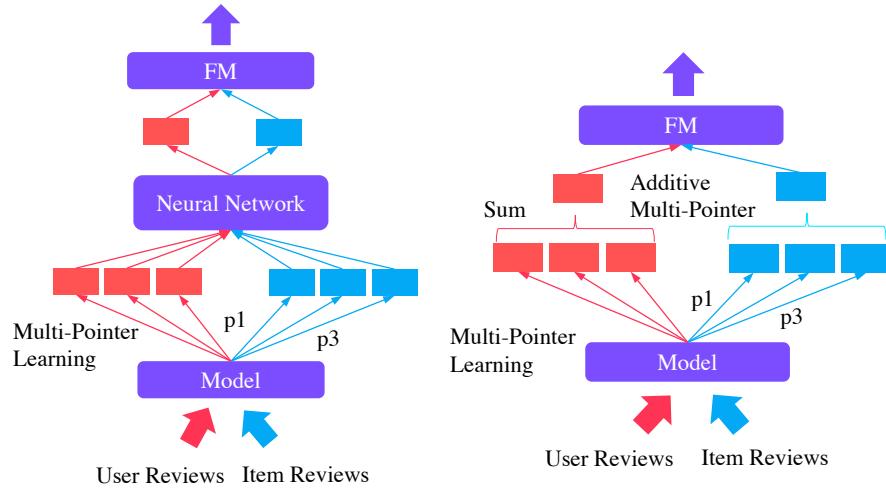


FIGURE 7.2: Neural Network and Additive based Multi-Pointer Learning.

pointer mechanism. We explore three ways to compose these embeddings (Figure 7.2).

- **Concatenation** - All pointer outputs are concatenated, e.g., $[\bar{a}'_1; \dots; \bar{a}'_{n_p}]$. This has implications to the subsequent layers, especially if n_p is large. Hence, we consider the next two alternatives.
- **Additive Composition** - All pointer outputs are summed, e.g., $\text{sum}(\bar{a}'_1, \dots, \bar{a}'_{n_p})$.
- **Neural Network** - All pointer outputs are concatenated and passed through a single non-linear layer with ReLU (σ_r) activations, e.g., $\sigma_r(W([\bar{a}'_1; \dots; \bar{a}'_{n_p}]) + b)$ which maps the concatenated vector into a d dimensional vector.

Note that this is applied to \bar{b}' as well but omitted for brevity. Let the output of this layer be a_f and b_f . In our experiments, we tune amongst the three above-mentioned schemes. More details are provided in the ablation study.

7.2.6 Prediction Layer

This layer accepts a_f, b_f as an input. The concatenation of $[a_f; b_f]$ is passed into a factorization machine (FM) [126]. FM accepts a real-valued feature vector and models the pairwise interactions between features using factorized parameters. The

FM function is defined as follows:

$$F(x) = w_0 + \sum_{i=1}^n w_i x_i + \sum_{i=1}^n \sum_{j=i+1}^n \langle v_i, v_j \rangle x_i x_j \quad (7.12)$$

where $x \in \mathbb{R}^k$ is a real-valued input feature vector. $\langle \cdot, \cdot \rangle$ is the dot product. The parameters $\{v_1 \dots v_n\}$ are factorized parameters (vectors of $v \in \mathbb{R}^k$) used to model pairwise interactions (x_i, x_j) . w_0 is the global bias and $\sum_{i=1}^n w_i x_i$ represents a linear regression component. The output of $F(x)$ is a scalar, representing the strength of the user-item interaction. The network is trained end-to-end by minimizing the standard mean squared error loss following [123].

7.3 Experiments

In this section, we present our experimental setup and empirical evaluation.

7.3.1 Datasets

We utilize datasets from two different sources which are described as follows:

- **Yelp Dataset Challenge** - Yelp is an online review platform for businesses such as restaurants, bars, spas, etc. We use the dataset from the latest challenge².
- **Amazon Product Reviews** - Amazon is a well-known e-commerce platform. Users are able to write reviews for products they have purchased. We use **23** datasets from the Amazon Product Review corpus³ [127, 128].

In total, we provide model comparisons over **24** benchmark datasets. For all datasets, we split interactions into training, development and testing sets. we utilize a time-based split, i.e., the last item of each user is added to the test set while the penultimate is used for development. For Amazon, the datasets are pre-processed in a 5-core fashion (i.e., each user and item have at least 5 reviews to be

²<https://www.yelp.com/dataset/challenge>

³<http://jmcauley.ucsd.edu/data/amazon/>

included). For Yelp, we use a 20-core setting, providing a comparison on a denser dataset. We tokenize the reviews using NLTK and retain words that appear at least 10 times in the vocabulary. We would like to emphasize that, when building user and item representations using their respective reviews, all reviews belonging to interactions from the test and development sets **were not included**. This is to prevent the problem from reverting to a sentiment analysis task, albeit noisier [120].

7.3.2 Compared Methods

We compare against a series of competitive baselines.

- **Matrix Factorization** (MF) - This is a standard and well-known baseline for CF. It represents the user and item rating with the inner product, i.e., $p^\top q$.
- **Factorization Machine** (FM) [126] - This is a general purpose machine learning algorithm that uses factorized parameters to model pairwise interaction within a real-valued feature vector. We concatenate the user-item latent embedding together and pass it through the FM model.
- **Multi-layered Perceptron** (MLP) - This is a strong neural baseline for CF and was used as a baseline in [198]. We use the same pyramidal scheme of 3 layers.
- **Neural Matrix Factorization** (NeuMF) [198] - This is the state-of-the-art model for interaction-only CF. It casts the MF model within a neural framework and combines the output with multi-layered perceptrons.
- **Deep Co-Operative Neural Networks** (DeepCoNN) [119] - This is a review-based convolutional recommendation model. It trains convolutional representations of user and item, and passes the concatenated embedding into a FM model.
- **TransNet** [120] - This is an improved adaptation of DeepCoNN which incorporates transform layers and an additional training step that enforces the transformed representation to be similar to the embedding of the actual target review.

- **Dual Attention CNN Model (D-ATT)** [123] - This is a recently proposed state-of-the-art CNN-based model that uses reviews for recommendation. This model is characterized by its usage of two forms of attentions (local and global). A final user (item) representation is learned by concatenating representations learned from both local and global attentions. The dot product between user and item representations is then used to estimate the rating score.

Given our already extensive comparisons against the state-of-the-art models, we omit comparisons with models such as HFT [122], Collaborative Topic Regression [125], Collaborative Deep Learning (CDL) [202] and ConvMF [121] since they have been outperformed by the recently proposed DeepCoNN or D-ATT model.

7.3.3 Experimental Setup

The evaluation metric is the well-known (and standard) mean-squared error which measures the square error between the rating prediction and ground truth. We implement all models ourselves in Tensorflow. All models are trained with Adam optimizer [153] with an initial learning rate of 10^{-3} . We train all models for a maximum of 20 epochs with early stopping (i.e., if model performance does not improve for 5 epochs) and report the test result from the best performing model on the development set. We found that models tend to converge before 20 epochs. However, an exception is that the MF baseline requires many more epochs to converge. As such, we train the MF model till convergence. For interaction only models, the embedding size is set to 50. For TransNet and DeepCoNN, the number of filters is set to 50 and the filter size is 3. For D-ATT, the global attention layer uses filter sizes of [2, 3, 4]. The word embedding layer is also set to 50 dimensions. We regularize models with a dropout rate of 0.2 and a fixed L2 regularization of 10^{-6} . Dropout is applied after all fully-connected and convolutional layers. We use two transform layers in the TransNet model. All word embeddings are learned from scratch as we found that using pretrained embeddings consistently degrades performance across all datasets (and models). The maximum document length is set to 600 words (20 reviews of 30 tokens each) which we empirically found to be a reasonable length-specific performance bound. We assign a special delimiter token to separate reviews within a user (item) document for DeepCoNN, TransNet, and

TABLE 7.1: Results on benchmark datasets.

Dataset	Interaction-based				Review-based				Improvement (%)		
	MF	FM	MLP	NMF	D-CON	TNET	D-ATT	MPCN	Δ_{DC}	Δ_{TN}	Δ_{DA}
Yelp17	1.735	1.726	1.727	1.691	1.385	1.363	1.428	1.349	2.7	1.0	5.9
Instant Video	2.769	1.331	1.532	1.451	1.285	1.007	1.004	0.997	28.9	1.0	0.7
Instruments	6.720	1.166	1.081	1.189	1.483	1.100	0.964	0.923	60.7	19.2	4.4
Digital Music	1.956	1.382	1.361	1.332	1.202	1.004	1.000	0.970	23.9	3.5	3.1
Baby	1.755	1.614	1.585	1.598	1.440	1.338	1.325	1.304	10.4	2.6	1.6
Patio / Lawn	2.813	1.311	1.279	1.251	1.534	1.123	1.037	1.011	51.7	11.1	2.6
Gourmet Food	1.537	1.348	1.442	1.464	1.199	1.129	1.143	1.125	6.6	0.4	1.6
Automotive	5.080	1.599	1.071	1.013	1.130	0.946	0.881	0.861	31.2	9.9	2.3
Pet Supplies	1.736	1.618	1.649	1.646	1.447	1.346	1.337	1.328	9.0	1.4	0.7
Office Products	1.143	0.998	1.122	1.069	0.909	0.840	0.805	0.779	16.7	7.8	3.3
Android Apps	1.922	1.871	1.805	1.842	1.517	1.515	1.509	1.494	1.5	1.4	1.0
Beauty	1.950	1.711	1.631	1.552	1.453	1.404	1.409	1.387	4.8	1.2	1.6
Tools / Home	1.569	1.310	1.356	1.314	1.208	1.122	1.101	1.096	10.2	2.4	0.5
Video Games	1.627	1.665	1.576	1.568	1.307	1.276	1.269	1.257	4.0	1.5	1.0
Toys / Games	1.802	1.195	1.286	1.222	1.057	0.974	0.982	0.973	8.6	0.1	0.9
Health	1.882	1.506	1.522	1.415	1.299	1.249	1.269	1.238	4.9	0.9	2.5
CellPhone	1.972	1.668	1.622	1.606	1.524	1.431	1.452	1.413	7.9	1.3	2.8
Sports	1.388	1.195	1.120	1.299	1.039	0.994	0.990	0.980	6.0	1.4	1.0
Kindle Store	1.533	1.217	1.231	1.398	0.823	0.797	0.813	0.775	6.2	2.8	4.9
Home / Care	1.667	1.547	1.584	1.654	1.259	1.235	1.237	1.220	3.2	1.2	1.4
Clothing	2.396	1.492	1.462	1.535	1.322	1.197	1.207	1.187	11.4	0.8	1.7
CDs / Vinyl	1.368	1.555	1.432	1.368	1.045	1.010	1.018	1.005	4.0	0.5	1.3
Movies / TV	1.690	1.852	1.518	1.775	1.960	1.176	1.187	1.144	71.3	2.8	3.8
Electronics	1.962	2.120	1.950	1.651	1.372	1.365	1.368	1.350	1.6	1.1	1.3

D-ATT. If FM is used, the number of factors is set to 10. For our proposed model, the number of pointers p is tuned amongst $\{1, 3, 5, 8, 10\}$. On most datasets, the optimal performance is reached with $2 - 3$ pointers.

7.3.4 Experimental Results

Table 7.1 reports the results of our experiments. The best performance is in bold-face. $\Delta_{DC}, \Delta_{TN}, \Delta_{DA}$ are the relative improvements (%) of MPCN over DeepCoNN (D-CON), TransNet (T-NET) and D-ATT respectively. MPCN achieves state-of-the-art performance, outperforming all existing methods on 24 benchmark datasets.

Firstly, we observe that our proposed MPCN is the top performing model on all 24 benchmark datasets. This ascertains the effectiveness of our proposed model. MPCN consistently and significantly outperforms DeepCoNN, TransNet, and D-ATT, which are all recent competitive review-based methods for a recommendation. The relative improvement is also encouraging with gains of up to 71% (DeepCoNN), 19% (TransNet) and 5% (D-ATT). On a majority of the datasets, performance gains are modest, seeing an improvement of 1% – 3% for most models. Notably,

the average percentage improvement of MPCN over DeepCoNN is 16%. The average performance gain over TransNet and D-ATT is a modest 3.2% and 2.2% respectively.

Pertaining to the relative ranking of the review-based models, our empirical evaluation reaffirms the claim of [120], showing that TransNet always outperforms DeepCoNN. However, the relative ranking of D-ATT and TransNet switches positions frequently. Notably, TransNet uses the test review(s) as an additional data source (albeit as a training target) while D-ATT does not make use of this information. The additional training step of TransNet is actually quite effective and hypothetically could be used to enhance D-ATT or MPCN as well. However, we leave that for future work.

Next, the performance of interaction-only models (MF, FM, etc.) is consistently lower than review-based models (e.g., DeepCoNN). The relative performance of all interaction models is generally quite inconsistent over various datasets. However, one consistent fact is that MF performs worse than other models most of the time. The top scoring interaction model often switches between FM and MLP.

Finally, we give readers a sense of computational runtime. We provide an estimate that we found generally quite universal across multiple datasets. Let t be the runtime of DeepCoNN, the runtime of MPCN $p = 1$ is approximately $\approx 0.4t$. MPCN with 2 and 3 pointers are $0.8t$ and $1.2t$ respectively. TransNet and D-ATT run at $\approx 2t$. On medium size datasets (e.g., Amazon Beauty), $t \approx 40s$ on a GTX1060 GPU (batch size is 128). We found that if $p_{opt} \leq 2$, then MPCN is faster than DeepCoNN. While $p_{opt} \leq 5$ is the threshold for being equal with D-ATT and TransNet in terms of runtime.

7.3.5 Hyperparameter & Ablation Analysis

In this section, we study the impact of key hyperparameter settings and various architectural choices on model performance.

TABLE 7.2: Ablation analysis on Amazon Product Reviews dataset.

Architecture	Beauty	Office	M-Instr	Inst-Vid
(0) Default	1.290	0.770	0.827	0.975
(1) Remove Gates	1.299	0.760	0.837	0.979
(2) Remove FM	1.286	0.808	0.924	0.997
(3) Aggregation (Concat)	1.279	0.788	0.832	0.971
(4) Aggregation (Additive)	1.290	0.767	0.829	0.971
(5) set $l = 0$	1.293	0.776	0.830	0.976
(6) set $l = 2$	1.295	0.775	0.829	0.974
(7) Remove WLCA	1.296	0.778	0.831	0.999
(8) Remove RLCA	1.304	0.789	0.839	0.1003

7.3.5.1 Ablation Analysis

In this section, we study the impacts of various architectural decisions on model performance. Table 7.2 reports an ablation analysis conducted on the development sets of four benchmark datasets (*Beauty*, *Office*, *Musical Instruments (M-Instr)* and *Amazon Instant Video (Inst-Vid)*). We report the results of several different model variations. We first begin describing the default setting in which ablation reports are deviated from. In the default setting, we use the standard model with all components (review gates, word-level co-attention, and FM prediction layer). The Multi-Pointer aggregation is set to use a neural network (single layer nonlinear transform). The number of layers in the co-attention layer is set to $l = 1$.

We report the validation results from 8 different variations, with the aims of clearly showcasing the importance of each component. In (1), we removed the review gating mechanism. In (2), we replaced the FM with the simple inner product. In (3-4), we investigated the effects of different pointer aggregation functions. They are the concatenate and additive operators respectively. In (5-6), we set $l = 0$ (remove layer) and $l = 2$. In (7), we removed the word level co-attention layer. In this case, the representation for users and items is simply the pointed review embedding. In (8), we removed the review-level co-attention (RLCA). This variation is no longer ‘hierarchical’, and simply applies word-level co-attention to user and item reviews.

Firstly, we observe the default setting is not universally the best across four datasets. As mentioned, the review gating mechanism helps in 3 out of 4 presented datasets. In the *Office* dataset, removing the review gating layer improves

TABLE 7.3: Validation MSE when varying the number of pointers.

Ptr	Patio	Automotive	Sports	Video Games
1	0.992	0.842	0.926	1.885
2	0.980	0.855	0.933	1.178
3	0.975	0.843	0.938	1.194
4	0.991	0.844	0.938	1.189
5	0.991	0.844	0.935	1.121

performance. We found this to be true across most datasets, i.e., the review gating mechanism helps more often than not, but not always. The impact of removing FM is quite easily noticed, leading to huge performance degradation on the *M-Instr* dataset. Deprovement on *Inst-Vid* and *Office* is also significant. On the other hand, removing FM marginally improved performance on *Beauty*. We also discovered that there is no straightforward choice of aggregation functions. Notably, the relative ranking of all three variants (concat, additive, and neural network) are always interchanging across different datasets. As such, they have to be tuned. We also noticed that the choice of $l = 1$ is safe across most datasets, as increasing or decreasing would often lead to performance degradation. Finally, removing the WLCA and RLCA consistently lowered performance on all datasets, which ascertains the effectiveness of the WLCA layer. Notably, removing RLCA seems to hurt performance more, which signifies that modeling at review-level is essential.

7.3.5.2 Effect of Number of Pointers

Table 7.3 reports the effect of varying pointers on performance. We use 4 datasets of varying sizes as an illustrative example (*Patio*, *Automotive*, *Sports* and *Video Games*). The datasets shown are sorted from smallest to largest in terms of the number of interactions. Clearly, we observe that the optimal number of pointers varies across all datasets. We found this to be true for the remainder datasets that are not shown. This seems to be more of a domain-dependent setting since we were not able to find any correlation with dataset size. For most datasets, the optimal pointers fall in the range of 1 – 3. In exceptional cases (*Video Games*), the optimal number of pointers was 5.

TABLE 7.4: Excerpts from the MPCN’s pointer mechanism.

User Review	Item Review
game is really beautiful! coolest rpg ever...	..a gift for a friend who really loves rpg games. he really loved it!
.. game is completely turned based, so you have time to ponder your actions..	..is a great and engaging puzzler game but wasn’t too challenging
just love this little guy ... phone is reasonably easy to put in and take out	..case really fits the 5s like a glove..
is a nice charger..but after a few months it wasn’t charging...	it is clearly a used or refurbished battery..
cocoa is a wonderful, rich tasting, not overly sweet product ..	used to eat the dark and milk chocolate, and then we tried this and can’t explain how good they are

7.3.6 In-Depth Model Analysis

In this section, we present several insights pertaining to the inner workings of our proposed model.

7.3.6.1 What are the pointers pointing to?

We list some observations by analyzing the behavior of the MPCN model. Firstly, we observed that pointers react to aspects and product sub-categories. In many cases, we observed that the pointer mechanism tries to find the most relevant review written by the user for a given product. If the target item is a phone case, then our model tries to find a review written by the user which is directed towards *another* phone case product. Intuitively, we believe that this is trying to solicit the user’s preferences about a type of product. We provide some qualitative examples in Table 7.4. Consider the context of video games, it finds that the user has written a review about *rpg* (roleplaying games). At the same time, it finds that the item review consists of a (positive) review of the item being a good rpg game. As a result, it surfaces both reviews and concurrently points to both of them. This follows suit for the other examples, e.g., it finds a matching clue of *puzzle games* (turn-based) in the second example. The last example is taken from the *Gourmet Food* dataset. In this case, it discovers that the user likes cocoa, and concurrently finds out that the product in question has some relevant information about chocolate.

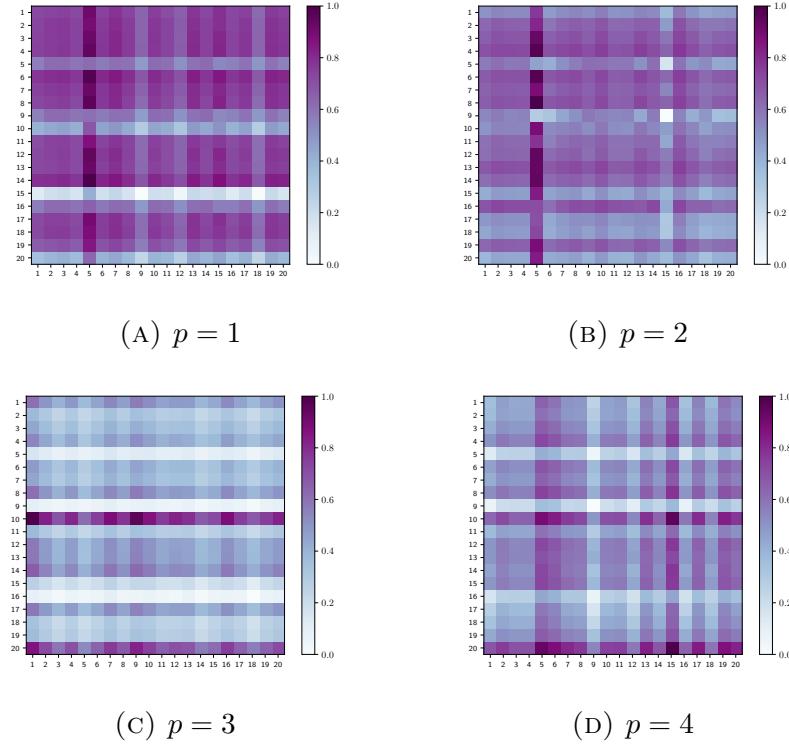


FIGURE 7.3: Visualization of Review-level Co-Attention.

7.3.6.2 Behavior of Multi-Pointer Learning

We study the behavior of our multi-pointer mechanism. First and foremost, this serves as another *sanity check* and to observe if multi-pointers are really necessary, i.e., if pointers are not pointing all to the same reviews. Hence, this section aims to provide better insight into the inner workings of our proposed model. We trained an MPCN model with four pointers. A quick observation is that all four pointers point to different reviews (given the same user-item pair). This is automatic, and does not require any special optimization constraint (such as explicitly enforcing the model to choose different reviews through an extra optimization term). Moreover, we analyze the affinity matrix from the review-level co-attention. Figure 7.3 shows the affinity matrix for pointers one to four. Matching patterns differ significantly across multiple calls, hence generating different pointers.

Secondly, it is also intuitive that it is not absolutely necessary for MPCN to always point to unique reviews given the same user-item pair. We observed a relatively significant *one-to-many* pointer pattern on top of the usual *one-to-one* pattern. In this case, the same review for user (item) is being matched with n (many)

TABLE 7.5: Analysis of multi-pointer behavior of MPCN.

Condition	D-Music	Apps	V-Games	Food	Yelp17
(1) All unique	85.2%	87.5%	82.0%	99.2%	99.2%
(2) 1 Repeated	13.2%	12.5%	17.2%	0.7%	0.7%
(3) All Repeated	1.6%	0.0%	1.6%	0.0%	0.0%
(4) One-to-Many	64.8%	57.8%	57.8%	23.4%	43.7%

different reviews from the item (user). This is also observed to be dataset/domain dependent. In a small minority of cases, all pointers pointed to the same reviews constantly (all repeated condition). However, this is understandable as there is just insufficient information in the user and item review bank. Additionally, we analyzed a small sample from the test set, determining if any of the following conditions hold for each test case.

Table 7.5 reports the percentages of test samples that fall into each category. We report results on five datasets *Digital Music (D-Music)*, *Android Apps (Apps)*, *Video Games (V-Games)*, *Gourmet Food (Food)* and *Yelp17*. Here, we observe that pointer behavior is largely influenced by domain. The first three are concerned with electronic domains while the last two are more focused on food (and restaurants). We clearly observe that Food and Yelp have very similar pointer behavior. In general, the electronic domains usually make an inference using fewer subsets of reviews. This is made evident by the high one-to-many ratio which signifies that there is often one important review written by the user (or item) that contributes more (and needs to be matched with multiple opposing reviews). Conversely, the food domains require more evidence across multiple reviews. We believe this is one of the biggest insights that our work offers, i.e., shedding light on how evidence aggregation works (and differs across domains) in the review-based recommendation.

7.4 Summary

In this chapter, we proposed a new state-of-the-art neural model for recommendation with reviews. The proposed Multi-Pointer Co-Attention Networks outperforms many strong competitors on 24 benchmark datasets from Amazon and Yelp,

demonstrating the effectiveness of NLU-based deep learning for recommender systems. We conduct extensive analysis on the inner workings of our proposed multi-pointer learning mechanism. By analyzing the pointer behavior across multiple domains, we conclude that different domains (such as food-related and electronics-related) have different ‘*evidence aggregation*’ patterns. The source code for this chapter can be found at https://github.com/vanzytay/KDD2018_MPCN.

Chapter 8

Multi-Granular Sequence Encoders for Natural Language Understanding

Natural language understanding (NLU) and Machine Reading Comprehension (MRC) have been extremely productive areas of research in recent years, giving rise to many highly advanced neural network architectures [25, 34, 110, 111, 116, 175, 203]. A common denominator in many of these models is the compositional encoder, i.e., usually a bidirectional recurrent-based (LSTM [18] or GRU [19]) encoder that sequentially parses the text sequence word-by-word. This helps to model the compositionality of words, capturing rich and complex linguistic and syntactic structure in language. In this chapter¹, we discuss our proposed Dilated Composition Units (DCU) for encoding sequences.

8.1 Introduction

While the usage of a recurrent encoder is often regarded as indispensable in highly complex NLU/MRC tasks, there are still several challenges and problems pertaining to its usage in modern NLU/MRC tasks. Firstly, documents can be extremely

¹This chapter is published as *Multi-granular Sequence Encoding via Dilated Composition Units for Reading Comprehension*, Proceedings of EMNLP 2018 [67].

long to the point where running a BiRNN model across a long document is computationally prohibitive. This is aggravated since NLU/MRC tasks can be easily extended to reasoning over multiple long documents. Secondly, recurrent encoders have limited access to long-term (far) context since each word is sequentially parsed. This restricts any form of multi-sentence and intra-document reasoning from happening within the compositional encoder layer.

To this end, we propose a new compositional encoder (Dilated Composition Units) that can either be used in place of standard RNN encoders or serve as a new module that is complementary to existing neural architectures. Our proposed encoder leverages *dilated compositions* to model relationships across multiple granularities. That is, for a given word in the target sequence, our encoder exploits both long-term (far) and short-term (near) information to decide how much information to retain for it. Intuitively, this can be interpreted as learning to compose based on modeling relationships between word-level, phrase-level, sentence-level, paragraph-level and so on. The output of the dilated composition mechanism acts as gating functions, which are then used to learn compositional representations of the input sequence.

A brief high-level overview of our proposed encoder is given as follows: Firstly, sequences are chunked into blocks based on user-defined (hyperparameter) block sizes. Block sizes are often dilated in nature, i.e., 1, 2, 4, 10, 25, etc., in order to capture more long-term (far) information. Our encoder takes the neural bag-of-words representation of each block size and compresses/folds all words (that reside in each block) into a single summed embedding. All blocks are then passed into fully-connected layers and re-expanded/unfolded to their original sequence lengths. For each word, the gating vectors are then constructed by modeling the relationships between all blocks that this word resides in. As such, this can be interpreted as a divide-and-conquer sequence encoding method.

This has several advantages. Firstly, we enable a major speedup by avoiding either costly step-by-step gate construction while still maintaining interactions between neighboring words. As such, our model belongs to a class of architectures which is inspired by QRNNs [204] and SRUs [205]. The key difference is that our gates are not constructed by convolution layers but explicit block-based matching across multiple ranges, both long and short. Secondly, modeling at a long-range (e.g., 25 or 50) enables our model to look further ahead as opposed to only one step forward.

As such, the learned gates possess not only information about nearby words but also a larger overview of the context. This is in similar spirit to self-attention, albeit occurring within the encoder. Thirdly, the final gates are formed by modeling relationships between multi-range projections (n -gram blocks), allowing for fine-grained intra-document relationships to be captured. Our work is mainly concerned with designing an efficient encoder that is able to capture not only compositional information but also long-range and short-range information. More specifically, our proposed recurrent DCU (Dilated Composition Unit) encoder takes on a similar architecture to Quasi-Recurrent Neural Networks [204] and Simple Recurrent Units [205]. In these models, gates are pre-learned and then applied. However, different from existing models such as QRNNs that use convolution layers as gates, we use block-based fold-unfold layers for learning gates. Our model also draws inspiration from dilation, in particular, dilated RNNs [133] and dilated convolutions [206], that intuitively help to model long-range dependencies. Notably, our work is orthogonal to recent advances that are targeted at speeding up the reading process. Such works include residual dilated convolutions [207], self-attention [208] and coarse-to-fine grained paradigm [209]. However, while speed is one of the clear benefits of this work, our work is the first to introduce the idea of block-based multi-granular reasoning. we believe that this new building block is complementary/useful to the NLU/MRC task in general.

The overall contributions of our work are as follows:

- We propose DCU (Dilated Compositional Units), a new compositional encoder for both fast and expressive sequence encoding. we propose an overall architecture that utilizes DCU within a Bi-Attentive framework for both multiple choice and span prediction NLU/MRC tasks. DCU can be used as a standalone (without RNNs) for fast reading and/or together with RNN models (i.e., DCU-LSTM) for more expressive reading.
- We conduct extensive experiments on three large-scale and challenging NLU/MRC datasets - RACE [43], SearchQA [48] and NarrativeQA [64]. Our model is lightweight, fast and efficient, achieving state-of-the-art or highly competitive performance on three datasets.

- Despite its simplicity, our model outperforms highly complex models such as Dynamic Fusion Networks (DFN) [210] on RACE. While DFN takes approximately a week to train, spending at least several hours per epoch, our model converges in less than 12 hours with only 4 – 5 minutes per epoch. Moreover, our model outperforms DFN by 2%–6% on the RACE benchmark and other strong baselines such as the Gated Attention Reader by 10%. On RACE, we outperform DFN without any recurrent and convolution layers. Ablation studies show an improvement of up to 6% when using DCU over a LSTM/GRU encoder.

8.2 Proposed Method

In this section, we describe our proposed Dilated Compositional Unit (DCU) encoder, along with a general Bi-Attention framework that incorporates our DCU encoders.

8.2.1 Dilated Compositional Units (DCU)

This section introduces our proposed DCU encoders. Figure 8.1 provides an illustration of the overall encoder architecture. It presents a high-level overview of (1) our proposed DCU encoder (left), (2) Span Prediction Architecture (center) and (3) Multiple Choice Architecture (right). In the DCU encoder, blocks are formed at multi-granular levels. A block embedding is learned for each granularity. The composition gates for each word are constructed by modeling the relationships between all NBOW (neural bag-of-words) blocks that it resides in.

Dilated Composition Mechanism The inputs to the DCU encoder are (1) a sequence $\{w_1, w_2 \dots w_\ell\}$, and (2) list of ranges $\{r_1, r_2 \dots r_k\}$ where k is the number of times the fold/unfold operation is executed. The final output of the encoder is a sequence of vectors which retains the same dimensionality as its inputs.

Fold Operation This section describes the operation for each r_j . For each r_j and the input sequence, the fold operation performs the summation of every r_j

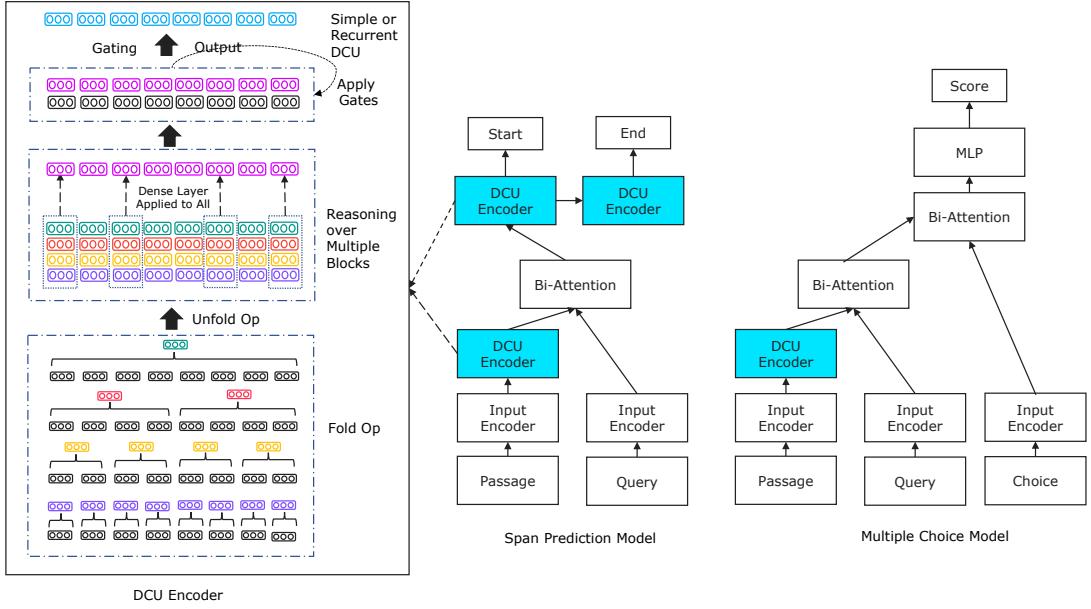


FIGURE 8.1: Architecture of DCU.

word. This is essentially the NBOW (neural bag-of-words) representation.

$$w_t = \sum_{n=a}^{a+b} x_n \quad (8.1)$$

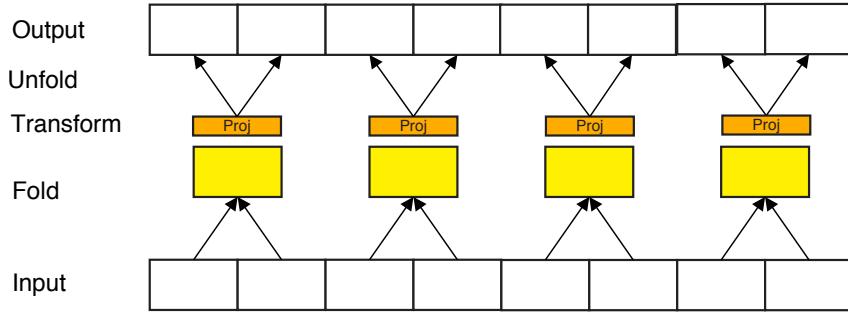
where $a = t \bmod r_j$ and $b = r_j - (t \bmod r_j)$. This reduces the overall document length to ℓ/r_j where each item in the sequence is the sum of every r_j word. Given the new sequence of ℓ/r_j tokens, we then pass each token into a single layered feed-forward neural network:

$$\bar{w}_t = \sigma_r(\mathbf{W}_a(w_t)) + \mathbf{b}_a \quad (8.2)$$

where $\mathbf{W}_a \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_a \in \mathbb{R}^d$ are the parameters of the fold layer. σ_r is the ReLU activation function. w_t is the t -th token in the sequence.

Unfold Operation Given the transformed tokens $\bar{w}_1, \bar{w}_2, \dots, \bar{w}_{\ell/r_j}$, we then expand/unfold them into the original sequence length. Note that for each r_j , the parameters $\mathbf{W}_a, \mathbf{b}_a$ are not shared between blocks. The unfold operator can be described as:

$$\bar{w}_t = [w_t; w_{t+1}; \dots; w_{t+r_j-1}] \quad (8.3)$$

FIGURE 8.2: Overview of the Fold-Unfold operation for $r_j = 2$.

where $[;]$ is the concat operator.

Figure 8.2 depicts the fold/unfold operation for a single value of r_j . Overall, the key intuition of *each* fold-unfold operation is to learn representations of a block of a single granularity (say, blocks of 2). The main rationale for unfolding is to enable reasoning over multiple blocks (or granularities). This is described in the next section.

Multi-Granular Reasoning From k different calls of the fold/unfold operation at different block sizes, we pass the concatenated vector of all transformed tokens into a two layered feed-forward neural network.

$$g_t = \mathbf{F}_2(\mathbf{F}_1([w_t^1; w_t^2; \dots; w_t^k])) \quad (8.4)$$

where $\mathbf{F}_1(\cdot), \mathbf{F}_2(\cdot)$ are feed-forward networks with ReLU activations, i.e., $\sigma_r(W_x + b)$. $[;]$ is the concatenation operator. g_t is interpreted as a gating vector learned from multiple granularities and Equation (8.4) is learning the relationships between a token's representation at multiple hierarchies depending on the values of r_j . Notably, it is easy to see that every n pairs of words will have the same gating vector where n is the lowest value of r_j . As such, the value of the unigram, i.e., $r_j = 1$ (projection of every single token) is critical as it prevents identical gating vectors across the sequence.

8.2.1.1 Encoding Operation

To learn the DCU encoded representation of each word, we consider two variations of DCU encoders.

Simple Encoding In this variation, we use g_t as a gating vector to control the fine-grained balance between the projection of each word w_t in the original input document and the original representation.

$$z_t = \tanh(\mathbf{W}_p w_t) + \mathbf{b}_p \quad (8.5)$$

$$y_t = \sigma(g_t) * w_t + (1 - \sigma(g_t)) z_t \quad (8.6)$$

where $\{y_1, y_2, \dots, y_\ell\}$ is the output document representation. σ is the sigmoid function. Note that this formulation is in similar spirit to highway networks [146]. However, since our gating function is learned via reasoning over multi-granular sequence blocks, it captures more compositionality and long-range context. Note that an optional and additional projection may be applied to w_t but we found that it did not yield much empirical benefit.

Recurrent Encoding (DCU cell) In the second variation, we consider a recurrent (sequential) variant. This is in similar spirit to QRNNs [204] and SRUs [205] which reduces computation cost by pre-learning the gating vectors. The following operations describe the operations of the recurrent DCU cell for each time step t .

$$c_t = g_t \odot c_{t-1} + (1 - g_t) \odot z_t \quad (8.7)$$

$$h_t = o_t \odot c_t \quad (8.8)$$

where c_t, h_t are the cell and hidden states at time step t . g_t are the gates learned from the output of the multi-range reasoning step. o_t is an additional output gate learned via applying an affine transform on the input vector w_t , i.e., $o_t = W_o(w_t) + b_o$. Similar to RNNs, the Recurrent DCU parses the input sequence word-by-word. However, the cost is significantly reduced because we do not have expensive matrix operations that are executed in a non-parallel fashion. Finally, the outputs of the DCU encoder are a series of hidden vectors $\{h_1, h_2, \dots, h_\ell\}$ for each word in the sequence.

8.2.2 Overall Model Architectures

This section describes the overall model architectures that utilize DCU encoders. In our experiments, we focus on both multiple-choice based (RACE) and span prediction NLU/MRC tasks (SearchQA, NarrativeQA). Since the core focus of this work is our encoder, we briefly provide the high-level details of our vanilla Bi-Attentive model. The Bi-Attentive models that are used in our experiments act as baselines, often being less complex than current competitive models such as BiDAF [110], AMANDA [117] or DFN [210]. Figure 8.1 (center and right side of the figure) provides a high-level illustration of these architectures.

8.2.2.1 Multiple Choice Model

In the Multiple Choice (MCQ) Model, there are three types of input sequences, namely Passage (P), Question (Q) and Answers (A_j). The output of the model (for each answer), is a score $s(P, Q, A_j) \in [0, 1]$ denoting the strength of A_j .

Input Encoding Each input sequence is passed into first a projection layer. To enhance the input word representations, we also include the standard EM (exact match) binary feature to each word. In this case, we use a three-way EM adaptation, i.e., $EM(P, Q)$, $EM(Q, A)$ and $EM(P, A)$. The projected embeddings are then passed into a single-layered highway network.

Compositional Encoder In our experiments, we vary the encoder in this layer. Typical choices of encoders in this layer are LSTMs or GRUs. we vary this in our experiments in order to benchmark the effectiveness of our proposed DCU encoder. The output of this layer has the same dimensions as its inputs (typically the hidden states of a RNN model).

Bi-Attention Layer This layer models the interactions between P, Q and A . Let $B(\cdot)$ be a standard bidirectional attention that utilizes mean-pooling aggregation. The scoring function is the bilinear product of the nonlinearly transformed input, i.e., $F(x)_i^\top MF(y)_i$. We first apply $B(P, Q)$ to form bi-attentive P^q, Q^p representations. Subsequently, we apply $B(P^q, A_j)$ to learn a vector representation

for each answer. A temporal sum pooling is applied on the outputs of P^{qa} , A_j^p and concatenated to form $a_j^f \in \mathbb{R}^{2d}$.

Answer Selection Let $\{a_1, a_2 \dots a_{N_a}\}$ be the inputs to this layer and N_a is the number of answer candidates. Motivated by the work in retrieval-based QA [3], we include word overlap features to each answer candidate. This word overlap feature is in similar spirit to the EM feature. Each overlap operation between two sequences returns four features. We convert each answer vector a_j into a scalar via $a_j^f = \text{Softmax}(\mathbf{W}_2(\sigma_r(\mathbf{W}_1([a_j]) + b_1) + b_2))$ where \mathbf{W}_*, b_* and $* = \{1, 2\}$ are standard dense layer parameters.

Optimization The MCQ-based model minimizes the multi-class cross entropy where the number of classes corresponds to the number of choices.

8.2.2.2 Span Prediction Model

In this model, the goal is to extract or predict a span s (start), e (end) where $P[s : e]$ is the answer to the query. As such, the key interaction in this architecture is between P and Q . For most part, the model architecture remains similar especially for the input encoding layer and compositional encoder layer. The key difference is that we reduce the number of input sequence from three to two.

Input Encoding This follows the same design as the MCQ-based model, albeit for two sequences instead of three. Similarly, the two-way EM feature is added before passing into the highway layer.

Compositional Encoder This layer remains identical to the MCQ-based model.

Bi-Attention Layer We adopt a different bi-attention function for span prediction. More specifically, we use the ‘SubMultNN’ or the ‘Mult’ adaptation from [79] (which is tuned) and compare aligned sequences between P and Q to form P^q , the query-dependent passage representation.

Answer Pointer Layer In this layer, we pass P^q through a two layered compositional encoder (which is varied similar to the compositional encoder layer and will be further elaborated in our experiments). The start pointer and end pointer are determined by $F(H_1), F(H_2)$ where H_1, H_2 are the hidden outputs from the first and second encoders respectively. $F(\cdot)$ is a linear transform, projecting each hidden state to a scalar. We pass both of them into softmax functions to obtain probability distributions.

Optimization Following [34, 110], we minimize the joint cross entropy loss of the start and end probability distributions. During inference, we follow [34] to find the best answer span.

8.3 Experiments

In this section, we report our experimental results and comparisons against other published works.

8.3.1 Datasets

For our experiments, we use one challenging multiple choice MRC dataset and two span-prediction MRC datasets.

RACE (Reading Comprehension from Examinations) [43] This is a recently proposed dataset that is constructed from real-world examinations. Given a passage, there are several questions with four options each. The authors argue that RACE is more challenging compared to popular benchmarks (e.g., SQuAD [1]) as more multi-sentence and compositional reasoning are required. There are two subsets of RACE, namely RACE-M (Middle school) and RACE-H (High school). The latter is considered to be harder than the former.

SearchQA [48] This dataset was discussed earlier in Chapter 5, which is a recent dataset that emulates a real-world QA system. It involves extracting passages from

search engine results and requiring models to answer questions by reasoning and reading these search snippets.

NarrativeQA [64] This dataset was discussed earlier in Chapter 5, which is a recent benchmark proposed for story-based reading comprehension. Different from many NLU/RC datasets, the answers are handwritten by human annotators. We focus on the *summaries* setting instead of reading full stories since our model is targetted at standard NLU/RC tasks.

MCQ datasets are evaluated using the standard accuracy metric. For RACE, we train models on the entire dataset, i.e., both RACE-M and RACE-H, and evaluate them separately. Moreover, the model selection is also based on each subset’s respective development set. For SearchQA, we follow [48, 117] which evaluates unigram exact match (EM) and n -gram F1 scores. For NarrativeQA, since the answers are human written and not constrained to a span that can be found in the passage, the evaluation metrics are BLEU-1, BLEU-4, Meteor and Rouge-L following [64].

8.3.2 Compared Methods

We describe the key competitors on each dataset.

RACE The key competitors are the Stanford Attention Reader (Stanford AR) [211], Gated Attention Reader (GA) [212], and Dynamic Fusion Networks (DFN) [210]. GA incorporates a multi-hop attention mechanism that helps to refine the answer representations. DFN is an extremely complex model. It uses (1) BiMPM’s matching functions [80] for extensive matching between Q , P and A , (2) multi-hop reasoning powered by ReasoNet [25] and (3) reinforcement learning techniques for dynamic strategy selection. A leaderboard for this dataset is maintained at http://www.qizhexie.com/data/RACE_leaderboard.

SearchQA The main competitor baseline is the AMANDA model proposed in [117]. AMANDA uses a multi-factor self-attention module, along with a question focused span prediction. AMANDA also uses BiLSTM layers for input encoding

and at the span prediction layers. We also compare against the reported ASR [177] baselines which were reported in [48].

NarrativeQA On this benchmark, we compare with the reported baselines in [64]. We compete on the summaries setting, in which the baselines are a context-less sequence to sequence (seq2seq) model, namely ASR [177] and BiDAF [110]. We also benchmark a stronger competitor, namely R-NET [113] on this benchmark. We use the open source implementation at <https://github.com/HKUST-KnowComp/R-Net>.

Our Methods Across our experiments, we benchmark several variants of our proposed DCU. The first is denoted as Sim-DCU which corresponds to the Simple DCU model described earlier. The model denoted by DCU (without any prefix) corresponds to the recurrent DCU model. Finally, the final variant is the DCU-LSTM which places a DCU encoder layer on top of a BiLSTM layer. We report the dimensions of the encoder as well as training time (per epoch) for each variant. The encompassing framework for DCU is the Bi-Attentive models described for MCQ-based problems and span prediction problems. Unless stated otherwise, the encoder in the pointer layer for span prediction models also uses DCU. However, for the Hybrid DCU-LSTM models, answer pointer layers use BiLSTMs. For the RACE dataset, we additionally report scores of an ensemble of nine Sim-DCU models. This is to facilitate comparison against ensemble models of [210]. We tune the dimensionality of the DCU cell within a range of $100 - 300$ in denominations of 50. The results reported are the best performing models on the held-out set.

8.3.3 Implementation Details

We implement all models in TensorFlow [152]. Word embeddings are initialized with $300d$ GloVe [16] vectors and are not fine-tuned during training. Dropout rate is tuned amongst $\{0.1, 0.2, 0.3\}$ on all layers including the embedding layer. For our DCU model, we use range values of $\{1, 2, 4, 10, 25\}$. DCU encoders are only applied on the passage and not the query. We adopt the Adam optimizer [153] with a learning rate of $0.0003/0.001/0.001$ for RACE/SearchQA/NarrativeQA respectively. The batch size is set to $64/256/32$ accordingly. The maximum sequence

TABLE 8.1: Results on RACE dataset.

Model	RACE-M	RACE-H	RACE	Time
Sliding Window [43]	37.3	30.4	32.2	N/A
Stanford AR [211]	44.2	43.0	43.3	N/A
GA [212]	43.7	44.2	44.1	N/A
ElimiNet [213]	N/A	N/A	44.5	N/A
Dynamic Fusion Network [210]	51.5	45.7	47.4	≈8 hours (1 week*)
Bi-Attention (No Encoder)	50.6	44.0	44.9	3 min (9 hours)
Bi-Attention (250d GRU)	48.5	42.1	44.0	16 min (2 days)
Bi-Attention (250d LSTM)	50.3	40.9	43.6	18 min (2 days)
Bi-Attention (250d Sim-DCU)	57.7	<u>47.4</u>	50.4	4 min (12 hours)
Bi-Attention (250d DCU)	<u>56.1</u>	47.5	<u>50.0</u>	12 min (20 hours)
GA + ElimiNet [213]	N/A	N/A	47.2	N/A
DFN [†] (x9) [210]	55.6	49.4	51.2	N/A
Bi-Attention (S-DCU) [†] (x9)	60.2	50.3	53.3	N/A

lengths are 500/200/1100 respectively. For NarrativeQA, we use the Rouge-L score to find the best approximate answer relative to the human written answer for training the span model. All models are trained and all runtime benchmarks are based on a TitanXP GPU.

8.3.4 Experimental Results on RACE

Table 8.1 reports our results on the RACE benchmark dataset. Competitor results are reported from [43, 210]. The best result for each category (single and ensemble) is in boldface. The last column reports estimated training time per epoch and total time for convergence. * is an estimated value that we obtain from the authors.

Our proposed DCU model achieves the best result for both single models and ensemble models. We outperform highly complex models such as DFN. We also pull ahead of other recent baselines such as ElimiNet [213] and GA by at least 5%. The best single model score from RACE-H and RACE-M alternates between Sim-DCU and DCU. Overall, there is a 6% improvement on the RACE-H dataset and 1.8% improvement on the RACE-M dataset. Our Sim-DCU model also runs at 4 minutes per iteration, which is dramatically faster and simpler than DFN or other recurrent models. We believe that this finding highlights the importance of designing strong and fast baselines for the task at hand.

TABLE 8.2: Results on SearchQA dataset.

Model	Dev		Test		Time
	Acc	F1	Acc	F1	
TF-IDF max [48]	13.0	N/A	12.7	N/A	N/A
ASR [177]	43.9	24.2	41.3	22.8	N/A
AMANDA [117]	48.6	57.7	46.8	56.6	$\approx 8^*$ min
Bi-Attention [†] (No Encoder)	12.4	20.2	18.9	12.3	≈ 17 sec
Bi-Attention [†] (150d BiLSTM)	40.0	51.3	38.6	49.0	≈ 7 min
Bi-Attention [†] (300d LSTM)	40.3	48.7	38.2	46.4	≈ 6 min
s height Bi-Attention [†] (300d Sim-DCU)	44.1	45.5	42.9	43.1	≈ 25 sec
Bi-Attention [†] (300d DCU)	48.6	54.8	46.8	53.3	≈ 2 min
Bi-Attention (200d Hybrid DCU-LSTM)	50.5	59.9	49.4	59.5	≈ 7 min

In general, we also found that the usage of a recurrent cell is not really crucial on this dataset since (1) Sim-DCU and DCU can achieve comparable performance to each other, (2) GRU and LSTM models do not have a competitive edge and (3) using no encoder can achieve comparable performance to DFN. Finally, an ensemble of Sim-DCU models achieve state-of-the-art performance on the RACE dataset, achieving an overall score of 53.3%.

8.3.5 Experimental Results on SearchQA

Table 8.2 reports our results on the SearchQA dataset. Unigram Accuracy and n -gram F1 are reported following [117]. All models with [†] use the same encoder in the answer pointer layer. * is an estimate running a replicated model with the same batch size ($b = 256$) as our models. We draw the reader’s attention to the performance of the 300d DCU encoder. We achieve the same accuracy as AMANDA without using any LSTM or GRU encoder. This model runs at 2 minutes per epoch, making it 4 times more efficient than AMANDA (estimated, with identical batch size). While AMANDA also uses multi-factor self-attention, along with character enhanced representations, our simple DCU encoder used within a mere baseline bi-attentive framework comes close in performance. Finally, the hybrid combination, DCU-LSTM significantly outperforms AMANDA by 3%.

Contrary to MCQ-based datasets, we found that the Sim-DCU model could not achieve comparable results to the recurrent DCU. We hypothesize that this is due to the need to predict spans. Nevertheless, the 300d DCU outperforms an LSTM encoder and remains competitive to a BiLSTM of similar dimensionality. In terms

TABLE 8.3: Results on NarrativeQA dataset.

Model	BLEU-1	BLEU-4	Meteor	Rouge-L	Time
Seq2Seq [†]	15.89	1.26	4.08	13.15	N/A
ASR [†] [177]	23.20	6.39	7.77	22.26	N/A
BiDAF [†] [110]	33.72	15.53	15.38	36.30	N/A
R-NET ^φ [113]	34.90	20.30	18.00	36.70	N/A
Bi-Attention (300d LSTM)	31.18	15.34	14.42	32.95	≈1 hr
Bi-Attention (150d BiLSTM)	34.22	18.22	16.19	38.32	≈1 hr
Bi-Attention (300d Sim-DCU)	9.15	1.69	3.95	11.16	1 min
Bi-Attention (300d DCU)	33.28	16.15	15.84	36.65	18 min
Bi-Attention (150d DCU-LSTM)	36.55	<u>19.79</u>	<u>17.87</u>	41.44	≈1 hr

of representation and parameter size, we consider a 150d BiLSTM to be equivalent to a 300d LSTM for a fair comparison. We also observe that LSTM and DCU are complementary. This shows that stacking a DCU encoder over standard LSTMs can give a performance boost relative to using each encoder separately.

8.3.6 Experimental Results on NarrativeQA

Table 8.3 reports our results on the NarrativeQA benchmark [64]. The baselines marked with [†] are baselines reported by [64] using the summaries setting. The baseline marked with ^φ was obtained by running an open-source implementation of R-NET on the benchmark.

First, we observe that 300d DCU can achieve comparable performance with BiDAF [110]. When compared with a BiLSTM of equal output dimensions (150d), we find that our DCU model performs competitively, with less than 1% deprovement across all metrics. However, the time cost required is significantly reduced. The performance of our model is significantly better than 300d LSTM model while also being significantly faster. Here, we note that Sim-DCU does not produce reasonable results at all, which seems to be in a similar vein to results on SearchQA, i.e., a recursive cell that processes word-by-word is mandatory for span prediction. However, our results show that it is not necessary to construct gates in a word-by-word fashion. Finally, DCU-LSTM significantly outperforms all models in terms of ROUGE-L, including BiDAF on this dataset. Performance improvement over the vanilla BiLSTM model ranges from 1% – 3% across all metrics, suggesting that DCU encoders are also effective as a complementary neural building block.

8.4 Summary

We proposed a novel neural architecture, the DCU encoder and an overall bi-attentive model for both MCQ-based and span prediction MRC tasks. We apply it to three MRC datasets and achieve competitive performance on all without the use of recurrent and convolution layers. Our proposed method outperforms DFN, an extremely complex model, without using any LSTM/GRU layer. We also remain competitive to AMANDA and BiDAF without any LSTM/GRU. While our proposed encoder demonstrates promise on reasoning and understanding natural language, we believe that our encoder is generalizable to other domains beyond reading comprehension. However, we defer this prospect to future work. The source code can be found at https://github.com/vanzytay/EMNLP18_DCU.

Chapter 9

Recurrently Controlled Recurrent Networks for Natural Language Understanding

Recurrent neural networks (RNNs) live at the heart of many sequence modeling problems. Designing powerful inductive biases for modeling sequences lies at the core of language processing research. RNNs are ubiquitous in NLU applications, which warrants investigation along this line. This chapter¹ presents a new recurrent architecture that demonstrates highly effective performance on a wide spectrum of NLP and NLU tasks.

9.1 Introduction

Within the core of RNNs, the incorporation of gated additive recurrent connections is extremely powerful, leading to the pervasive adoption of models such as Gated Recurrent Units (GRU) [19] or Long Short-Term Memory (LSTM) [18] across many NLP/NLU applications [22, 23, 85, 111]. In these models, the key idea is that the gating functions control information flow and compositionality over time, deciding how much information to read/write across time steps. This not only serves as a

¹This chapter is published as *Recurrently Controlled Recurrent Networks*, Proceedings of NeurIPS 2018 [68].

protection against vanishing/exploding gradients but also enables greater relative ease in modeling long-range dependencies.

There are two common ways to increase the representation capability of RNNs. Firstly, the number of hidden dimensions could be increased. Secondly, recurrent layers could be stacked on top of each other in a hierarchical fashion [214], with each layer’s input being the output of the previous, enabling hierarchical features to be captured. Notably, the wide adoption of stacked architectures across many applications [113, 154, 215, 216] signify the need for designing complex and expressive encoders. Unfortunately, these strategies may face limitations. For example, the former might run a risk of overfitting and/or hitting a wall in performance. On the other hand, the latter might be faced with the inherent difficulties of going deep such as vanishing gradients or difficulty in feature propagation across deep RNN layers [217].

In recent years, many RNN variants have been proposed, ranging from multi-scale models [133, 218, 219] to tree-structured encoders [83, 220]. Models that are targetted at improving the internals of the RNN cell have also been proposed [132, 221]. Given the importance of sequence encoding in NLP, the design of effective RNN units for this purpose remains an active area of research.

Stacking RNN layers is the most common way to improve representation power. This has been used in many highly performant models ranging from speech recognition [215] to machine reading [113]. The BCN model [85] similarly uses multiple BiLSTM layers within their architecture. Models that use shortcut/residual connections in conjunction with stacked RNN layers are also notable [154, 173, 217, 222].

Notably, a recently emerging trend is to model sequences without recurrence. This is primarily motivated by the fact that recurrence is an inherent prohibitor of parallelism. To this end, many works have explored the possibility of using attention as a replacement for recurrence. In particular, self-attention [24] has been a popular choice. This has sparked many innovations, including general purpose encoders such as DiSAN [223] and Block Bi-DiSAN [224]. The key idea in these works is to use multi-headed self-attention and positional encodings to model temporal information.

While attention-only models may come close in performance, some domains may still require the complex and expressive recurrent encoders. Moreover, we note that in [223, 224], the scores on multiple benchmarks (e.g., SST, TREC, SNLI, MultiNLI) do not outperform (or even approach) the state-of-the-art, most of which are models that still heavily rely on bidirectional LSTMs [83, 85, 154, 225]. While self-attentive RNN-less encoders have recently been popular, our work moves in an orthogonal and possibly complementary direction, advocating a stronger RNN unit for sequence encoding instead. Nevertheless, it is also good to note that our RCRN model outperforms DiSAN in all our experiments.

Another line of work is also concerned with eliminating recurrence. SRUs (Simple Recurrent Units) [205] are recently proposed networks that remove the sequential dependencies in RNNs. SRUs can be considered as a special case of Quasi-RNNs [204], which performs incremental pooling using pre-learned convolutional gates. Zhang et al. [226] proposed sentence-state LSTMs (S-LSTM) that exchanges incremental reading for a single global state.

This chapter proposes Recurrently Controlled Recurrent Networks (RCRN), a new recurrent architecture and a general purpose neural building block for sequence modeling. RCRN proposes a new way of enhancing the representation capability of RNNs without going deep. For the first time, we propose a controller-listener architecture that uses one recurrent unit to control another recurrent unit. RCRNs are characterized by its usage of two key components - a recurrent controller cell and a listener cell. The *controller* cell controls the information flow and compositionality of the *listener* RNN. The key motivation behind RCRN is to provide expressive and powerful sequence encoding. However, unlike stacked architectures, all RNN layers operate jointly on the same hierarchical level, effectively avoiding the need to go deeper. Therefore, RCRNs provide a new alternative way of utilizing multiple RNN layers in conjunction by allowing one RNN to control another RNN. As such, our key aim in this work is to show that our proposed controller-listener architecture is a viable replacement for the widely adopted stacked recurrent architecture.

To demonstrate the effectiveness of our proposed RCRN model, we conduct extensive experiments on a plethora of diverse NLP tasks where sequence encoders such as LSTMs/GRUs are highly essential. These tasks include sentiment analysis

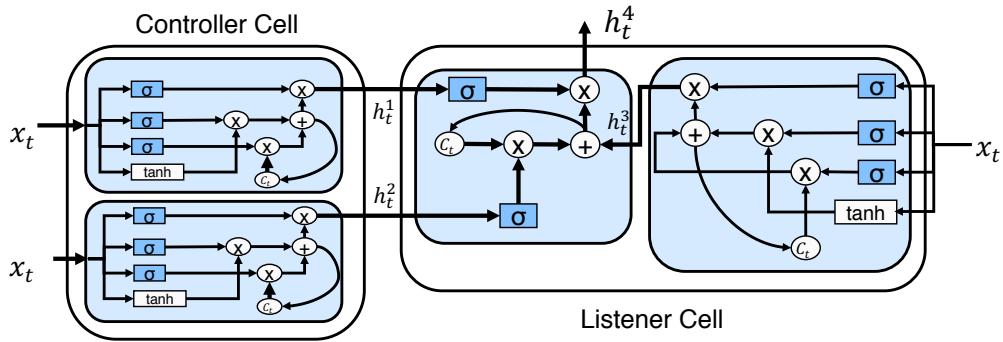


FIGURE 9.1: Architecture of RCRN.

(SST, IMDb, Amazon Reviews), question classification (TREC), entailment classification (SNLI, SciTail), answer retrieval (WikiQA, TrecQA) and machine reading comprehension (NarrativeQA). Experimental results show that RCRN outperforms BiLSTMs and multi-layered/stacked BiLSTMs on all **26** datasets, suggesting that RCRNs are viable replacements for the widely adopted stacked recurrent architectures. Additionally, RCRN achieves close to state-of-the-art performance on several datasets.

9.2 Proposed Method

This section formally introduces the RCRN architecture. Our model is split into two main components - a controller cell and a listener cell. Figure 9.1 illustrates the model architecture.

9.2.1 Controller Cell

The goal of the controller cell is to learn gating functions in order to influence the target cell. In order to control the target cell, the controller cell constructs a forget gate and an output gate which are then used to influence the information flow of the listener cell. For each gate (output and forget), we use a separate RNN cell. As such, the controller cell comprises two cell states and an additional set of

parameters. The equations of the controller cell are defined as follows:

$$i_t^1 = \sigma_s(W_i^1 x_t + U_i^1 h_{t-1}^1 + b_i^1) \text{ and } i_t^2 = \sigma_s(W_i^2 x_t + U_i^2 h_{t-1}^2 + b_i^2) \quad (9.1)$$

$$f_t^1 = \sigma_s(W_f^1 x_t + U_f^1 h_{t-1}^1 + b_f^1) \text{ and } f_t^2 = \sigma_s(W_f^2 x_t + U_f^2 h_{t-1}^2 + b_f^2) \quad (9.2)$$

$$o_t^1 = \sigma_s(W_o^1 x_t + U_o^1 h_{t-1}^1 + b_o^1) \text{ and } o_t^2 = \sigma_s(W_o^2 x_t + U_o^2 h_{t-1}^2 + b_o^2) \quad (9.3)$$

$$c_t^1 = f_t^1 c_{t-1}^1 + i_t^1 \sigma(W_c^1 x_t + U_c^1 h_{t-1}^1 + b_c^1) \quad (9.4)$$

$$c_t^2 = f_t^2 c_{t-1}^2 + i_t^2 \sigma(W_c^2 x_t + U_c^2 h_{t-1}^2 + b_c^2) \quad (9.5)$$

$$h_t^1 = o_t^1 \odot \sigma(c_t^1) \text{ and } h_t^2 = o_t^2 \odot \sigma(c_t^2) \quad (9.6)$$

where x_t is the input to the model at time step t . W_*^k, U_*^k, b_*^k are the parameters of the model where $k = \{1, 2\}$ and $* = \{i, f, o\}$. σ_s is the sigmoid function and σ is the tanh nonlinearity. \odot is the Hadamard product. The controller RNN has two cell states denoted as c^1 and c^2 respectively. h_t^1, h_t^2 are the outputs of the unidirectional controller cell at time step t . Next, we consider a bidirectional adaptation of the controller cell. Let Equation (9.1) to Equation (9.6) be represented by the function $CT()$, the bidirectional adaptation is represented as:

$$\overrightarrow{h}_t^1, \overrightarrow{h}_t^2 = \overrightarrow{CT}(h_{t-1}^1, h_{t-1}^2, x_t) \quad t = 1, \dots, \ell \quad (9.7)$$

$$\overleftarrow{h}_t^1, \overleftarrow{h}_t^2 = \overleftarrow{CT}(h_{t+1}^1, h_{t+1}^2, x_t) \quad t = M, \dots, 1 \quad (9.8)$$

$$h_t^1 = [\overrightarrow{h}_t^1; \overleftarrow{h}_t^1] \text{ and } h_t^2 = [\overrightarrow{h}_t^2; \overleftarrow{h}_t^2] \quad (9.9)$$

The outputs of the bidirectional controller cell are h_t^1, h_t^2 for time step t . These hidden outputs act as gates for the listener cell.

9.2.2 Listener Cell

The listener cell is another recurrent cell. The final output of the RCRN is generated by the listener cell which is being influenced by the controller cell. First, the listener cell uses a base recurrent model to process the sequence input. The

equations of this base recurrent model are defined as follows:

$$i_t^3 = \sigma_s(W_i^3 x_t + U_i^3 h_{t-1}^3 + b_i^3) \quad (9.10)$$

$$f_t^3 = \sigma_s(W_f^3 x_t + U_f^3 h_{t-1}^3 + b_f^3) \quad (9.11)$$

$$o_t^3 = \sigma_s(W_o^3 x_t + U_o^3 h_{t-1}^3 + b_o^3) \quad (9.12)$$

$$c_t^3 = f_t^3 c_{t-1}^3 + i_t^3 \sigma(W_c^3 x_t + U_c^3 h_{t-1}^3 + b_c^3) \quad (9.13)$$

$$\overrightarrow{h}_t^3 = o_t^3 \odot \sigma(c_t^3) \quad (9.14)$$

Similarly, a bidirectional adaptation is used, obtaining $h_t^3 = [\overrightarrow{h}_t^3, \overleftarrow{h}_t^3]$. Next, using h_t^1, h_t^2 (outputs of the controller cell), we define another recurrent operation as follows:

$$c_t^4 = \sigma_s(h_t^1) \odot c_{t-1}^4 + (1 - \sigma_s(h_t^1)) \odot h_t^3 \quad (9.15)$$

$$h_t^4 = h_t^2 \odot c_t^3 \quad (9.16)$$

where c_t^j, h_t^j and $j = \{3, 4\}$ are the cell and hidden states at time step t . W_*^3, U_*^3 are the parameters of the listener cell where $* = \{i, f, o\}$. Note that h_t^1 and h_t^2 are the outputs of the controller cell. In this formulation, $\sigma_s(h_t^1)$ acts as the forget gate for the listener cell. Likewise $\sigma_s(h_t^2)$ acts as the output gate for the listener.

9.2.3 Overall RCRN Architecture, Variants and Implementation

Intuitively, the overall architecture of the RCRN model can be explained as follows: Firstly, the controller cell can be thought of as two BiRNN models which hidden states are used as the forget and output gates for another recurrent model, i.e., the listener. The listener uses a single BiRNN model for sequence encoding and then allows this representation to be altered by listening to the controller. An alternative interpretation to our model architecture is that it is essentially a ‘recurrent-over-recurrent’ model. Clearly, the formulation we have used above uses BiLSTMs as the atomic building block for RCRN. Hence, we note that it is also possible to have a simplified variant of RCRN that uses GRUs as the atomic block which we found to have performed slightly better on certain datasets.

Cuda-level Optimization For efficiency purposes, we use the cuDNN optimized version of the base recurrent unit (LSTMs/GRUs). Additionally, note that the final recurrent cell (Equation (9.15)) can be subject to CUDA-level optimization following simple recurrent units (SRU) [205]. The key idea is that this operation can be performed along the dimension axis, enabling greater parallelization on the GPU [205]. Note that this form of cuda-level optimization was also performed in the Quasi-RNN model [204], which effectively subsumes the SRU model. We adapt the CUDA kernel as a custom Tensorflow op in our experiments. While the authors of SRU release their cuda-op at <https://github.com/taolei87/sru>, we use a third-party open-source Tensorflow version which can be found at https://github.com/JonathanRaiman/tensorflow_qrnn.git.

On Parameter Cost and Memory Efficiency Note that a single RCRN model is equivalent to a stacked BiLSTM of 3 layers. This is clear when we consider how two controller BiRNNs are used to control a single listener BiRNN. As such, for our experiments, when considering only the encoder and keeping all other components constant, 3L-BiLSTM has equal parameters to RCRN while RCRN and 3L-BiLSTM are approximately three times larger than BiLSTM.

9.3 Experiments

This section discusses the overall empirical evaluation of our proposed RCRN model.

9.3.1 Tasks and Datasets

In order to verify the effectiveness of our proposed RCRN architecture, we conduct extensive experiments across several tasks in the NLP/NLU domain.

Sentiment Analysis Sentiment analysis is a text classification problem in which the goal is to determine the polarity of a given sentence/document. We conduct experiments on both sentence and document levels. More concretely, we use 16

Amazon review datasets from [227], the well-established Stanford Sentiment Tree-Bank (SST-5/SST-2) [228] and the IMDb Sentiment dataset [229]. All tasks are binary classification tasks with the exception of SST-5. The metric is the accuracy score.

Question Classification The goal of this task is to classify questions into fine-grained categories such as *number* or *location*. We use the TREC question classification dataset [230]. The metric is the accuracy score.

Entailment Classification This is a well-established and popular task in the field of natural language understanding and inference. Given two sentences s_1 and s_2 , the goal is to determine if s_2 entails or contradicts s_1 . We use two popular benchmark datasets, i.e., the Stanford Natural Language Inference (SNLI) corpus [45], and SciTail (Science Entailment) [84] datasets (discussed in Chapter 3). This is a pairwise classification problem in which the metric is also the accuracy score.

Answer Retrieval This is a standard problem in information retrieval and learning-to-rank. Given a question, the task at hand is to rank candidate answers. We use the popular WikiQA [231] and TrecQA [232] (discussed in Chapter 4) datasets. For TrecQA, we use the cleaned setting as denoted by [167]. The evaluation metrics are the MAP (Mean Average Precision) and Mean Reciprocal Rank (MRR) ranking metrics.

Machine Reading Comprehension This task involves reading documents and answering questions about these documents. We use the recent NarrativeQA [64] dataset (discussed in Chapter 5) which involves reasoning and answering questions over story summaries. We follow the original paper and reported scores on BLEU-1, BLEU-4, Meteor, and Rouge-L.

9.3.2 Task-Specific Model Architectures and Implementation Details

In this section, we describe the task-specific model architectures for each task.

Classification Model This architecture is used for all text classification tasks (sentiment analysis and question classification datasets). We use 300D GloVe [16] vectors with 600D CoVe [85] vectors as pretrained embedding vectors. An optional character-level word representation is also added (constructed with a standard BiGRU model). The output of the embedding layer is passed into the RCRN model directly without using any projection layer. Word embeddings are not updated during training. Given the hidden output states of the $200d$ dimensional RCRN cell, we take the concatenation of the max, mean and min pooling of all hidden states to form the final feature vector. This feature vector is passed into a single dense layer with ReLU activations of $200d$ dimensions. The output of this layer is then passed into a softmax layer for classification. This model optimizes the cross entropy loss. We train this model using Adam optimizer [153] and learning rate is tuned amongst $\{0.001, 0.0003, 0.0004\}$.

Entailment Model This architecture is used for entailment tasks. This is a pairwise classification models with two input sequences. Similar to the singleton classification model, we utilize the identical input encoder (GloVe, CoVE and character RNN) but include an additional part-of-speech (POS tag) embedding. we pass the input representation into a two layer highway network [146] of 300 hidden dimensions before passing into the RCRN encoder. The feature representation of s_1 and s_2 is the concatenation of the max and mean pooling of the RCRN hidden outputs. To compare s_1 and s_2 , we pass $[s_1, s_2, s_1 \odot s_2, s_1 - s_2]$ into a two layer highway network. This output is then passed into a softmax layer for classification. We train this model using Adam optimizer and learning rate is tuned amongst $\{0.001, 0.0003, 0.0004\}$. We mainly focus on the *encoder-only* setting which does not allow cross sentence attention. This is a commonly tested setting on the SNLI dataset.

Ranking Model This architecture is used for the ranking tasks (i.e., answer retrieval). We use the model architecture from Attentive Pooling BiLSTMs (AP-BiLSTM) [100] as our base and swap the RNN encoder with our RCRN encoder. The dimensionality is set to 200. The similarity scoring function is the cosine similarity and the objective function is the pairwise hinge loss with a margin of 0.1. We use negative sampling of $n = 6$ to train our model. We train our model using Adadelta [181] with a learning rate of 0.2.

TABLE 9.1: Results on Amazon Reviews dataset for sentiment analysis.

Dataset	BiLSTM	2L-BiLSTM	SLSTM	BiLSTM [†]	3L-BiLSTM [†]	RCRN
Camera	87.1	88.1	90.0	87.3	89.7	90.5
Video	84.7	85.2	86.8	87.5	87.8	88.5
Health	85.5	85.9	86.5	85.5	89.0	90.5
Music	78.7	80.5	82.0	83.5	85.7	86.0
Kitchen	82.2	83.8	84.5	81.7	84.5	86.0
DVD	83.7	84.8	85.5	84.0	86.0	86.8
Toys	85.7	85.8	85.3	87.5	90.5	90.8
Baby	84.5	85.5	86.3	85.0	88.5	89.0
Books	82.1	82.8	83.4	86.0	87.2	88.0
IMDB	86.0	86.6	87.2	86.5	88.0	89.8
MR	75.7	76.0	76.2	77.7	77.7	79.0
Apparel	86.1	86.4	85.8	88.0	89.2	90.5
Magazines	92.6	92.9	93.8	93.7	92.5	94.8
Electronics	82.5	82.3	83.3	83.5	87.0	89.0
Sports	84.0	84.8	85.8	85.5	86.5	88.0
Software	86.7	87.0	87.8	88.5	90.3	90.8
Macro Avg	84.3	84.9	85.6	85.7	87.5	88.6

Machine Reading Comprehension Model We use R-NET [113] as the base model. Since R-NET uses three Bidirectional GRU layers as the encoder, we replaced this stacked BiGRU layer with RCRN. For fairness, we use the GRU variant of RCRN instead. The dimensionality of the encoder is set to 75. We train both models using Adam optimizer with a learning rate of 0.001.

For all datasets, we include additional ablative baselines, swapping the RCRN with (1) a standard BiLSTM model and (2) a stacked BiLSTM of 3 layers (3L-BiLSTM). This is to fairly observe the impact of different encoder models based on the same overall model framework.

9.3.3 Performance Results

This section discusses the overall results of our experiments.

Sentiment Analysis On the 16 review datasets (Table 9.1) from [226, 227], our proposed RCRN architecture achieves the highest score on all 16 datasets, outperforming the existing state-of-the-art model - sentence state LSTMs (SLSTM)

TABLE 9.2: Results on SST-5 dataset for sentiment analysis.

Model/Reference	Acc
MVN [233]	51.5
DiSAN [223]	51.7
DMN [234]	52.1
LSTM-CNN [225]	52.4
NTI [156]	53.1
BCN [85]	53.7
BCN + ELMo [17]	54.7
BiLSTM	51.3
3L-BiLSTM	52.6
RCRN	54.3

TABLE 9.3: Results on SST-2 dataset for sentiment analysis.

Model/Reference	Acc
P-LSTM [235]	89.2
CT-LSTM [236]	89.4
TE-LSTM [237]	89.6
NSE [238]	89.7
BCN [85]	90.3
BMLSTM [239]	91.8
BiLSTM	89.7
3L-BiLSTM	90.0
RCRN	90.6

TABLE 9.4: Results on IMDb dataset for sentiment analysis.

Model/Reference	Acc
Res. BiLSTM [222]	90.1
4L-QRNN [204]	91.4
BCN [85]	91.8
oh-LSTM [240]	91.9
TRNN [241]	93.8
Virtual [242]	94.1
BiLSTM	90.9
3L-BiLSTM	91.8
RCRN	92.8

TABLE 9.5: Results on TREC dataset for question classification.

Model/Reference	Acc
CNN-MC [57]	92.2
SRU [205]	93.9
DSCNN [243]	95.4
DC-BiLSTM [173]	95.6
BCN [85]	95.8
LSTM-CNN [225]	96.1
BiLSTM	95.8
3L BiLSTM	95.4
RCRN	96.2

[226]. The macro average performance gain over BiLSTMs (+4%) and Stacked (2 X BiLSTM) (+3.4%) is also notable. On the same architecture, our RCRN outperforms ablative baselines BiLSTM by +2.9% and 3L-BiLSTM by +1.1% on average across 16 datasets. Note that on Table 9.1, baselines marked with \dagger are models implemented by us.

Results on SST-5 (Table 9.2) and SST-2 (Table 9.3) are also promising. More concretely, our RCRN architecture achieves state-of-the-art results on SST-5 and SST-2. RCRN also outperforms many strong baselines such as DiSAN [223], a self-attentive model and Bi-Attentive classification network (BCN) [85] that also use CoVE vectors. On SST-2, strong baselines such as Neural Semantic Encoders [238] and similarly the BCN model are also outperformed by our RCRN model.

Finally, on the IMDb sentiment classification dataset (Table 9.4), RCRN achieved 92.8% accuracy. Our proposed RCRN outperforms Residual BiLSTMs [222], 4-layered Quasi Recurrent Neural Networks (QRNN) [204] and the BCN model which can be considered to be very competitive baselines. RCRN also outperforms ablative baselines BiLSTM (+1.9%) and 3L-BiLSTM (+1%).

Question Classification Our results on the TREC question classification dataset (Table 9.5) is also promising. RCRN achieved a state-of-the-art score of 96.2% on this dataset. A notable baseline is the Densely Connected BiLSTM [173], a deep residual stacked BiLSTM model which RCRN outperforms (+0.6%). Our model also outperforms BCN (+0.4%) and SRU (+2.3%). Our ablative BiLSTM baselines achieve a reasonably high score, possibly due to CoVE Embeddings [85]. However, our RCRN can further increase the performance score.

TABLE 9.6: Results on SNLI dataset for entailment classification.

Model/Reference	Acc
Multi-head [24]	84.2
Att. Bi-SRU [205]	84.8
DiSAN [223]	85.6
Shortcut [154]	85.7
Gumbel LSTM [83]	86.0
Dynamic Meta Emb [244]	86.7
BiLSTM	85.5
3L-BiLSTM	85.1
RCRN	85.8

TABLE 9.7: Results on SciTail dataset for entailment classification.

Model/Reference	Acc
ESIM [76]	70.6
DecompAtt [21]	72.3
DGEM [84]	77.3
CAFE [174]	83.3
CSRAN [60]	86.7
OpenAI GPT [15]	88.3
BiLSTM	80.1
3L-BiLSTM	79.6
RCRN	81.1

Entailment Classification Results on entailment classification are also optimistic. On SNLI (Table 9.6), RCRN achieves 85.8% accuracy, which is competitive to Gumbel LSTM. However, RCRN outperforms a wide range of baselines, including self-attention based models such as multi-head [24] and DiSAN [223]. There is also a performance gain of +1% over Bi-SRU even though our model does not use attention at all. RCRN also outperforms shortcut stacked encoders, which use a series of BiLSTM connected by shortcut layers. Additionally, we experimented with adding cross sentence attention, in particular adding the attention of [21] on 3L-BiLSTM and RCRN. We found that they performed comparably (both at ≈ 87.0). We did not have resources to experiment further even though intuitively incorporating different/newer variants of attention [60, 76, 245] and/or ELMo [17] can definitely raise the score further. However, we hypothesize that cross sentence attention forces less reliance on the encoder. Therefore, stacked BiLSTMs and RCRNs perform similarly.

The results on SciTail similarly show that RCRN is more effective than BiLSTM

TABLE 9.8: Results on WikiQA and TrecQA datasets for answer retrieval.

Model	WikiQA		TrecQA	
	MAP	MRR	MAP	MRR
BiLSTM	68.5	69.8	72.4	82.5
3L-BiLSTM	69.3	71.3	73.0	83.6
RCRN	71.1	72.3	75.4	85.5
AP-BiLSTM	63.9	69.9	75.1	80.0
AP-3L-BiLSTM	69.8	71.3	73.3	83.4
AP-RCRN	72.4	73.7	77.9	88.2

(+1%). Moreover, RCRN outperforms several baselines in [84] including models that use cross sentence attention such as DecompAtt [21] and ESIM [76]. However, it still falls short to recent state-of-the-art models such as OpenAI’s Generative Pretrained Transformer [15].

Answer Retrieval Results on the answer retrieval (Table 9.8) task show that RCRN leads to considerable improvements on both WikiQA and TrecQA datasets. We investigate two settings. Firstly, we reimplement AP-BiLSTM and swap the BiLSTM for RCRN encoders. Secondly, we completely remove all attention layers from both models to test the ability of the standalone encoder. Without attention, RCRN gives an improvement of $+ \approx 2\%$ on both datasets. With attentive pooling, RCRN maintains a $+ \approx 2\%$ improvement in terms of MAP score. However, the gains on MRR are greater ($+4 - 7\%$). Notably, the AP-RCRN model outperforms the official results reported in [100]. Overall, we observe that RCRN is much stronger than BiLSTMs and 3L-BiLSTMs on this task.

Machine Reading Comprehension Results (Table 9.9) show that enhancing R-NET with RCRN can lead to considerable improvements. This leads to an improvement of $\approx 1\% - 2\%$ on all four metrics. Note that our model only uses a single layered RCRN while R-NET uses 3 layered BiGRUs. This empirical evidence might suggest that RCRN is a better way to utilize multiple recurrent layers.

Overall Results Across all 26 datasets, RCRN outperforms not only standard BiLSTMs but also 3L-BiLSTMs which have approximately equal parameterization. 3L-BiLSTMs are overall better than BiLSTMs but lose out on a minority of datasets. RCRN outperforms a wide range of competitive baselines such as

TABLE 9.9: Results on NarrativeQA dataset for machine reading comprehension

Model	Bleu-1	Bleu-4	Meteor	Rouge
Seq2Seq	16.1	1.40	4.2	13.3
ASR	23.5	5.90	8.0	23.3
BiDAF	33.7	15.5	15.4	36.3
R-NET	34.9	20.3	18.0	36.7
RCRN	38.1	21.8	18.1	38.3

TABLE 9.10: Training/Inference times on IMDb dataset.

	Training Time (sec/epoch)					Inference (sec/epoch)				
	16	32	64	128	256	16	32	64	128	256
3 layer BiLSTM	29	50	113	244	503	12	20	38	72	150
BiLSTM	18	30	63	131	272	9	15	28	52	104
1 layer BiLSTM (cuDNN)	5	6	9	14	26	2	3	4	6	10
3 layer BiLSTM (cuDNN)	10	14	23	42	80	4	5	9	16	32
RCRN (cuDNN)	19	29	53	101	219	8	12	23	41	78
RCRN (cuDNN +CUDA)	10	13	21	40	78	4	5	8	15	29

DiSAN, Bi-SRUs, BCN and LSTM-CNN, etc. We achieve (close to) state-of-the-art performance on SST, TREC question classification and 16 Amazon Review datasets.

9.3.4 Runtime Analysis

This section aims to get a benchmark on model performance with respect to model efficiency. In order to do that, we benchmark RCRN along with BiLSTMs and 3 layered BiLSTMs (with and without cuDNN optimization) on different sequence lengths (i.e., 16, 32, 64, 128, 256). We use the IMDb sentiment task. We use the same standard hardware (a single Nvidia GTX1070 card) and an identical overarching model architecture. The dimensionality of the model is set to 200 with a fixed batch size of 32. Finally, we also benchmark a CUDA optimized adaptation of RCRN which has been described earlier (Section 9.2.3).

Table 9.10 reports training/inference times of all benchmarked models. The fastest model is naturally the 1 layer BiLSTM (cuDNN). Intuitively, the speed of RCRN should be roughly equivalent to using 3 BiLSTMs. Surprisingly, we found that the CUDA optimized RCRN performs consistently slightly faster than the 3 layer BiLSTM (cuDNN). At the very least, RCRN provides comparable efficiency to

using stacked BiLSTM and empirically we show that there is nothing to lose in this aspect. However, we note that CUDA-level optimizations have to be performed. Finally, the non-cuDNN optimized BiLSTM and stacked BiLSTMs are also provided for reference.

9.4 Summary

We proposed Recurrently Controlled Recurrent Networks (RCRN), a new recurrent architecture and encoder for a myriad of NLP and NLU tasks. RCRN operates in a novel controller-listener architecture which uses RNNs to learn the gating functions of another RNN. We apply RCRN to a potpourri of NLP tasks and achieve promising/highly competitive results on all tasks and 26 benchmark datasets. Overall findings suggest that the controller-listener architecture is more effective than stacking RNN layers. Moreover, RCRN remains equally (or slightly more) efficient compared to stacked RNNs of approximately equal parameterization. There are several potential interesting directions for further investigating RCRNs. These include investigating RCRNs controlling other RCRNs and investigating RCRNs in other domains where recurrent models are also prevalent for sequence modeling. The source code of our model can be found at https://github.com/vanzytay/NIPS2018_RCRN.

Chapter 10

Hyperbolic Representations for Natural Language Understanding

Neural ranking models are commonplace in many modern natural language understanding and question answering (QA) systems [3, 163]. In these applications, the problem of question answering is concerned with learning to rank candidate answers in response to questions. While highly performant systems (as introduced in previous chapters) are great, there are occasions which call for model efficiency, i.e., model deployment on mobile devices or requirements for high latency inference. To this end, we seek out novel inductive biases, such as representation learning in non-Euclidean spaces. In this chapter¹, we discuss our proposed HyperQA model.

10.1 Introduction

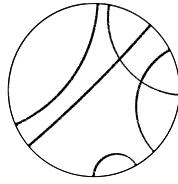
In this chapter, we propose an extremely simple neural ranking based NLU model for question answering that achieves highly competitive results on several benchmarks with only a fraction of the run time and only 40K-90K parameters (as opposed to millions). Our neural ranking models the relationships between QA pairs in hyperbolic space instead of Euclidean space. Hyperbolic space is an embedding space with a constant negative curvature in which the distance towards the border

¹This chapter is published as *Hyperbolic Representation Learning for Fast and Efficient Neural Question Answering*, Proceedings of WSDM 2018 [2].

is increasing exponentially. Intuitively, this makes it suitable for learning embeddings that reflect a natural hierarchy (e.g., networks, text, etc.) which we believe might benefit neural ranking models for QA. Notably, our work is inspired by the recently inceptioned Poincaré embeddings [246] which demonstrates the effectiveness of inducing a structural (hierarchical) bias in the embedding space for improved generalization. In our early empirical experiments, we discovered that a simple feed-forward neural network trained in hyperbolic space is capable of outperforming more sophisticated models on several standard benchmark datasets. We believe that this can be attributed to two reasons. Firstly, latent hierarchies are prominent in QA. Aside from the natural hierarchy of questions and answers, conceptual hierarchies also exist. Secondly, natural language is inherently hierarchical which can be traced to power law distributions and Zipf’s law [247].

In this work, we seek a new paradigm for neural ranking for QA. While many recent works try to *out-stack* each other with new layers, we strip down our network instead. Our work is inspired by the very recent Poincarè embeddings [246] which demonstrates the superiority and efficiency of generalization in hyperbolic space. Moreover, this alleviates many overfitting and complexity issues that Euclidean embeddings might face especially if the data has intrinsic hierarchical structure. It is good to note that power-law distributions, such as Zipf’s law, have been known to be from innate hierarchical structure [247]. Specifically, the defining characteristic of hyperbolic space is much quicker expansion relative to that of Euclidean space which makes it naturally equipped for modeling hierarchical structure. Hyperbolic spaces have been applied to domains such as complex network modeling [248], social networks [249] and geographic routing [250].

There are several key geometric intuitions regarding hyperbolic spaces. Firstly, the concepts of distance and area are *warped* in hyperbolic spaces. Specifically, each tile in Figure 10.1(A) is of equal area in hyperbolic space but diminishes towards zero in Euclidean space towards the boundary. Secondly, hyperbolic spaces are *conformal*, i.e., angles in hyperbolic spaces and Euclidean spaces are identical. In Figure 10.1(B), the arcs on the curve are parallel lines that are orthogonal to the boundary. Finally, hyperbolic spaces can be regarded as *larger* spaces relative to Euclidean spaces due to the fact that the concept of relative distance can be expressed much better, i.e., not only does the distance between two vectors encode



(A) ‘Circle Limit 1’ by M.C Escher (B) Hyperbolic parallel lines

FIGURE 10.1: Illustration of hyperbolic space.

information but also *where* a vector is placed in hyperbolic space. This enables efficient representation learning.

In [246], Nickel et al. applied the hyperbolic distance (specifically, the Poincarè distance) to model taxonomic entities and graph nodes. Notably, our work, to the best of our knowledge, is the only work that learns QA embeddings in hyperbolic space. Moreover, questions and answers introduce an interesting layer of complexity to the problem since QA embeddings are in fact compositions of their constituent word embeddings. On the other hand, nodes in a graph and taxonomic entities in [246] are already at its most abstract form, i.e., symbolic objects. As such, we believe it would be interesting to investigate the impacts of QA in hyperbolic space in lieu of the added compositional nature.

The key contributions in this work are as follows:

- We propose a new neural ranking model for ranking of question answer pairs. For the first time, our proposed model, HYPERQA, performs matching of question and answer in hyperbolic space. To the best of our knowledge, we are the first to model QA pairs in hyperbolic space. While hyperbolic geometry and embeddings have been explored in the domains of complex networks or graphs [248], our work is the first to investigate the suitability of this metric space for question answering.
- HYPERQA is an extremely fast and parameter efficient model that achieves very competitive results on multiple QA benchmarks such as TrecQA, WikiQA and YahooCQA. The efficiency and speed of HYPERQA are attributed by the fact that we do not use any sophisticated neural encoder and have no complicated word interaction layer. In fact, HYPERQA is a mere single layered neural network with only 90K parameters. Very surprisingly, HYPERQA actually outperforms many state-of-the-art models such as Attentive Pooling

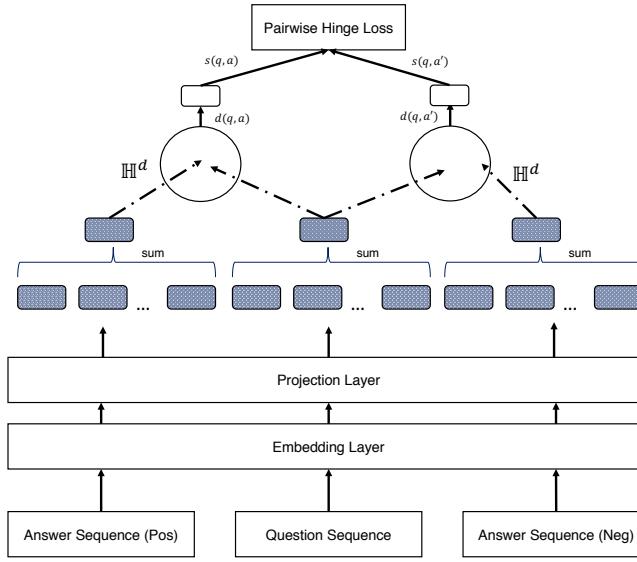


FIGURE 10.2: Architecture of HyperQA.

BiLSTMs [100, 101] and Multi-Perspective CNNs [163]. We believe that this allows us to reconsider if many of these complex word interaction layers are really necessary for good performance.

- We conduct extensive qualitative analysis of both the learned QA embeddings and word embeddings. We discover several interesting properties of QA embeddings in hyperbolic space. Due to its compositional nature, we find that our model learns to self-organize not only at the QA level but also at the word-level. Our qualitative studies enable us to gain a better intuition pertaining to the good performance of our model.

10.2 Proposed Method

This section outlines the overall architecture of our proposed model. Similar to many neural ranking models for QA, our network has ‘two’ sides with shared parameters, i.e., one for question and another for answer. However, since we optimize for a pairwise ranking loss, the model takes in a positive (correct) answer and a negative (wrong) answer, and aims to maximize the margin between the scores of the correct QA pair and the negative QA pair. Figure 10.2 depicts the overall model architecture.

10.2.1 Embedding Layer

Our model accepts three sequences as an input, i.e., the question (denoted as q), the correct answer (denoted as a) and a randomly sampled corrupted answer (denoted as a'). Each sequence consists of M words where M_q and M_a are predefined maximum sequence length for questions and answers respectively. Each word is represented as a one-hot vector (representing a word in the vocabulary). As such, this layer is a look-up layer that converts each word into a low-dimensional vector by indexing onto the word embedding matrix. In our implementation, we initialize this layer with pretrained word embeddings [16]. Note that this layer is not updated during training. Instead, we utilize a projection layer that learns a task-specific projection of the embeddings.

10.2.2 Projection Layer

In order to learn a task-specific representation for each word, we utilize a projection layer. The projection layer is essentially a single layered neural network that is applied to **each word** in all three sequences.

$$x = \sigma(\mathbf{W}_p z + b_p) \quad (10.1)$$

where $\mathbf{W}_p \in \mathbb{R}^{d \times n}$, $z \in \mathbb{R}^n$, $x \in \mathbb{R}^d$ and σ is a non-linear function such as the rectified linear unit (ReLU). The output of this layer is a sequence of d -dimensional embeddings for each sequence (question, positive answer and negative answer). Note that the parameters of this projection layer are shared for both question and answer.

10.2.3 Learning QA Representations

In order to learn question and answer representations, we simply take the sum of all word embeddings in the sequence.

$$y^* = \sum_{i=1}^{M_*} x_i^* \quad (10.2)$$

where $* = \{q, a, a'\}$. M is the predefined max sequence length (specific to question and answer) and $x_1, x_2 \dots x_M$ are d -dimensional embeddings of the sequence. This is essentially the neural bag-of-words (BoW) representation. Unlike popular neural encoders such as LSTM or CNN, the NBOW representation does not add any parameters and is much more efficient. Additionally, we constrain the question and answer embeddings to the unit ball before passing to the next layer, i.e., $\|y^*\| \leq 1$. This is easily done via $y^* = \frac{y^*}{\|y^*\|}$ where $\|y^*\| > 1$. Note that this projection of QA embeddings onto the unit ball is mandatory and absolutely crucial for HYPERQA to even work.

10.2.4 Hyperbolic Representations of QA Pairs

Neural ranking models are mainly characterized by the interaction function between question and answer representations. In our work, we mainly adopt the hyperbolic distance function to model the relationships between questions and answers. While there exist multiple models of hyperbolic geometry such as the Beltrami-Klein model or the Hyperboloid model, we adopt the Poincarè ball/disk due to its ease of differentiability and freedom from constraints [246]. Formally, let $\mathcal{B}^d = \{x \in \mathbb{R}^d \mid \|x\| < 1\}$ be the *open* d -dimensional unit ball, our model corresponds to the Riemannian manifold (\mathcal{B}^d, g_x) and is equipped with the Riemannian metric tensor given as follows:

$$g_x = \left(\frac{2}{1 - \|x\|^2} \right)^2 g^E \quad (10.3)$$

where g^E is the Euclidean metric tensor. The hyperbolic distance function between question and answer is defined as:

$$d(q, a) = \text{arcosh}(1 + 2 \frac{\|q - a\|^2}{(1 - \|q\|^2)(1 - \|a\|^2)}) \quad (10.4)$$

where $\|\cdot\|$ denotes the Euclidean norm and $q, a \in \mathbb{R}^d$ are the question and answer embeddings respectively. Note that *arcosh* is the inverse hyperbolic cosine function, i.e., $\text{arcosh } x = \ln(x + \sqrt{(x^2 - 1)})$. Notably, $d(q, a)$ changes smoothly with respect to the position of q and a which enables the automatic discovery of latent hierarchies. As mentioned earlier, the distance increases exponentially as the norm of the vectors approaches 1. As such, the latent hierarchies of QA embeddings are

captured through the norm of the vectors. From a visual perspective, the origin can be seen as the root of a tree that branches out towards the boundaries of the hyperbolic ball. This self-organizing ability of the hyperbolic distance is visually and qualitatively analyzed in later sections.

10.2.5 Final Transform

Finally, we pass the hyperbolic distance through a linear transformation described as follows:

$$s(q, a) = w_f d(q, a) + b_f \quad (10.5)$$

where $w_f \in \mathbb{R}^1$ and $b_f \in \mathbb{R}^1$ are scalar parameters of this layer. The performance of this layer is empirically motivated and was selected amongst other variants such as $\exp(-d(q, a))$, non-linear activations such as sigmoid function or the raw hyperbolic distance.

10.2.6 Optimization and Learning

This section describes the optimization and learning process of HYPERQA. Our model learns via a pairwise ranking loss, which is well suited for metric-based learning algorithms.

10.2.6.1 Pairwise Hinge Loss

Our network minimizes the pairwise hinge loss which is defined as follows:

$$L = \sum_{(q,a) \in \Delta_q} \sum_{(q,a') \notin \Delta_q} \max(0, s(q, a) + \lambda - s(q, a')) \quad (10.6)$$

where Δ_q is the set of all QA pairs for question q , $s(q, a)$ is the score between q and a , and λ is the margin which controls the extent of discrimination between positive QA pairs and corrupt QA pairs. The adoption of the pairwise hinge loss is motivated by the good empirical results demonstrated in Rao et al. [167]. Additionally, we

also adopt the *mix sampling* strategy for sampling negative samples as described in their work.

10.2.6.2 Gradient Conversion

Since our network learns in hyperbolic space, parameters have to be learned via stochastic Riemannian optimization methods such as RSGD [251].

$$\theta_{t+1} = \mathfrak{R}_{\theta_t}(-\eta \nabla_R \ell(\theta_t)) \quad (10.7)$$

where \mathfrak{R}_{θ_t} denotes a retraction onto \mathcal{B} at θ_t . η is the learning rate and $\nabla_R \ell(\theta_t)$ is the Riemannian gradient with respect to θ_t . Fortunately, the Riemannian gradient can be easily derived from the Euclidean gradient in this case [251]. In order to do so, we can simply scale the Euclidean gradient by the inverse of the metric tensor g_θ^{-1} . Overall, the final gradients used to update the parameters are:

$$\nabla_R = \frac{(1 - \|\theta_t\|^2)^2}{4} \nabla_E \quad (10.8)$$

For more details, including the derivation of the Euclidean gradient, we refer interested readers to [246, 251] for more details. For practical purposes, we simply utilize the automatic gradient feature of TensorFlow [152] but convert the gradients in Equation (10.8) before updating the parameters.

10.3 Experiments

This section describes our empirical evaluation and its results.

10.3.1 Datasets

In the spirit of experimental rigor, we conduct our empirical evaluation based on four popular and well-studied benchmark datasets for question answering.

TrecQA As discussed in Chapter 4, TrecQA is the benchmark dataset provided by Wang et al. [232]. This dataset was collected from TREC QA tracks 8-13 and is

TABLE 10.1: Statistics of datasets.

	TrecQA	WikiQA	YahooCQA	SemEvalCQA
Train Qns	1229	94	501K	4.8K
Dev Qns	82	65	6.2K	224
Test Qns	100	68	6.2K	327
Train Pairs	53K	5.9K	253K	36K
Dev Pairs	1.1K	1.1K	31.7K	2.4K
Test Pairs	1.5K	1.4K	31.7K	3.2K

comprised of factoid based questions which mainly answers ‘who’, ‘what’, ‘where’, ‘when’, ‘why’ type of questions. There are two versions, clean and raw as noted by [167] which we evaluate our models on.

WikiQA As discussed in Chapter 9, this is a recently popular benchmark dataset [231] for open-domain question answering based on factual questions from Wikipedia and Bing search logs.

YahooCQA This is a benchmark dataset for community-based question answering that was collected from Yahoo Answers. In this dataset, the answer lengths are relatively longer than TrecQA and WikiQA. Therefore, we filtered answers that have more than 50 words and less than 5 characters. The train-dev-test splits for this dataset are provided by [252].

SemEvalCQA This is a well-studied benchmark dataset from SemEval-2016 Task 3 Subtask A (CQA). This is a real-world dataset obtained from Qatar Living Forums. In this dataset, there are ten answers in each question ‘thread’ which are marked as ‘Good’, ‘Potentially Useful’ or ‘Bad’. We treat ‘Good’ as positive and anything else as negative labels.

Statistics pertaining to each dataset is given in Table 10.1.

10.3.2 Compared Baselines

In this section, we introduce the following baselines for comparison.

TrecQA The key competitors on the dataset are mainly the CNN model of Severyn et al. [3], the Attention-based Neural Matching Model (aNMM) of Yang et al. [162], HD-LSTM (Tay et al.) [252] and Multi-Perspective CNN (MP-CNN) [98] proposed by He et al. Lastly, we also compare with the pairwise ranking adaption of the MP-CNN (Rao et al.) [167]. Additionally and due to long standing nature of this dataset, there have been a huge number of works based on traditional feature engineering approaches [232, 253–255] which we also report. For the clean version of this dataset, we also compare with AP-CNN and QA-BiLSTM/CNN [100].

WikiQA The key competitors of this dataset are the Paragraph Vector (PV) [256] of Le and Mikolv, CNN model from Yu et al. [257] and LCLR (Yih et al.) [258]. These three baselines are reported in the original WikiQA paper [231] which also include variations that include handcrafted features. Additional strong baselines include QA-BiLSTM and QA-CNN from [100] along with AP-BiLSTM and AP-CNN which are attentive pooling improvements of the former. Finally, we also report the Pairwise Ranking MP-CNN from Rao et al. [167].

YahooCQA The key competitors of this dataset are the Neural Tensor LSTM (NTN-LSTM) and HD-LSTM from Tay et al. [252] along with their implementation of the Convolutional Neural Tensor Network [95], CNN model of Severyn et al. [3] and the Okapi BM-25 [259] benchmark. Additionally, we also report our own implementations of QA-BiLSTM, QA-CNN, AP-BiLSTM and AP-CNN on this dataset based on our experimental setup.

SemEvalCQA The key competitors of this dataset are the CNN-based ARC-I/II architecture by Hu et al. [260], the Attentive Pooling CNN [100], Kelp (Filice et al.) [168] a feature engineering based SVM method, ConvKN [169] a combination of convolutional tree kernels with CNN and finally AI-CNN (Attentive Interactive CNN) [101], a tensor-based attentive pooling neural model. A comparison with AI-CNN (with features) is also included.

Since the training splits are standard, we are able to directly report the results from the original papers.

10.3.3 Evaluation Protocol

This section describes the key evaluation protocol/metrics and implementation details of our experiments.

Metrics We adopt a dataset specific evaluation protocol in which we follow the prior work in their evaluation protocols. Specifically, TrecQA and WikiQA adopt the Mean Reciprocal Rank (MRR) and MAP (Mean Average Precision) metrics which are commonplace in IR research. On the other hand, YahooCQA and SemEvalCQA are evaluated based on MAP and Precision@1 (abbreviated P@1) which are determined based on whether the top predicted answer is the ground truth. For all competitor methods, we report the performance results from the original paper.

Training time & Parameter Size Additionally, we report the parameter size and runtime (seconds per epoch) of selected models. We selectively re-implement some of the key competitors with the best performance and benchmark their training time on our machine/GPU (a single Nvidia GTX1070). For reporting the parameter size and training time, we try our best to follow the hyperparameters stated in the original papers. As such, the same model can have different training time and parameter size on different datasets.

Hyperparameters HYPERQA is implemented in TensorFlow [152]. We adopt the AdaGrad [261] optimizer with initial learning rate tuned amongst $\{0.2, 0.1, 0.05, 0.01\}$. The batch size is tuned amongst $\{50, 100, 200\}$. Models are trained for 25 epochs and the model parameters are saved each time the performance on the validation set is topped. The dimension of the projection layer is tuned amongst $\{100, 200, 300, 400\}$. L2 regularization is tuned amongst $\{0.001, 0.0001, 0.00001\}$. The negative sampling rate is tuned from 2–8. Finally, the margin λ is tuned amongst $\{1, 2, 5, 10, 20\}$. For TrecQA, WikiQA and YahooCQA, we initialize the embedding layer with GloVe [16] and use the version with $d = 300$ and trained on 840 billion words. For SemEvalCQA, we trained our own GloVe model using the unannotated corpus provided by the task. In this case, the embedding dimension is tuned amongst

$\{100, 200, 300\}$. Embeddings are not updated during training. For the SemEval-CQA dataset, we also found that it helped to concatenate the raw QA embeddings before passing into the final layer.

10.3.4 Results and Analysis

In this section, we present our empirical results on all datasets.

Experimental Results on TrecQA Table 10.2 reports the results on TrecQA (raw). HYPERQA achieves very competitive performance on both MAP and MRR metrics. Specifically, HYPERQA outperforms the basic CNN model of (Severyn et al.) by $2\% - 3\%$ in terms of MAP/MRR. Moreover, the CNN model of Severyn et al. uses handcrafted features which HYPERQA does not require. Similarly, the aNMM model and HD-LSTM also benefit from additional features but are outperformed by HYPERQA. HYPERQA also outperforms MP-CNN but is around 10 times faster and has 100 times less parameters. MP-CNN consists of a huge number of filter banks and utilizes heavy parameterization to match multiple perspectives of questions and answers. On the other hand, our proposed HYPERQA is merely a single layered neural network with 90K parameters and yet outperforms MP-CNN. Similarly, Table 10.3 reports the results on TrecQA (clean). Similarly, HYPERQA also outperforms MP-CNN, AP-CNN and QA-CNN. On both datasets, the performance of HYPERQA is competitive to Rank MP-CNN.

Experimental Results on WikiQA Table 10.4 reports our results on the WikiQA dataset. Firstly, we observe that HYPERQA outperforms a myriad of complex neural architectures. Notably, we obtain a clear performance gain of $2\% - 3\%$ in terms of MAP/MRR against models such as AP-CNN or AP-BiLSTM. Our model also outperforms MP-CNN which is severely equipped with parameterized word matching mechanisms. We achieve competitive results relative to the Rank MP-CNN. Finally, HYPERQA is extremely efficient and fast, clocking 2s per epoch compared to 33s per epoch for Rank MP-CNN. The parameter cost is also 90k vs 10 million which is a significant improvement.

TABLE 10.2: Results on TrecQA (raw) dataset.

Model	MAP	MRR	# Params	Time
Wang et al. (2007)	0.603	0.685	-	-
Heilman et al. (2010)	0.609	0.692	-	-
Wang et al. (2010)	0.595	0.695	-	-
Yao (2013)	0.631	0.748	-	-
Severyn et al. (2013)	0.678	0.736	-	-
Yih et al (2014)	0.709	0.770	-	-
CNN-Cnt (Yu et al.)	0.711	0.785	-	-
BLSTM + BM25	0.713	0.791	-	-
CNN (Severyn)	0.746	0.808	-	-
aNMM	0.750	0.811	-	-
HD-LSTM	0.750	0.815	-	-
MP-CNN	0.762	0.822	10.0M	141s
Rank MP-CNN	0.780	0.830	10.0M	130s
HYPERQA	<u>0.770</u>	<u>0.825</u>	90K	12s

TABLE 10.3: Results on TrecQA (clean) dataset.

Model	MAP	MRR	# Params	Time
QA-LSTM / CNN	0.728	0.832	-	-
AP-CNN	0.753	0.851	-	-
MP-CNN	0.777	0.836	10M	141
Rank MP-CNN	0.801	0.877	10M	130s
HyperQA	<u>0.784</u>	<u>0.865</u>	90K	12s

TABLE 10.4: Results on WikiQA dataset

Model	MAP	MRR	#Params	Time
PV	0.511	0.516	-	-
PV-Cnt	0.599	0.609	-	-
LCLR	0.599	0.609	-	-
CNN-Cnt	0.652	0.665	-	-
QA-BiLSTM	0.656	0.670	-	-
QA-CNN	0.670	0.682	-	-
AP-BiLSTM	0.671	0.684	-	-
AP-CNN	0.688	0.696	-	-
MP-CNN	0.693	0.709	10.0M	35s
Rank MP-CNN	0.701	0.718	10.0M	33s
HYPERQA	0.712	0.727	90K	2s

Experimental Results on YahooCQA Table 10.5 reports the experimental results on YahooCQA. First, we observe that HYPERQA outperforms AP-BiLSTM and AP-CNN significantly. Specifically, we outperform AP-BiLSTM, the runner-up model by 6% in terms of MRR and 10% in terms of MAP. Notably, HYPERQA is 32 times faster than AP-BiLSTM and has 20 times less parameters. Our approach

TABLE 10.5: Results on YahooCQA dataset.

Model	P@1	MRR	# Params	Time
Random Guess	0.200	0.457	-	-
BM-25	0.225	0.493	-	-
CNN	0.413	0.632	-	-
CNTN	0.465	0.632	-	-
LSTM	0.465	0.669	-	-
NTN-LSTM	0.545	0.731	-	-
HD-LSTM	0.557	0.735	-	-
QA-BiLSTM	0.508	0.683	1.40M	440s
QA-CNN	0.564	0.727	90.9K	60s
AP-CNN	0.560	0.726	540K	110s
AP-BiLSTM	0.568	0.731	1.80M	640s
HYPERQA	0.683	0.801	90.0K	20s

TABLE 10.6: Results on SemEvalCQA dataset.

Model	P@1	MAP	#Params	Time
ARC-I	0.741	0.771	-	-
ARC-II	0.753	0.780	-	-
AP-CNN	0.755	0.771	-	-
Kelp	0.751	0.792	-	-
ConvKN	0.755	0.777	-	-
AI-CNN	0.763	0.792	140K	3250s
AI-CNN + features	<u>0.769</u>	0.801	140K	3250s
HyperQA	0.809	<u>0.795</u>	45K	10s

shows that complicated attentive pooling mechanisms are not necessary for good performance.

Experimental Results on SemEvalCQA Table 10.6 reports the experimental results on SemEvalCQA. Our proposed approach achieves highly competitive performance on this dataset. Specifically, we have obtained the best P@1 performance overall, outperforming the state-of-the-art AI-CNN model by 3% in terms of P@1. The performance of our model on MAP is marginally short from the best performing model. Notably, AI-CNN has benefited from external handcrafted features. As such, comparing AI-CNN (w/o features) with HYPERQA shows that our proposed model is a superior neural ranking model. Next, we draw the readers attention to the time cost of AI-CNN. The training time per epoch is $\approx 3250s$ per epoch which is about 300 times longer than our model. AI-CNN is extremely cost prohibitive, i.e., attentive pooling is already very expensive and yet AI-CNN performs 3D attentive pooling. Evidently, its performance can be easily superseded

TABLE 10.7: Overall performance of HYPERQA.

Ours against	Performance	Params	Speed
AP-BiLSTM	1-7% better	20x less	32 x faster
AP-CNN	1-12% better	Same	3x faster
AI-CNN	Competitive	3x less	300x faster
MP-CNN	1-2% better	100x less	10x faster
Rank MP-CNN	Competitive	100x less	10x faster

in a much smaller training time and parameter cost. This raises questions about the effectiveness of the 3D attentive pooling mechanism.

Overall Analysis Overall, we summarize the key findings of our experiments.

- It is possible to achieve very competitive performance with small parameterization and no word matching or interaction layers. HYPERQA outperforms complex models such as MP-CNN and AP-BiLSTM on multiple datasets.
- The relative performance of HYPERQA is significantly better on large datasets, e.g., YahooCQA (253K training pairs) as opposed to smaller ones like WikiQA (5.9K training pairs). We believe that this is due to the fact that hyperbolic space is *seemingly* larger than Euclidean space.
- HYPERQA is extremely fast and trains at 10 – 20 times faster than complex models like MP-CNN. Note that if CPUs are used instead of GPUs which speeds convolutions up significantly, this disparity would be significantly larger.
- Our proposed approach does not require handcrafted features and yet outperforms models that benefit from them. This is evident on all datasets, i.e., HYPERQA outperforms CNN model with features (TrecQA and WikiQA) and AI-CNN + features on SemEvalCQA.

10.3.5 Effects of QA Embedding Size

In this section, we study the effects of the QA embedding size on performance. Figure 10.3 describes the relationship between QA embedding size (d) and MAP on the WikiQA dataset. Additionally, we include a simple baseline (CosineQA) which is

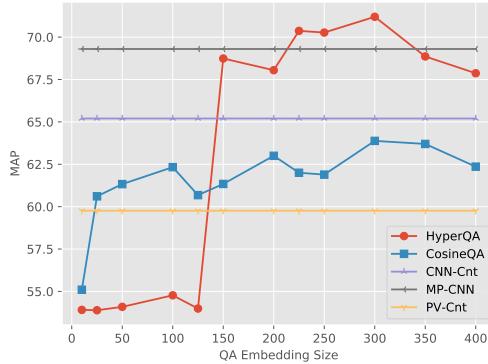


FIGURE 10.3: Effects of QA embedding size on WikiQA.

exactly the same as HYPERQA but uses cosine similarity instead of hyperbolic distance. The MAP scores of three other reported models (MP-CNN, CNN-Cnt and PV-Cnt) are also reported for reference. Firstly, we notice the disparity between HYPERQA and CosineQA in terms of performance. While CosineQA maintains a stable performance throughout embedding size, the performance of HYPERQA rapidly improves at $d > 150$. In fact, the performance of HYPERQA at $d = 150$ (45K parameters) is already similar to the Multi-Perspective CNN [98] which contains 10 million parameters. Moreover, the performance of HYPERQA outperforms MP-CNN with $d = 250\text{-}300$.

10.3.6 Discussion and Analysis

This section delves into qualitative analysis of our model.

10.3.7 Analysis of QA Embeddings

Figure 10.4 and Figure 10.5 show the visualization of QA embeddings on the test set TrecQA projected in 3-dimensional space using t-SNE [262]. QA embeddings are extracted from the network as discussed in Section 10.2.3. We observe that question embeddings form a ‘sphere’ over answer embeddings. Contrastingly, this is not exhibited when the cosine similarity is used as shown in Figure 10.5. It is important to note that these are embeddings from the **test** set which have not been trained and therefore the model is not explicitly told whether a particular textual

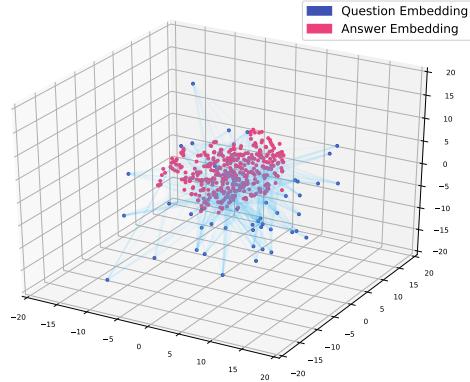


FIGURE 10.4: Embeddings from HYPERQA.

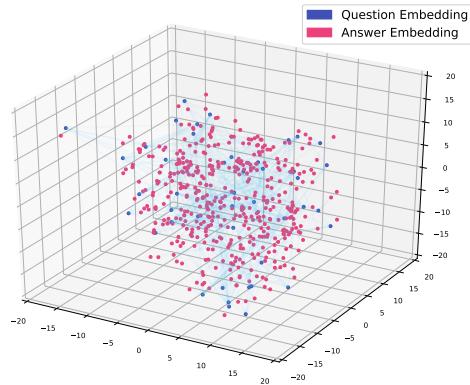


FIGURE 10.5: Embeddings from CosineQA.

input is a question or answer. This demonstrates the innate ability of HYPERQA to self-organize and learn latent hierarchies. Additionally, Figure 10.6(A) shows a histogram of the vector norms of question and answer embeddings. We can clearly see that questions in general have a higher vector norm² and are at a different hierarchical level from answers. In order to further understand what the model is doing, we delve deeper into visualization at a word-level.

10.3.8 Analysis of Word Embeddings

Table 10.8 shows some examples of words at each hierarchical level of the sphere on TrecQA. Recall that the vector norms³ allow us to infer the distance of the word embedding from the origin which depicts its hierarchical level in our context.

²We extract QA embeddings right before the constraining/normalization layer.

³Note that word embeddings are not constrained to $\|x\| < 1$.

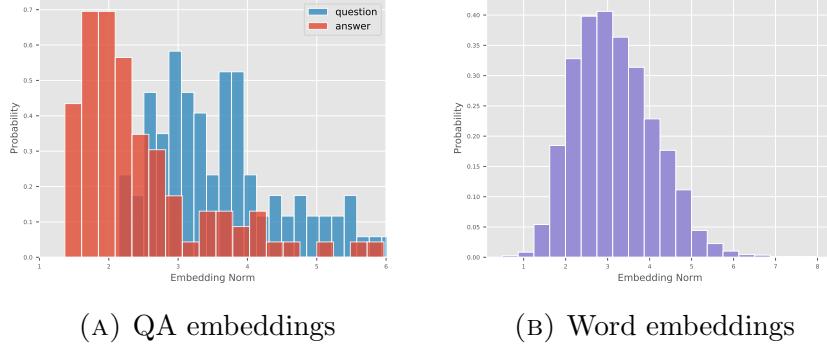


FIGURE 10.6: Histogram plots of embedding norms.

Interestingly, we find that HYPERQA exhibits self-organizing ability even at a word-level. Specifically, we notice that the words closer to the origin are common words such as ‘to’, ‘and’ which do not have much semantic values for QA problems. At the middle of the hierarchy ($\|w\| \approx 3$), we notice that there are more verbs. Finally, as we move towards the surface of the ‘sphere’, the words become rarer and reflect more domain-specific words such as ‘ebay’ and ‘spielberg’. Moreover, we also found many names and proper nouns occurring at this hierarchical level.

Additionally, we also observe that words such as ‘where’ or ‘what’ have relatively high vector norms and located quite high up in the hierarchy. This is in concert with Figure 10.4 which shows the question embeddings form a sphere around the answer embeddings. At last, we parsed QA pairs word-by-word according to hierarchical level (based on their vector norm).

Table 10.9 reports the outcome of this experiment where $H1 - H5$ are hierarchical levels based on vector norms. Most informative word matches are in boldface. Note that some words might be omitted from the answer for clarity.

First, we find that questions often start with the overall context and drill down into more specific query words. Take the first example in Table 10.9, it begins at a top level with ‘burger king’ and then drills down progressively to ‘what is gross sales?’. Similarly in the second example, it begins with ‘florence nightingale’ and drills down to ‘famous’ at H3 in which a match is being found with ‘nursing’ in the same hierarchical level. Overall, based on our qualitative analysis, we *believe* that, HYPERQA builds two hierarchical structures at the **word-level** (in vector space) towards the middle which strongly facilitates word-level matching. Pertaining to answers, it seems like the model builds a hierarchy by splitting on conjunctive words

TABLE 10.8: Examples of words in each hierarchical level.

$\ w\ $	Words (w)
0-1	to, and, an, on, in, of, its, the, had, or, go
1-2	be, a, was, up, put, said, but
2-3	judging, returning, volunteered, managing, meant, cited
3-4	responsibility, engineering, trading, prosecuting
4-5	turkish, autonomous, cowboys, warren, seven, what
5-6	ebay, magdalena, spielberg, watson, nova

TABLE 10.9: Analysis of QA pairs with respect to hierarchy.

Question		H1	H2	H3	H4	H5
What is the gross sale of Burger King	Q	are	sales , today	gross	is, what	burger, king
	A	based	sales, 14,billion , 183	diageo	contributed	burger, corp
What is Florence Nightingale famous for	Q	in, the	for	famous	what	florence, nightingale
	A	of, in	was	nursing	founder, modern, born	nightingale, italy
Who is the founder of twitter?	Q	the, of	-	twitter , founder	-	who, is
	A	and, the	in, net-working, launched	twitter, jack dorsey , july	match, social	-

(‘and’), i.e., the root node of the tree starts by conjunctive words and semantically segments.

10.4 Summary

We proposed a new neural ranking model for question answering. Our proposed HYPERQA achieves very competitive performance on four well-studied benchmark datasets. Our model is light-weight, fast and efficient outperforming many state-of-the-art models with complex word interaction layers or attentive mechanisms.

Our model only has 40K-90K parameters as opposed to millions of parameters which plague many competitor models. We derive qualitative insights pertaining to our model which enable us to further understand its inner workings. Finally, we believe that the superior generalization of our model (despite small parameters) can be attributed to ‘dual hierarchical matching’ which is effectively an automatic, free and parameter-less word interaction layer. The source code of our model can be found https://github.com/vanzytay/WSDM2018_HyperQA.

Chapter 11

Quaternion Representations for Natural Language Understanding

Neural network architectures such as Transformers [24, 263] and attention networks [23, 110, 264] are dominant solutions in both NLP and NLU research today. Many of these architectures are primarily concerned with learning useful feature representations from data in which providing a strong architectural inductive bias is known to be extremely helpful for obtaining stellar results. In previous chapters, we have studied a wide range of neural models and architectures ranging from individual units to entire end-to-end architectures. Many of these models are known to be heavily parameterized, with state-of-the-art models easily containing millions or billions of parameters [13–15, 24]. This renders practical deployment challenging. As such, the enabling of efficient and lightweight adaptations of these models, without significantly degrading performance, would certainly have a positive impact on many real-world applications. In this chapter¹, we discuss our proposed Quaternion models for natural language understanding.

11.1 Introduction

This chapter explores a new way to improve/maintain the performance of these neural architectures while substantially reducing the parameter cost (compression

¹This chapter is published as *Lightweight and Efficient Neural Natural Language Processing with Quaternion Networks*, Proceedings of ACL 2019 [69].

of up to 75%). In order to achieve this, we move beyond real space, exploring computation in Quaternion space (*i.e.*, hypercomplex numbers) as an inductive bias. Hypercomplex numbers comprise a real and three imaginary components (*e.g.*, i, j, k) in which inter-dependencies between these components are encoded naturally during training via the Hamilton product \otimes . Hamilton products have fewer degrees of freedom, enabling up to four times the compression of model size. Technical details are deferred to subsequent sections.

While Quaternion connectionist architectures have been considered in various deep learning application areas such as speech recognition [137], kinematics/human motion [265] and computer vision [266], our work is the first hypercomplex inductive bias designed for a wide spread of NLP tasks. Other fields have motivated the usage of Quaternions primarily due to their natural 3 or 4 dimensional input features (*e.g.*, RGB scenes or 3D human poses) [137, 265]. In a similar vein, we can similarly motivate this by considering the multi-sense nature of natural language [267–269]. In this case, having multiple embeddings or components per token is well-aligned with this motivation.

Latent interactions between components may also enjoy additional benefits, especially pertaining to applications that require learning pairwise affinity scores [110, 264]. Intuitively, instead of regular (real) dot products, Hamilton products \otimes extensively learn representations by matching across multiple (inter-latent) components in hypercomplex space. Alternatively, the effectiveness of multi-view and multi-headed [24] approaches may also explain the suitability of Quaternion spaces in NLP models. The added advantage to multi-headed approaches is that Quaternion spaces explicitly encode latent interactions between these components or heads via the Hamilton product which intuitively increases the expressiveness of the model. Conversely, multi-headed embeddings are generally independently produced.

To this end, we propose two Quaternion-inspired neural architectures, namely the Quaternion Attention model and the Quaternion Transformer. In this chapter, we devise and formulate a new attention (and self-attention) mechanism in Quaternion space using Hamilton products. Transformation layers are aptly replaced with Quaternion feed-forward networks, yielding substantial improvements in parameter size (of up to 75% compression) while achieving comparable (and occasionally better) performance.

The overall contributions of our work are as follows:

- We propose Quaternion neural models for NLP. More concretely, we propose a novel Quaternion Attention model and Quaternion Transformer for a wide range of NLP tasks. To the best of our knowledge, this is the first formulation of hypercomplex attention and Quaternion models for NLP.
- We evaluate our Quaternion NLP models on a wide range of diverse NLP tasks such as pairwise NLU (natural language inference, question answering, paraphrase identification, dialogue prediction), neural machine translation (NMT), sentiment analysis, mathematical language understanding (MLU), and subject-verb agreement (SVA).
- Our experimental results show that Quaternion models achieve comparable or better performance to their real-valued counterparts with up to a 75% reduction in parameter costs. The key advantage is that these models are expressive (due to Hamiltons) and also parameter efficient. Moreover, our Quaternion components are self-contained and play well with real-valued counterparts.

11.2 Proposed Method

11.2.1 Background on Quaternion Algebra

This section introduces the necessary background for this chapter. We introduce Quaternion algebra along with Hamilton products, which form the crux of our proposed approaches.

Quaternion A Quaternion $Q \in \mathbb{H}$ is a hypercomplex number with three imaginary components as follows:

$$Q = r + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}, \quad (11.1)$$

where $\mathbf{ijk} = \mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$ and noncommutative multiplication rules apply: $\mathbf{ij} = \mathbf{k}, \mathbf{jk} = \mathbf{i}, \mathbf{ki} = \mathbf{j}, \mathbf{ji} = -\mathbf{k}, \mathbf{kj} = -\mathbf{i}, \mathbf{ik} = -\mathbf{j}$. In Equation (11.1), r is a

real number and similarly, x, y, z are real numbers that represent the imaginary components of the Quaternion vector Q . Operations on Quaternions are defined in the following.

Addition and Subtraction The addition of two Quaternions is defined as:

$$Q + P = Q_r + P_r + (Q_x + P_x)\mathbf{i} + (Q_y + P_y)\mathbf{j} + (Q_z + P_z)\mathbf{k}$$

where Q and P with subscripts denote the real value and imaginary components of Quaternion Q and P . Subtraction follows this same principle analogously but flipping $+$ with $-$.

Scalar Multiplication Scalar α multiplies across all components, *i.e.*,

$$\alpha Q = \alpha r + \alpha x\mathbf{i} + \alpha y\mathbf{j} + \alpha z\mathbf{k}$$

Conjugate The conjugate of Q is defined as:

$$Q^* = r - x\mathbf{i} - y\mathbf{j} - z\mathbf{k}.$$

Norm The unit Quaternion Q^\triangleleft is defined as:

$$Q^\triangleleft = \frac{Q}{\sqrt{r^2 + x^2 + y^2 + z^2}}$$

Hamilton Product The Hamilton product, which represents the multiplication of two Quaternions Q and P , is defined as:

$$\begin{aligned} Q \otimes P &= (Q_r P_r - Q_x P_x - Q_y P_y - Q_z P_z) \\ &\quad + (Q_x P_r + Q_r P_x - Q_z P_y + Q_y P_z) \mathbf{i} \\ &\quad + (Q_y P_r + Q_z P_x + Q_r P_y - Q_x P_z) \mathbf{j} \\ &\quad + (Q_z P_r - Q_y P_x + Q_x P_y + Q_r P_z) \mathbf{k} \end{aligned} \tag{11.2}$$

which intuitively encourages inter-latent interaction between all the four components of Q and P . In this work, we use Hamilton products extensively for vector and matrix transformations that live at the heart of attention models for NLP.

11.2.2 Quaternion Models of Language

In this section, we propose Quaternion neural models for language processing tasks. We begin by introducing the building blocks, such as Quaternion Feed-Forward Layer, Quaternion Attention, and Quaternion Transformers.

11.2.2.1 Quaternion Feed-Forward Layer

A Quaternion Feed-Forward Layer is similar to a feed-forward layer in real space, while the former operates in hypercomplex space where Hamilton product is used. Let $W \in \mathbb{H}$ be the weight parameter of a Quaternion feed-forward layer and let $Q \in \mathbb{H}$ be the layer input. The linear output of the layer is the Hamilton product of two Quaternions: $W \otimes Q$.

Saving Parameters? How and Why In lieu of the fact that it might not be completely obvious at first glance why Quaternion models result in models with smaller parameterization, we dedicate the following to address this.

For the sake of parameterization comparison, let us express the Hamilton product $W \otimes Q$ in a Quaternion feed-forward layer in the form of matrix multiplication, which is used in real-space feed-forward. Recall the definition of Hamilton product in Equation (11.2). Putting aside the Quaterion unit basis $[1, \mathbf{i}, \mathbf{j}, \mathbf{k}]^\top$, $W \otimes Q$ can be expressed as:

$$\begin{bmatrix} W_r & -W_x & -W_y & -W_z \\ W_x & W_r & -W_z & W_y \\ W_y & W_z & W_r & -W_x \\ W_z & -W_y & W_x & W_r \end{bmatrix} \begin{bmatrix} r \\ x \\ y \\ z \end{bmatrix} \quad (11.3)$$

where $W = W_r + W_x\mathbf{i} + W_y\mathbf{j} + W_z\mathbf{k}$ and Q is defined in Equation (11.1).

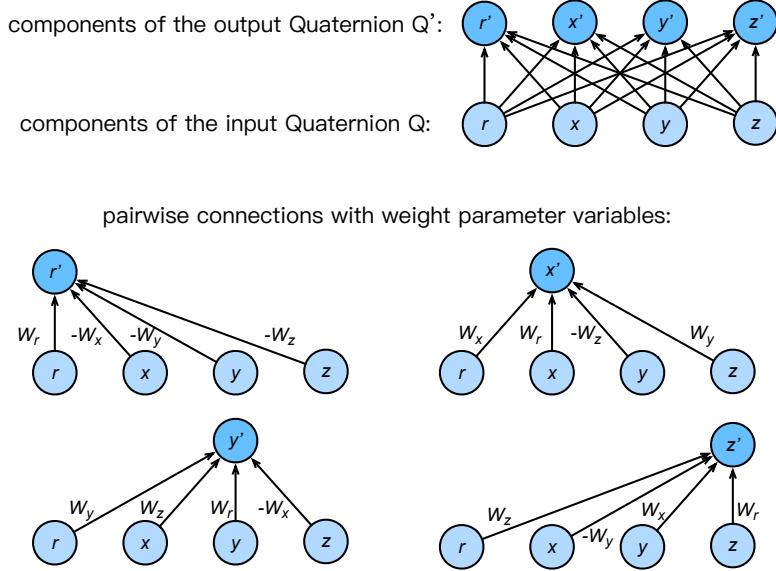


FIGURE 11.1: Quaternion weight sharing.

We highlight that, there are only 4 distinct parameter variable elements (4 degrees of freedom), namely W_r, W_x, W_y, W_z , in the weight matrix (left) of Equation (11.3), as illustrated by Figure 11.1. While in real-space feed-forward layer, all the elements of the weight matrix are different parameter variables ($4 \times 4 = 16$ degrees of freedom). In other words, the degrees of freedom in Quaternion feed-forward layer is only a quarter of those in its real-space counterpart, resulting in a 75% reduction in parameterization. Such a parameterization reduction can also be explained by *weight sharing* [137, 270]. As shown in Figure 11.1, 4 weight parameter variables (W_r, W_x, W_y, W_z) are used in 16 pairwise connections between components of the input and output Quaternions.

Nonlinearity Nonlinearity can be added to a Quaternion feed-forward layer and component-wise activation is adopted [270]:

$$\alpha(Q) = \alpha(r) + \alpha(x)\mathbf{i} + \alpha(y)\mathbf{j} + \alpha(z)\mathbf{k}$$

where Q is defined in Equation (11.1) and $\alpha(\cdot)$ is a nonlinear function such as tanh or ReLU.

Quaternion Attention Next, we propose a Quaternion Attention model to compute attention and alignment between two sequences. Let $A \in \mathbb{H}^{\ell_a \times d}$ and $B \in \mathbb{H}^{\ell_b \times d}$

be input word sequences, where ℓ_a, ℓ_b are numbers of tokens in each sequence and d is the dimension of each input vector. We first compute:

$$E = A \otimes B^\top$$

where $E \in \mathbb{H}^{\ell_a \times \ell_b}$. We apply $\text{Softmax}(.)$ to E component-wise:

$$G = \text{ComponentSoftmax}(E)$$

$$B' = G_R B_R + G_X B_X \mathbf{i} + G_Y B_Y \mathbf{j} + G_Z B_Z \mathbf{k}$$

where G and B with subscripts represent the real and imaginary components of G and B respectively. Similarly, we perform the same on A which is described as follows:

$$F = \text{ComponentSoftmax}(E^\top)$$

$$A' = F_R A_R + F_X A_X \mathbf{i} + F_Y A_Y \mathbf{j} + F_Z A_Z \mathbf{k}$$

where A' is the aligned representation of B and B' is the aligned representation of A . Next, given $A' \in \mathbb{R}^{\ell_b \times d}, B' \in \mathbb{R}^{\ell_a \times d}$ we then compute and **compare** the learned alignments:

$$\begin{aligned} C_1 &= \sum \text{QFFN}([A'_i; B_i, A'_i \otimes B_i; A'_i - B_i]) \\ C_2 &= \sum \text{QFFN}([B'_i; A_i, B'_i \otimes A_i; B'_i - A_i]) \end{aligned}$$

where $\text{QFFN}(.)$ is a Quaternion feed-forward layer with nonlinearity and $[;]$ is the component-wise contatentation operator. i refers to word positional indices and \sum over words in the sequence. Both outputs C_1, C_2 are then passed into the Quaternion feed-forward layer as follows:

$$Y = \text{QFFN}([C_1; C_2; C_1 \otimes C_2; C_1 - C_2])$$

where $Y \in \mathbb{H}$ is a Quaternion valued output. In order to train our model end-to-end with real-valued losses, we concatenate each component and pass into a final linear layer for classification.

11.2.3 Quaternion Transformer

This section describes our Quaternion adaptation of Transformer networks. Transformer [24] can be considered as state-of-the-art across many NLP tasks. Transformer networks are characterized by stacked layers of linear transforms along with its signature self-attention mechanism. For the sake of brevity, we outline the specific changes we make to the Transformer model.

Quaternion Self-Attention The standard self-attention mechanism considers the following:

$$A = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

where Q, K, V are traditionally learned via linear transforms from the input X . The key idea here is that we replace this linear transform with a Quaternion transform.

$$Q = W_q \otimes X; K = W_k \otimes X; V = W_v \otimes X$$

where \otimes is the Hamilton product and X is the input Quaternion representation of the layer. In this case, since computation is performed in Quaternion space, the parameters of W is effectively reduced by 75%. Similarly, the computation of self-attention also relies on Hamilton products. The revised Quaternion self-attention is defined as follows:

$$A = \text{ComponentSoftmax}\left(\frac{Q \otimes K}{\sqrt{d_k}}\right)V \quad (11.4)$$

Note that in Equation (11.4), $Q \otimes K$ returns four $\ell \times \ell$ matrices (attention weights) for each component (r, i, j, k) . Softmax is applied component-wise, along with multiplication with V which is multiplied in similar fashion to the Quaternion Attention model. Note that the Hamilton product in the self-attention itself does not change the parameter size of the network.

Quaternion Transformer Block Aside from the linear transformations for forming query, key, and values. Transformers also contain positional feed-forward networks with ReLU activations. Similarly, we replace the feed-forward connections

(FFNs) with Quaternion FFNs. We denote this as Quaternion Transformer (*full*) while denoting the model that only uses Quaternion FFNs in the self-attention as (*partial*). Finally, the remainder of the Transformer networks remain identical to the original design [24] in the sense that component-wise functions are applied unless specified above.

11.2.4 Embedding Layers

In the case where the word embedding layer is trained from scratch (*i.e.*, using byte-pair encoding in machine translation), we treat each embedding to be the concatenation of its four components. In the case where pre-trained embeddings such as GloVe [187] are used, a nonlinear transform is used to project the embeddings into Quaternion space.

11.2.5 Connection to Real Components

A vast majority of neural components in the deep learning arsenal operate in real space. As such, it would be beneficial for our Quaternion-inspired components to interface seamlessly with these components. If the input to a Quaternion module (such as Quaternion FFN or attention modules) is real, we simply treat the real-valued input as a concatenation of components r, x, y, z . Similarly, the output of the Quaternion module, if passed to a real-valued layer, is treated as a $[r; x; y; z]$, where $[;]$ is the concatenation operator.

Output Layer and Loss Functions To train our model, we simply concatenate all r, i, j, k components into a single vector at the final output layer. For example, for classification, the final Softmax output is defined as follows:

$$Y = \text{Softmax}(W([r; x; y; z]) + b)$$

where $Y \in \mathbb{R}^{|C|}$, $|C|$ is the number of classes and x, y, z are the imaginary components. Similarly for sequence loss (for sequence transduction problems), the same can be also done.

Parameter Initialization It is intuitive that specialized initialization schemes ought to be devised for Quaternion representations and their modules [137, 270].

$$w = |w|(\cos(\theta) + q_{imag}^\triangleleft \sin(\theta))$$

where q_{imag}^\triangleleft is the normalized imaginary constructed from uniform randomly sampling from $[0, 1]$. θ is randomly and uniformly sampled from $[-\pi, \pi]$. However, our early experiments show that, at least within the context of NLP applications, this initialization performed comparable or worse than the standard Glorot initialization. Hence, we opt to initialize all components independently with Glorot initialization.

11.3 Experiments

This section describes our experimental setup across multiple diverse NLP tasks. All experiments were run on NVIDIA Titan X hardware.

Our Models On pairwise text classification, we benchmark the Quaternion Attention model (Q-Att), testing the ability of Quaternion models on pairwise representation learning. On all the other tasks, such as machine translation and subject-verb agreement, we evaluate Quaternion Transformers. We evaluate two variations of Transformers, *full* and *partial*. The *full* setting converts all linear transformations into Quaternion space and is approximately 25% of the actual Transformer size. The second setting (*partial*) only reduces the linear transforms at the self-attention mechanism. Tensor2Tensor² is used for Transformer benchmarks, which uses its default hyperparameters and encoding for all experiments.

11.3.1 Pairwise Natural Language Understanding

We evaluate our proposed Quaternion Attention (Q-Att) model on pairwise text classification tasks. This task involves predicting a label or ranking score for sentence pairs. We use a total of seven datasets from problem domains as follows:

²<https://github.com/tensorflow/tensor2tensor>.

- **Natural language inference (NLI)** - This task is concerned with determining if two sentences entail or contradict each other. We use SNLI [6], SciTail [271], MNLI [272] as benchmark datasets.
- **Question answering (QA)** - This task involves learning to rank question-answer pairs. We use WikiQA [109] which comprises QA pairs from Bing Search.
- **Paraphrase detection** - This task involves detecting if two sentences are paraphrases of each other. We use Tweets [273] dataset and the Quora paraphrase dataset [51].
- **Dialogue response selection (RS)** - This is a response selection task that tries to select the best response given a message. We use the Ubuntu dialogue corpus, UDC [56].

Implementation Details We implement Q-Att in TensorFlow [274], along with the Decomposable Attention baseline [264]. Both models optimize the cross entropy loss (e.g., binary cross entropy for ranking tasks such as QA and RS). Models are optimized with Adam optimizer with the learning rate tuned amongst {0.001, 0.0003} and the batch size tuned amongst {32, 64}. Embeddings are initialized with GloVe [187]. For Q-Att, we use an additional transform layer to project the pre-trained embeddings into Quaternion space. The measures used are generally the accuracy measure (for NLI and paraphrase detection tasks) and ranking measures (MAP/MRR/Top-1) for ranking tasks (QA and RS).

Baselines and Comparison We use the Decomposable Attention model as a baseline, adding $[a_i; b_i; a_i \odot b_i; a_i - b_i]$ before the compare³ layers since we found this simple modification to increase performance. This also enables fair comparison with our variation of Quaternion Attention which uses Hamilton product over element-wise multiplication. We denote this as DeAtt. We evaluate at a fixed representation size of $d = 200$ (equivalent to $d = 50$ in Quaternion space). We also include comparisons at equal parameterization ($d = 50$ and approximately $200K$ parameters) to observe the effect of Quaternion representations.

³This follows the matching function of [275].

TABLE 11.1: Results for pairwise NLU tasks.

Task	NLI			QA		Paraphrase		RS	
Measure	Accuracy			MAP/MRR		Accuracy		Top-1	
Model	SNLI	SciTail	MNLI	WikiQA	Tweet	Quora	UDC	# θ	
DeAtt d=50	83.4	73.8	69.9/70.9	66.0/67.1	77.8	82.2	48.7	200K	
DeAtt d=200	86.2	79.0	73.6/73.9	67.2/68.3	80.0	85.4	51.8	700K	
Q-Att d=50	85.4	79.6	72.3/72.9	66.2/68.1	80.1	84.1	51.5	200K	

TABLE 11.2: Results for sentiment analysis.

Model	IMDb	SST	# Params
Transformer	82.6	78.9	400K
Quaternion Transformer (<i>full</i>)	83.9 (+1.3%)	80.5 (+1.6%)	100K (-75.0%)
Quaternion Transformer (<i>partial</i>)	83.6 (+1.0%)	81.4 (+2.5%)	300K (-25.0%)

Results Table 11.1 reports results on seven different and diverse datasets. We observe that a tiny Q-Att model ($d = 50$) achieves comparable (or occasionally marginally better or worse) performance compared to DeAtt ($d = 200$), gaining a 68% parameter savings. The results actually improve on certain datasets (2/7) and are comparable (often less than a percentage point difference) compared with the $d = 200$ DeAtt model. Moreover, we scaled the parameter size of the DeAtt model to be similar to the Q-Att model and found that the performance degrades quite significantly (about 2% – 3% lower on all datasets). This demonstrates the quality and benefit of learning with Quaternion space.

11.3.2 Sentiment Analysis

We evaluate on the task of document-level sentiment analysis which is a binary classification problem.

Implementation Details We compare our proposed Quaternion Transformer against the vanilla Transformer. In this experiment, we use the *tiny* Transformer setting in Tensor2Tensor with a vocab size of 8K. We use two datasets, namely IMDb [276] and Stanford Sentiment Treebank (SST) [228].

TABLE 11.3: Results for neural machine translation.

Model	BLEU			# Params
	<i>En-Vi</i>	<i>En-Ro</i>	<i>En-Et</i>	
Transformer Base	28.4	22.8	14.1	44M
Quaternion Transformer (<i>full</i>)	28.0	18.5	13.1	11M (-75%)
Quaternion Transformer (<i>partial</i>)	30.9	22.7	14.2	29M (-32%)

Results Table 11.2 reports results of the sentiment classification task on IMDb and SST. We observe that both the *full* and *partial* variations of Quaternion Transformers outperform the base Transformer. We observe that Quaternion Transformer (*partial*) obtains a +1.0% lead over the vanilla Transformer on IMDb and +2.5% on SST. This is while having a 24.5% saving in parameter cost. Finally, the *full* Quaternion version leads by +1.3%/1.6% gains on IMDb and SST respectively while maintaining a 75% reduction in parameter cost. This supports our core hypothesis of improving accuracy while saving parameter costs.

11.3.3 Neural Machine Translation

We evaluate our proposed Quaternion Transformer against vanilla Transformer on three datasets on this neural machine translation (NMT) task. More concretely, we evaluate on IWSLT 2015 English Vietnamese (*En-Vi*), WMT 2016 English-Romanian (*En-Ro*) and WMT 2018 English-Estonian (*En-Et*).

Implementation Details We implement models in Tensor2Tensor and trained for 50k steps for both models. We use the default base single GPU hyperparameter setting for both models and average checkpointing. Note that our goal is not to obtain state-of-the-art models but to fairly and systematically evaluate both vanilla and Quaternion Transformers.

Results Table 11.3 reports the results on neural machine translation. Parameter size excludes word embeddings. Our proposed Quaternion Transformer achieves comparable or higher performance with only 67.9% parameter costs of the base Transformer model.

On the IWSLT’15 En-Vi dataset, the *partial* adaptation of the Quaternion Transformer outperforms (+2.5%) the base Transformer with a 32% reduction in parameter cost. On the other hand, the *full* adaptation comes close (−0.4%) with a 75% reduction in parameter cost. On the WMT’16 En-Ro dataset, Quaternion Transformers do not outperform the base Transformer. We observe a −0.1% degrade in performance on the *partial* adaptation and −4.3% degrade on the *full* adaptation of the Quaternion Transformer. However, we note that the drop in performance with respect to parameter savings is still quite decent, *e.g.*, saving 32% parameters for a drop of only 0.1 BLEU points. The *full* adaptation loses out comparatively. On the WMT’18 En-Et dataset, the *partial* adaptation achieves the best result with 32% less parameters. The *full* adaptation, comparatively, only loses by 1.0 BLEU score from the original Transformer yet saving 75% parameters.

WMT’14 En-De Quaternion Transformer achieves a BLEU score of 26.42/25.14 for partial/full settings respectively on the standard WMT’14 En-De benchmark. This is using a single GPU trained for 1M steps with a batch size of 8192.

11.3.4 Mathematical Language Understanding

We include evaluations on a newly released mathematical language understanding (MLU) dataset [277]. This dataset is a character-level transduction task that aims to test a model’s compositional reasoning capabilities. For example, given an input $x = 85, y = -523$ and $x * y$ the model strives to decode an output of −44455. Several variations of these problems exist, mainly switching and introduction of new mathematical operators.

Implementation Details We train Quaternion Transformer for 100K steps using the default Tensor2Tensor setting following the original work [277]. We use the tiny hyperparameter setting. Similar to NMT, we report both *full* and *partial* adaptations of Quaternion Transformers. Baselines are reported from the original work as well, which include comparisons from Universal Transformers [263] and ACT Universal Transformers. The evaluation measure is accuracy per sequence, which counts a generated sequence as correct if and only if the entire sequence is an exact match.

TABLE 11.4: Results for mathematical language understanding (MLU).

Model	Acc / Seq	# Params
Universal Transformer	78.8	-
ACT U-Transformer	84.9	-
Transformer	76.1	400K
Quaternion Transformer (<i>full</i>)	78.9 (+2.8%)	100K (-75%)
Quaternion Transformer (<i>partial</i>)	84.4 (+8.3%)	300K (-25%)

TABLE 11.5: Results for subject-verb agreement (SVA).

Model	Acc	Params
Transformer	94.8	400K
Quaternion (<i>full</i>)	94.7	100K
Quaternion (<i>partial</i>)	95.5	300K

Results Table 11.4 reports our experimental results on the MLU dataset. We observe a modest +7.8% accuracy gain when using the Quaternion Transformer (*partial*) while saving 24.5% parameter costs. Quaternion Transformer outperforms Universal Transformer and marginally is outperformed by Adaptive Computation Universal Transformer (ACT U-Transformer) by 0.5%. On the other hand, a *full* Quaternion Transformer still outperforms the base Transformer (+2.8%) with 75% parameter saving.

11.3.5 Subject Verb Agreement

Additionally, we compare our Quaternion Transformer on the subject-verb agreement task [278]. The task is a binary classification problem, determining if a sentence, *e.g.*, ‘*The keys to the cabinet* ____ .’ follows by a plural/singular.

Implementation We use the Tensor2Tensor framework, training Transformer and Quaternion Transformer with the tiny hyperparameter setting with $10k$ steps.

Results Table 11.5 reports the results on the SVA task. Results show that Quaternion Transformers perform equally (or better) than vanilla Transformers. On this task, the partial adaptation performs better, improving Transformers by +0.7% accuracy while saving 25% parameters.

11.4 Summary

This chapter advocates for lightweight and efficient neural NLP via Quaternion representations. More concretely, we proposed two models - Quaternion Attention model and Quaternion Transformer. We evaluate these models on eight different NLP tasks and a total of *thirteen* datasets. Across all datasets, the Quaternion model achieves comparable performance while reducing parameter size. All in all, we demonstrated the utility and benefits of incorporating Quaternion algebra in state-of-the-art neural models. We believe that this direction paves the way for more efficient and effective representation learning in NLP.

Chapter 12

Conclusion

This chapter concludes the thesis and provides a retrospective summary of the proposed contributions. In this thesis, we presented novel state-of-the-art models for a series of natural language understanding (NLU) tasks such as natural language inference (NLI), machine reading comprehension (MRC) and retrieval-based NLU. We also proposed novel building blocks (e.g., encoding units) and embedding spaces for training efficient NLU models. Finally, we go on to elaborate on possible potential research directions.

12.1 Overall Summary

On well-contested international benchmarks for language inference (e.g., SNLI, MultiNLI), our models (e.g., CAFE, CSRAN) have held state-of-the-art performance on these leaderboards for extended periods of time, effectively pushing and advancing research in this field of language understanding. CAFE is largely based on factorized attention, using a newly proposed technique which we call *alignment factorization* and an overall paradigm, *Compare-Propagate*. These enable efficient computation due to the compression of alignment/attention and outperforms all existing competitors with fewer parameter costs. Additionally, CSRAN presents an extended deep version of CAFE, but proposing Multi-level Attention Refinement (MAR) and Co-Stack Residual Affinity (CSRA) mechanisms for further improving the results. Additionally, we also proposed Multi-Cast Attention Networks

(MCAN), which incorporates multi-casting of diverse attention flavors. MCAN achieves competitive performance on a myriad of retrieval-based NLU problems.

On Machine Reading Comprehension, our proposed DecaProp model similarly holds the state-of-the-art on several benchmark tasks such as NewsQA and NarrativeQA. DecaProp operates on a BAC (Bidirectional Attention Connector) module, chaining the entire network up with residual attention features. DecaProp achieves highly performant results and outperforming existing state-of-the-art models by up to 10% accuracy.

Additionally, we presented a state-of-the-art recommender system that is largely inspired by NLU-based research, leveraging user reviews for reasoning and recommendation. The proposed Multi-Pointer Co-Attention Network is a hierarchical model that reads and reasons over user review banks. We show that a two-layered attention-based reasoning model is capable of not only performing better but also producing explainable results. We analyze the model outputs, and show that different domains have different evidence aggregation patterns. This sheds light into the behavior of deep learning models on these real-world applications.

We then studied and presented two new sequential inductive biases for natural language understanding, e.g., Dilated Composition Units (DCU) and Recurrently Controlled Recurrent Networks (RCRN) which have demonstrated success across a myriad of NLU and NLP benchmarks. We show that designing specialized encoders brings about significant benefits as opposed to using off-the-shelf components such as GRU/LSTM units. More concretely, DCUs are faster due to non-reliant on auto-regressive computation. However, they gain expressiveness from multi-granular reasoning over sequences. On the other hand, RCRN learns powerful encoders by parameterizing recurrent gating functions with additional RNN units.

Finally, we study two novel inductive biases for making NLU models efficient. Hyperbolic representation, which is a naturally hierarchically structured non-Euclidean space. We show that simple, feed-forward networks trained in hyperbolic space may outperform advanced competitor models. We also studied Hypercomplex Quaternion representation which enables parameter savings by *Quaternion weight sharing*. We proposed Quaternion models of language, attaining competitive performance with approximately four times fewer parameters. While pursuing performance is

a great direction, we also advocate for lightweight neural models that are suitable for industrial and practical use.

12.2 Future Directions and Challenges

This section provides several future directions that are exciting. While there might be no principled solution to any of the following, many of these future directions remain as interesting goal posts for AI research in language understanding.

12.2.1 Multi-document Reasoning and Understanding

Although the works that we presented in this thesis have achieved stellar performance on many standard benchmarks [45, 46, 87], we believe it is good to investigate specialized inductive bias for enabling reasoning over multiple documents. In current NLU or QA setups, the reasoning is often constrained to sentence pairs or document-query pairs. To this end, the full potential of learning document-document relations is under-explored. We believe the study of specialized attention modules would go a long way in pushing this direction. Notably, while our proposed Multi-Pointer Co-Attention Networks [4] acknowledges that user reviews should be modeled independently and hierarchically, there might be still room for improvement in the introspection aspect. Several new recent works have proposed new challenges pertaining to this aspect [279, 280].

12.2.2 Conversational Language Understanding

This thesis presents several comprehensive experiments on a multitude of NLU tasks. However, conversations and dialogue remain central to society and human communication. Conversations may be informal with a huge requirement for modeling co-reference or user information. To this end, integrating ideas from NLU to this domain area might be an interesting future direction. This can be also interpreted as a form of multi-document reasoning in which each dialogue line can be treated as a single document. Recent challenges have started pushing boundaries of

MRC into this conversational setting [35] which makes it an exciting future direction. This would have immense applications, especially pertaining to the chatbot and personal assistant industry.

12.2.3 Efficient Models for Reading Long Documents

A major problem in MRC and NLU models that reading and comprehending long documents are extremely impossible and challenging. A long document may manifest in many domains such as narratives, web documents, or legal documents. The challenges may largely stem from the hardware inability to fit several thousand tokens into memory and reason over them. On the other hand, attention methods may face bottleneck, distributing their relative weights too sparsely across too many possible values. Existing self-attention [24] or alignment techniques, typically suffer from quadratic time and space complexity. This aggravates the problem of processing long documents. As such, this remains a highly technical challenge for future work with many areas for innovation.

12.2.4 Language Understanding for Fact Verification

The benefit of designing many general purpose neural modules for NLU is that we are able to simply utilize them in new applications. With the prevalent problem of misinformation and fake news, it is certainly a promising direction to enable fact verification via language understanding. More often than not, a fine-grained understanding of facts (within documents) is required for making complex decisions pertaining to misinformation. We hypothesize that NLU modules and models will play a huge role in detecting and eliminating misinformation.

Bibliography

- [1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016. [1](#), [3](#), [4](#), [6](#), [20](#), [82](#), [89](#), [135](#)
- [2] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 583–591. ACM, 2018. [1](#), [11](#), [12](#), [158](#)
- [3] Aliaksei Severyn and Alessandro Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pages 373–382, 2015. doi: 10.1145/2766462.2767738. [1](#), [134](#), [158](#), [167](#)
- [4] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Multi-range reasoning for machine comprehension. *arXiv preprint arXiv:1803.09074*, 2018. [1](#), [25](#), [26](#), [82](#), [196](#)
- [5] Karl Moritz Hermann, Tomas Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015. [1](#), [2](#), [3](#), [20](#), [89](#)
- [6] 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. [1](#), [188](#)
- [7] Mengqiu Wang, Noah A Smith, and Teruko Mitamura. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proceedings of the 2007*

- Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), 2007.* 1, 19
- [8] Aliaksei Severyn and Alessandro Moschitti. Automatic feature engineering for answer selection and extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 458–467, 2013. 1
- [9] Vasin Punyakanok, Dan Roth, and Wen-tau Yih. Natural language inference via dependency tree mapping: An application to question answering. Technical report, 2004. 1
- [10] Michael Heilman and Noah A Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1011–1019. Association for Computational Linguistics, 2010. 1
- [11] Jianfeng Gao, Michel Galley, Lihong Li, et al. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval*, 13(2-3): 127–298, 2019. 2
- [12] Adam Trischler, Zheng Ye, Xingdi Yuan, and Kaheer Suleman. Natural language comprehension with the epireader. *arXiv preprint arXiv:1606.02270*, 2016.
- [13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 15, 178
- [14] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. 15
- [15] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2, 154, 155, 178
- [16] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*,

- October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543, 2014. [2](#), [42](#), [48](#), [83](#), [137](#), [150](#), [162](#), [168](#)
- [17] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018. [2](#), [15](#), [17](#), [42](#), [43](#), [152](#), [154](#)
- [18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. [2](#), [7](#), [23](#), [24](#), [126](#), [142](#)
- [19] 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. [7](#), [23](#), [126](#), [142](#)
- [20] Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. Ordered neurons: Integrating tree structures into recurrent neural networks. *Proceedings of ICLR*, 2019. [2](#)
- [21] Ankur P. 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, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2249–2255, 2016. [2](#), [16](#), [25](#), [26](#), [29](#), [32](#), [42](#), [43](#), [49](#), [52](#), [55](#), [56](#), [57](#), [154](#), [155](#)
- [22] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015. [16](#), [25](#), [29](#), [43](#), [52](#), [142](#)
- [23] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. [22](#), [25](#), [26](#), [52](#), [142](#), [178](#)
- [24] 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 6000–6010, 2017. [7](#), [15](#), [25](#), [26](#), [77](#), [143](#), [154](#), [178](#), [179](#), [185](#), [186](#), [197](#)

- [25] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM, 2017. [2](#), [19](#), [20](#), [126](#), [136](#)
- [26] Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, et al. Conversational ai: The science behind the alexa prize. *arXiv preprint arXiv:1801.03604*, 2018. [2](#)
- [27] Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, et al. Smart reply: Automated response suggestion for email. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 955–964. ACM, 2016.
- [28] Aston Zhang, Lluis Garcia-Pueyo, James B Wendt, Marc Najork, and Andrei Broder. Email category prediction. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 495–503. International World Wide Web Conferences Steering Committee, 2017. [2](#)
- [29] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, 2014. [2](#)
- [30] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM, 2014. [2](#), [18](#), [52](#)
- [31] Zhengyuan Liu, Jia Hui Hazel Lim, Nur Farah Ain Binte Sahimi, Shao Chuen Tong, Sharon Ong, Angela Ng, Sheldon Shao Guang Lee, Michael Ross Macdonald, Savitha Ramasamy, Pavitra Krishnaswamy, et al. Fast prototyping a dialogue comprehension system for nurse-patient conversations on symptom monitoring. *arXiv preprint arXiv:1903.03530*, 2019. [2](#)

- [32] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016. [2](#)
- [33] Adam L. Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu O. Mittal. Bridging the lexical chasm: statistical approaches to answer-finding. In *SIGIR*, pages 192–199, 2000. doi: 10.1145/345508.345576. [2](#)
- [34] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*, 2016. [2](#), [15](#), [19](#), [20](#), [73](#), [77](#), [82](#), [90](#), [126](#), [135](#)
- [35] Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. *arXiv preprint arXiv:1808.07042*, 2018. [2](#), [197](#)
- [36] Jinfeng Rao, Wei Yang, Yuhao Zhang, Ferhan Ture, and Jimmy Lin. Multi-perspective relevance matching with hierarchical convnets for social media search. *arXiv preprint arXiv:1805.08159*, 2018. [2](#), [6](#)
- [37] Lisa Bauer, Yicheng Wang, and Mohit Bansal. Commonsense for generative multi-hop question answering tasks. *arXiv preprint arXiv:1809.06309*, 2018. [3](#)
- [38] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. Variational reasoning for question answering with knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. [3](#)
- [39] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*, 2017.
- [40] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. Multi-step retriever-reader interaction for scalable open-domain question answering. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=HkfPSh05K7>. [3](#), [18](#), [19](#)
- [41] Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. Swag: A large-scale adversarial dataset for grounded commonsense inference. *arXiv preprint arXiv:1808.05326*, 2018. [3](#)

- [42] Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2012. 3
- [43] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*, 2017. 3, 10, 21, 128, 135, 138
- [44] Bill MacCartney. *Natural language inference*. Citeseer, 2009. 4
- [45] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 632–642, 2015. 4, 16, 28, 41, 149, 196
- [46] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*, 2016. 4, 9, 21, 75, 82, 89, 196
- [47] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017. 4, 6, 21
- [48] Matthew Dunn, Levent Sagun, Mike Higgins, Ugur Guney, Volkan Cirik, and Kyunghyun Cho. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*, 2017. 6, 9, 21, 75, 82, 128, 135, 136, 137, 139
- [49] Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. *Transactions of the Association of Computational Linguistics*, 6:317–328, 2018. 6, 9, 89, 98, 99, 100
- [50] Aliaksei Severyn and Alessandro Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015. 6, 18, 52

- [51] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*, 2017. [188](#)
- [52] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Cross temporal recurrent networks for ranking question answer pairs. *arXiv preprint arXiv:1711.07656*, 2017. [6](#), [19](#)
- [53] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015. [6](#)
- [54] Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. Match-srnn: Modeling the recursive matching structure with spatial rnn. *arXiv preprint arXiv:1604.04378*, 2016. [6](#), [18](#)
- [55] Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. Knowledge enhanced hybrid neural network for text matching. *arXiv preprint arXiv:1611.04684*, 2016. [6](#), [18](#), [19](#), [61](#), [62](#)
- [56] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*, 2015. [6](#), [19](#), [61](#), [188](#)
- [57] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014. [7](#), [153](#)
- [58] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning- Volume 70*, pages 1243–1252. JMLR.org, 2017. [7](#)
- [59] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. Compare, compress and propagate: Enhancing neural architectures with alignment factorization for natural language inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1565–1575, 2018. [8](#), [12](#), [28](#)
- [60] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. Co-stack residual affinity networks with multi-level attention refinement for matching text sequences. In

- Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4492–4502, 2018. 8, 12, 28, 154
- [61] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Multi-cast attention networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD ’18, pages 2299–2308, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5552-0. doi: 10.1145/3219819.3220048. URL <http://doi.acm.org/10.1145/3219819.3220048>. 8, 12, 51
- [62] Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Jian Su. Densely connected attention propagation for reading comprehension. In *Advances in Neural Information Processing Systems*, pages 4906–4917, 2018. 9, 12, 73
- [63] Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*, 2017. 9, 21, 75, 82
- [64] Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. *arXiv preprint arXiv:1712.07040*, 2017. 9, 21, 75, 82, 128, 136, 137, 140, 149
- [65] Yi Tay, Shuohang Wang, Anh Tuan Luu, Jie Fu, Minh C. Phan, Xingdi Yuan, Jinfeng Rao, Siu Cheung Hui, and Aston Zhang. Simple and effective curriculum pointer-generator networks for reading comprehension over long narratives. In *Proceedings of the ACL 2019*, 2019. 9, 12, 89
- [66] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. Multi-pointer co-attention networks for recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2309–2318. ACM, 2018. 10, 12, 105
- [67] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. Multi-granular sequence encoding via dilated compositional units for reading comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2141–2151, Brussels, Belgium, October–November 2018. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/D18-1238>. 10, 12, 126

- [68] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. Recurrently controlled recurrent networks. In *Advances in Neural Information Processing Systems*, pages 4731–4743, 2018. [11](#), [12](#), [142](#)
- [69] Yi Tay, Aston Zhang, Anh Tuan Luu, Jinfeng Rao, Shuai Zhang, Shuohang Wang, Jie Fu, and Siu Cheung Hui. Lightweight and efficient neural natural language processing with quaternion networks. In *Proceedings of the ACL 2019*, 2019. [12](#), [178](#)
- [70] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. 2016. [15](#)
- [71] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. 2019. <http://www.d2l.ai>. [15](#)
- [72] Bill Maccartney. *Natural Language Inference*. PhD thesis, Stanford, CA, USA, 2009. AAI3364139. [15](#)
- [73] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW’05, pages 177–190, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-33427-0, 978-3-540-33427-9.
- [74] Bill MacCartney and Christopher D. Manning. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING ’08, pages 521–528, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. ISBN 978-1-905593-44-6.
- [75] Adrian Iftene and Alexandra Balahur-Dobrescu. Hypothesis transformation and semantic variability rules used in recognizing textual entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, RTE ’07, pages 125–130, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. [15](#)
- [76] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced LSTM for natural language inference. In *Proceedings of*

- the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1657–1668, 2017. doi: 10.18653/v1/P17-1152. [16](#), [29](#), [41](#), [42](#), [43](#), [47](#), [49](#), [52](#), [58](#), [154](#), [155](#)
- [77] Shuohang Wang and Jing Jiang. Learning natural language inference with LSTM. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1442–1451, 2016. [43](#)
- [78] Hong Yu and Tsendsuren Munkhdalai. Neural semantic encoders. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 397–407, 2017. [16](#)
- [79] Shuohang Wang and Jing Jiang. A compare-aggregate model for matching text sequences. *CoRR*, abs/1611.01747, 2016. [16](#), [18](#), [134](#)
- [80] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4144–4150, 2017. doi: 10.24963/ijcai.2017/579. [16](#), [18](#), [32](#), [41](#), [43](#), [63](#), [136](#)
- [81] Yichen Gong, Heng Luo, and Jian Zhang. Natural language inference over interaction space. *CoRR*, abs/1709.04348, 2017. [16](#), [32](#), [41](#), [43](#), [49](#)
- [82] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. Disan: Directional self-attention network for rnn/cnn-free language understanding. *CoRR*, abs/1709.04696, 2017. [17](#), [43](#)
- [83] Jihun Choi, Kang Min Yoo, and Sang-goo Lee. Unsupervised learning of task-specific tree structures with tree-lstms. *arXiv preprint arXiv:1707.02786*, 2017. [17](#), [43](#), [143](#), [144](#), [154](#)
- [84] Tushar Khot, Ashish Sabharwal, and Peter Clark. Scitail: A textual entailment dataset from science question answering. In *AAAI*, 2018. [17](#), [41](#), [42](#), [44](#), [49](#), [149](#), [154](#), [155](#)

- [85] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308, 2017. [17](#), [42](#), [43](#), [86](#), [142](#), [143](#), [144](#), [150](#), [152](#), [153](#)
- [86] Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016. [17](#)
- [87] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *CoRR*, abs/1704.05426, 2017. [17](#), [41](#), [44](#), [46](#), [196](#)
- [88] Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. R3: Reinforced reader-ranker for open-domain question answering. *arXiv preprint arXiv:1709.00023*, 2017. [18](#), [20](#), [82](#)
- [89] Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. Denoising distantly supervised open-domain question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1736–1745, 2018. [18](#), [19](#), [90](#)
- [90] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, 2014. [18](#), [62](#), [64](#)
- [91] Yi Tay, Minh C. Phan, Anh Tuan Luu, and Siu Cheung Hui. Learning to rank question answer pairs with holographic dual LSTM architecture. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 695–704, 2017. doi: 10.1145/3077136.3080790. [18](#)
- [92] Di Wang and Eric Nyberg. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual*

- Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 707–712, 2015. [18](#)
- [93] Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016*, 2016. [18](#)
- [94] Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. Inter-weighted alignment network for sentence pair modeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1179–1189, 2017. [18, 52, 63](#)
- [95] Xipeng Qiu and Xuanjing Huang. Convolutional neural tensor network architecture for community-based question answering. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1305–1311, 2015. [18, 62, 167](#)
- [96] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2835–2841, 2016. [18, 62](#)
- [97] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. Text matching as image recognition. 2016. [18, 62](#)
- [98] Hua He, Kevin Gimpel, and Jimmy J. Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1576–1586, 2015. [18, 63, 167, 173](#)
- [99] Jinfeng Rao, Wei Yang, Yuhao Zhang, Ferhan Ture, and Jimmy Lin. Multi-perspective relevance matching with hierarchical convnets for social media search, 2018. [18](#)

- [100] Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. Attentive pooling networks. *CoRR*, abs/1602.03609, 2016. [18](#), [26](#), [52](#), [57](#), [62](#), [63](#), [64](#), [66](#), [150](#), [155](#), [161](#), [167](#)
- [101] Xiaodong Zhang, Sujian Li, Lei Sha, and Houfeng Wang. Attentive interactive neural networks for answer selection in community question answering. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3525–3531, 2017. [18](#), [19](#), [52](#), [64](#), [161](#), [167](#)
- [102] Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. Word-entity duet representations for document ranking. *arXiv preprint arXiv:1706.06636*, 2017. [18](#)
- [103] Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. Sentence similarity learning by lexical decomposition and composition. *arXiv preprint arXiv:1602.07019*, 2016. [18](#), [48](#), [49](#), [63](#)
- [104] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Cross temporal recurrent networks for ranking question answer pairs, 2017. [18](#), [64](#), [65](#)
- [105] Pengfei Liu, Xipeng Qiu, Yaqian Zhou, Jifan Chen, and Xuanjing Huang. Modelling interaction of sentence pair with coupled-lstms. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1703–1712, 2016. [18](#)
- [106] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM '18*, pages 583–591, 2018. ISBN 978-1-4503-5581-0. [18](#), [63](#)
- [107] Peng Zhang, Jiabin Niu, Zhan Su, Benyou Wang, Liqun Ma, and Dawei Song. End-to-end quantum-like language models with application to question answering. 2018. [18](#)
- [108] Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. R 3: Reinforced ranker-reader for open-domain question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. [19](#), [90](#), [99](#)

- [109] Yi Yang, Wen-tau Yih, and Christopher Meek. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, 2015. [19](#), [188](#)
- [110] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016. [19](#), [20](#), [25](#), [26](#), [73](#), [74](#), [82](#), [90](#), [126](#), [133](#), [135](#), [137](#), [140](#), [178](#), [179](#)
- [111] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604, 2016. [19](#), [26](#), [52](#), [55](#), [73](#), [86](#), [126](#), [142](#)
- [112] Caiming Xiong, Victor Zhong, and Richard Socher. Dcn+: Mixed objective and deep residual coattention for question answering. *arXiv preprint arXiv:1711.00106*, 2017. [19](#), [20](#), [26](#), [74](#), [75](#)
- [113] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198, 2017. [19](#), [26](#), [73](#), [74](#), [80](#), [86](#), [99](#), [137](#), [140](#), [143](#), [151](#)
- [114] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017. [19](#), [90](#)
- [115] Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*, 2016. [19](#)
- [116] Minghao Hu, Yuxing Peng, and Xipeng Qiu. Mnemonic reader for machine comprehension. *arXiv preprint arXiv:1705.02798*, 2017. [19](#), [20](#), [126](#)
- [117] Souvik Kundu and Hwee Tou Ng. A question-focused multi-factor attention network for question answering. 2018. [19](#), [20](#), [73](#), [82](#), [133](#), [136](#), [139](#)

- [118] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018. [20](#), [26](#), [74](#), [75](#), [86](#)
- [119] Lei Zheng, Vahid Noroozi, and Philip S. Yu. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017*, pages 425–434, 2017. [21](#), [22](#), [106](#), [107](#), [116](#)
- [120] Rose Catherine and William Cohen. Transnets: Learning to transform for recommendation. *arXiv preprint arXiv:1704.02298*, 2017. [22](#), [106](#), [107](#), [116](#), [119](#)
- [121] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems, RecSys '16*, New York, NY, USA, 2016. ACM. [117](#)
- [122] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM, 2013. [21](#), [106](#), [117](#)
- [123] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys '17*, 2017. [21](#), [22](#), [106](#), [107](#), [115](#), [117](#)
- [124] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 193–202, 2014. doi: 10.1145/2623330.2623758. [21](#), [106](#)
- [125] Chong Wang and David M Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international*

- conference on Knowledge discovery and data mining*, pages 448–456. ACM, 2011. 21, 117
- [126] Steffen Rendle. Factorization machines. In *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, pages 995–1000, 2010. 22, 106, 114, 116
- [127] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM, 2015. 22, 115
- [128] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, pages 507–517. International World Wide Web Conferences Steering Committee, 2016. 22, 115
- [129] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576*, 2016. 24
- [130] Nan Rosemary Ke, Anirudh Goyal ALIAS PARTH GOYAL, Olexa Bilaniuk, Jonathan Binas, Michael C Mozer, Chris Pal, and Yoshua Bengio. Sparse attentive backtracking: Temporal credit assignment through reminding. In *Advances in Neural Information Processing Systems*, pages 7640–7651, 2018. 24
- [131] Adam Santoro, Ryan Faulkner, David Raposo, Jack Rae, Mike Chrzanowski, Theophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy Lillicrap. Relational recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 7299–7310, 2018. 24
- [132] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015. 24, 143
- [133] Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark A Hasegawa-Johnson, and Thomas S

- Huang. Dilated recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 76–86, 2017. [24](#), [128](#), [143](#)
- [134] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019. [24](#)
- [135] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems*, pages 5345–5355, 2018. [24](#)
- [136] Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128, 2016. [24](#)
- [137] Titouan Parcollet, Mirco Ravanelli, Mohamed Mørchid, Georges Linarès, Chiheb Trabelsi, Renato De Mori, and Yoshua Bengio. Quaternion recurrent neural networks. *arXiv preprint arXiv:1806.04418*, 2018. [24](#), [179](#), [183](#), [187](#)
- [138] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Couplenet: Paying attention to couples with coupled attention for relationship recommendation. In *Twelfth International AAAI Conference on Web and Social Media*, 2018. [26](#)
- [139] Yi Tay, Luu Anh Tuan, Siu Cheung Hui, and Jian Su. Reasoning with sarcasm by reading in-between. *arXiv preprint arXiv:1805.02856*, 2018. [26](#)
- [140] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014. [26](#)
- [141] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015. [26](#)
- [142] Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, et al. Hyperbolic attention networks. *arXiv preprint arXiv:1805.09786*, 2018. [26](#)

- [143] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. Hermitian co-attention networks for text matching in asymmetrical domains. In *IJCAI*, pages 4425–4431, 2018. [26](#)
- [144] Steffen Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, 2010. [30](#), [34](#), [36](#), [58](#), [78](#)
- [145] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Recurrent neural network-based sentence encoder with gated attention for natural language inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP, RepEval@EMNLP 2017, Copenhagen, Denmark, September 8, 2017*, pages 36–40, 2017. [32](#)
- [146] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015. [32](#), [55](#), [74](#), [75](#), [132](#), [150](#)
- [147] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617*, 2017. [37](#)
- [148] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and H Chi. Latent cross: Making use of context in recurrent recommender systems. 2018. [37](#)
- [149] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences, 2017. [38](#), [48](#), [49](#)
- [150] Dirk Weissenborn. Reading twice for natural language understanding. *CoRR*, abs/1706.02596, 2017. [42](#), [44](#)
- [151] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, and Diana Inkpen. Natural language inference with external knowledge. *arXiv preprint arXiv:1711.04289*, 2017. [42](#), [43](#)
- [152] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur,

- Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. [42](#), [48](#), [83](#), [137](#), [165](#), [168](#)
- [153] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. [42](#), [48](#), [62](#), [83](#), [117](#), [137](#), [150](#)
- [154] Yixin Nie and Mohit Bansal. Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP, RepEval@EMNLP 2017, Copenhagen, Denmark, September 8, 2017*, pages 41–45, 2017. [43](#), [143](#), [144](#), [154](#)
- [155] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 551–561, 2016. [43](#)
- [156] Hong Yu and Tsendsuren Munkhdalai. Neural tree indexers for text understanding. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 11–21, 2017. [43](#), [152](#)
- [157] Lei Sha, Baobao Chang, Zhifang Sui, and Sujian Li. Reading and thinking: Re-read LSTM unit for textual entailment recognition. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2870–2879, 2016. [43](#)
- [158] Gaurav Singh Tomar, Thyago Duque, Oscar Täckström, Jakob Uszkoreit, and Dipanjan Das. Neural paraphrase identification of questions with noisy pretraining. *arXiv preprint arXiv:1704.04565*, 2017. [48](#), [49](#)

- [159] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010. 48
- [160] Reza Ghaeini, Sadid A Hasan, Vivek Datla, Joey Liu, Kathy Lee, Ashequl Qadir, Yuan Ling, Aaditya Prakash, Xiaoli Z Fern, and Oladimeji Farri. Drbilstm: Dependent reading bidirectional lstm for natural language inference. *arXiv preprint arXiv:1802.05577*, 2018. 49
- [161] Bingning Wang, Kang Liu, and Jun Zhao. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016. 52
- [162] Liu Yang, Qingyao Ai, Jiafeng Guo, and W. Bruce Croft. anmm: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 287–296, 2016. doi: 10.1145/2983323.2983818. 167
- [163] Hua He and Jimmy J. Lin. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 937–948, 2016. 52, 158, 161
- [164] Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. Incorporating loose-structured knowledge into lstm with recall gate for conversation modeling. *arXiv preprint arXiv:1605.05110*, 2016. 61
- [165] Zhengdong Lu and Hang Li. A deep architecture for matching short texts. In *Advances in Neural Information Processing Systems*, 2013. 62
- [166] Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. What is the jeopardy model? A quasi-synchronous grammar for QA. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2007. 63

- [167] Jinfeng Rao, Hua He, and Jimmy J. Lin. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 1913–1916, 2016. doi: 10.1145/2983323.2983872. [63](#), [149](#), [164](#), [166](#), [167](#)
- [168] Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. Kelp at semeval-2016 task 3: Learning semantic relations between questions and answers. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 1116–1123, 2016. [64](#), [167](#)
- [169] Alberto Barrón-Cedeño, Giovanni Da San Martino, Shafiq R. Joty, Alessandro Moschitti, Fahad Al-Obaidli, Salvatore Romeo, Kateryna Tymoshenko, and Antonio Uva. Convkn at semeval-2016 task 3: Answer and question selection for question answering on arabic and english fora. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 896–903, 2016. [64](#), [167](#)
- [170] 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. [74](#)
- [171] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269, 2017. doi: 10.1109/CVPR.2017.243. URL <https://doi.org/10.1109/CVPR.2017.243>. [74](#), [75](#)
- [172] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. 2015. [74](#), [75](#)
- [173] Zixiang Ding, Rui Xia, Jianfei Yu, Xiang Li, and Jian Yang. Densely connected bidirectional lstm with applications to sentence classification. *arXiv preprint arXiv:1802.00889*, 2018. [75](#), [143](#), [153](#)

- [174] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. A compare-propagate architecture with alignment factorization for natural language inference. *arXiv preprint arXiv:1801.00102*, 2017. [78](#), [154](#)
- [175] Dirk Weissenborn, Georg Wiese, and Laura Seiffe. Making neural qa as simple as possible but not simpler. *arXiv preprint arXiv:1703.04816*, 2017. [82](#), [126](#)
- [176] Dirk Weissenborn. Reading twice for natural language understanding. *arXiv preprint arXiv:1706.02596*, 2017. [82](#)
- [177] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*, 2016. [82](#), [99](#), [137](#), [139](#), [140](#)
- [178] Nan Rosemary Ke, Konrad Zolna, Alessandro Sordoni, Zhouhan Lin, Adam Trischler, Yoshua Bengio, Joelle Pineau, Laurent Charlin, and Chris Pal. Focused hierarchical rnns for conditional sequence processing. *arXiv preprint arXiv:1806.04342*, 2018. [82](#)
- [179] Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Andrea Gesmundo, Neil Houlsby, Wojciech Gajewski, and Wei Wang. Ask the right questions: Active question reformulation with reinforcement learning. *arXiv preprint arXiv:1705.07830*, 2017. [82](#)
- [180] Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. Evidence aggregation for answer re-ranking in open-domain question answering. *arXiv preprint arXiv:1711.05116*, 2017. [83](#)
- [181] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. [83](#), [98](#), [150](#)
- [182] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027, 2016. [83](#)
- [183] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks

- from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.
- [184] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009. [90](#)
- [185] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017. [90, 94](#)
- [186] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015. [94](#)
- [187] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. [98, 186, 188](#)
- [188] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017. [102](#)
- [189] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, 2016. [102](#)
- [190] Shuai Zhang, Lina Yao, and Aixin Sun. Deep learning based recommender system: A survey and new perspectives. *arXiv preprint arXiv:1707.07435*, 2017. [105](#)
- [191] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. 2017.
- [192] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Latent relational metric learning via memory-based attention for collaborative ranking. In *Proceedings of the*

- 2018 World Wide Web Conference on World Wide Web, pages 729–739. International World Wide Web Conferences Steering Committee, 2018.
- [193] Shuai Zhang, Yi Tay, Lina Yao, and Aixin Sun. Next item recommendation with self-attention. *arXiv preprint arXiv:1808.06414*, 2018.
- [194] Shuai Zhang, Lina Yao, Aixin Sun, Sen Wang, Guodong Long, and Manqing Dong. Neurec: On nonlinear transformation for personalized ranking. *arXiv preprint arXiv:1805.03002*, 2018. [105](#)
- [195] Da Cao, Xiangnan He, Liqiang Nie, Xiaochi Wei, Xia Hu, Shunxiang Wu, and Tat-Seng Chua. Cross-platform app recommendation by jointly modeling ratings and texts. *ACM Transactions on Information Systems (TOIS)*, 35(4):37, 2017. [106](#)
- [196] Wei Zhang, Quan Yuan, Jiawei Han, and Jianyong Wang. Collaborative multi-level embedding learning from reviews for rating prediction. In *IJCAI*, pages 2986–2992, 2016.
- [197] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pages 1661–1670. ACM, 2015. [106](#)
- [198] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW ’17, pages 173–182, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-4913-0. [107](#), [116](#)
- [199] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. [111](#), [112](#)
- [200] Chris J Maddison, Daniel Tarlow, and Tom Minka. A* sampling. In *Advances in Neural Information Processing Systems*, pages 3086–3094, 2014. [111](#)
- [201] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. [112](#)

- [202] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1244. ACM, 2015. [117](#)
- [203] Shuohang Wang, Mo Yu, Shiyu Chang, and Jing Jiang. A co-matching model for multi-choice reading comprehension. *arXiv preprint arXiv:1806.04068*, 2018. [126](#)
- [204] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. *CoRR*, abs/1611.01576, 2016. [127](#), [128](#), [132](#), [144](#), [148](#), [152](#), [153](#)
- [205] Tao Lei and Yu Zhang. Training rnns as fast as cnns. *arXiv preprint arXiv:1709.02755*, 2017. [127](#), [128](#), [132](#), [144](#), [148](#), [153](#), [154](#)
- [206] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016. [128](#)
- [207] Felix Wu, Ni Lao, John Blitzer, Guandao Yang, and Kilian Weinberger. Fast reading comprehension with convnets. *arXiv preprint arXiv:1711.04352*, 2017. [128](#)
- [208] Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. Fast and accurate reading comprehension by combining self-attention and convolution. In *International Conference on Learning Representations*, 2018. URL <https://openreview.net/forum?id=B14T1G-RW>. [128](#)
- [209] Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. Coarse-to-fine question answering for long documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 209–220, 2017. [128](#)
- [210] Yichong Xu, Jingjing Liu, Jianfeng Gao, Yelong Shen, and Xiaodong Liu. Towards human-level machine reading comprehension: Reasoning and inference with multiple strategies. *arXiv preprint arXiv:1711.04964*, 2017. [129](#), [133](#), [136](#), [137](#), [138](#)

- [211] Danqi Chen, Jason Bolton, and Christopher D Manning. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*, 2016. [136](#), [138](#)
- [212] Bhuvan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*, 2016. [136](#), [138](#)
- [213] Soham Parikh, Ananya Sai, Preksha Nema, and Mitesh M Khapra. Eliminet: A model for eliminating options for reading comprehension with multiple choice questions, 2018. URL <https://openreview.net/forum?id=B1bgpzZAZ>. [138](#)
- [214] Salah El Hihi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in neural information processing systems*, pages 493–499, 1996. [143](#)
- [215] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013. [143](#)
- [216] 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. [143](#)
- [217] Yu Zhang, Guoguo Chen, Dong Yu, Kaisheng Yaco, Sanjeev Khudanpur, and James Glass. Highway long short-term memory rnns for distant speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 5755–5759. IEEE, 2016. [143](#)
- [218] Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. A clockwork rnn. *arXiv preprint arXiv:1402.3511*, 2014. [143](#)
- [219] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016. [143](#)
- [220] 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. [143](#)

- [221] Ivo Danihelka, Greg Wayne, Benigno Uria, Nal Kalchbrenner, and Alex Graves. Associative long short-term memory. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1986–1994, 2016. [143](#)
- [222] Shayne Longpre, Sabeek Pradhan, Caiming Xiong, and Richard Socher. A way out of the odyssey: Analyzing and combining recent insights for lstms. *arXiv preprint arXiv:1611.05104*, 2016. [143](#), [152](#), [153](#)
- [223] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. Disan: Directional self-attention network for rnn/cnn-free language understanding. *arXiv preprint arXiv:1709.04696*, 2017. [143](#), [144](#), [152](#), [153](#), [154](#)
- [224] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. Bi-directional block self-attention for fast and memory-efficient sequence modeling. *arXiv preprint arXiv:1804.00857*, 2018. [143](#), [144](#)
- [225] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*, 2016. [144](#), [152](#), [153](#)
- [226] Yue Zhang, Qi Liu, and Linfeng Song. Sentence-state lstm for text representation. *arXiv preprint arXiv:1805.02474*, 2018. [144](#), [151](#), [153](#)
- [227] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*, 2017. [149](#), [151](#)
- [228] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. Citeseer, 2013. [149](#), [189](#)
- [229] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics, 2011. [149](#)

- [230] Ellen M Voorhees et al. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82, 1999. [149](#)
- [231] Yi Yang, Wen-tau Yih, and Christopher Meek. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 2013–2018, 2015. [149](#), [166](#), [167](#)
- [232] Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. What is the jeopardy model? A quasi-synchronous grammar for QA. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pages 22–32, 2007. [149](#), [165](#), [167](#)
- [233] Hongyu Guo, Colin Cherry, and Jiang Su. End-to-end multi-view networks for text classification. *arXiv preprint arXiv:1704.05907*, 2017. [152](#)
- [234] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaaan 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. [152](#)
- [235] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*, 2015. [152](#)
- [236] Moshe Looks, Marcello Herreshoff, DeLesley Hutchins, and Peter Norvig. Deep learning with dynamic computation graphs. *arXiv preprint arXiv:1702.02181*, 2017. [152](#)
- [237] Minlie Huang, Qiao Qian, and Xiaoyan Zhu. Encoding syntactic knowledge in neural networks for sentiment classification. *ACM Transactions on Information Systems (TOIS)*, 35(3):26, 2017. [152](#)
- [238] Tsendsuren Munkhdalai and Hong Yu. Neural semantic encoders. *corr abs/1607.04315*, 2016. [152](#), [153](#)
- [239] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017. [152](#)

- [240] Rie Johnson and Tong Zhang. Supervised and semi-supervised text categorization using lstm for region embeddings. *arXiv preprint arXiv:1602.02373*, 2016. [152](#)
- [241] Adj B Dieng, Chong Wang, Jianfeng Gao, and John Paisley. Topicrnn: A recurrent neural network with long-range semantic dependency. *arXiv preprint arXiv:1611.01702*, 2016. [152](#)
- [242] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*, 2016. [152](#)
- [243] Rui Zhang, Honglak Lee, and Dragomir Radev. Dependency sensitive convolutional neural networks for modeling sentences and documents. *arXiv preprint arXiv:1611.02361*, 2016. [153](#)
- [244] Douwe Kiela, Changhan Wang, and Kyunghyun Cho. Context-attentive embeddings for improved sentence representations. *arXiv preprint arXiv:1804.07983*, 2018. [154](#)
- [245] Seonhoon Kim, Jin-Hyuk Hong, Inho Kang, and Nojun Kwak. Semantic sentence matching with densely-connected recurrent and co-attentive information. *arXiv preprint arXiv:1805.11360*, 2018. [154](#)
- [246] Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *CoRR*, abs/1705.08039, 2017. [159](#), [160](#), [163](#), [165](#)
- [247] Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. *Physical Review E*, 67(2):026112, 2003. [159](#)
- [248] Dmitri V. Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. Hyperbolic geometry of complex networks. *CoRR*, abs/1006.5169, 2010. [159](#), [160](#)
- [249] Kevin Verbeek and Subhash Suri. Metric embedding, hyperbolic space, and social networks. *Computational Geometry*, 59:1–12, 2016. [159](#)
- [250] Robert Kleinberg. Geographic routing using hyperbolic space. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications*,

- Joint Conference of the IEEE Computer and Communications Societies, 6-12 May 2007, Anchorage, Alaska, USA*, pages 1902–1909, 2007. doi: 10.1109/INFCOM.2007.221. [159](#)
- [251] Silvere Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Trans. Automat. Contr.*, 58(9):2217–2229, 2013. doi: 10.1109/TAC.2013.2254619. [165](#)
- [252] Yi Tay, Minh C. Phan, Luu Anh Tuan, and Siu Cheung Hui. Learning to rank question answer pairs with holographic dual lstm architecture. 2017. doi: 10.1145/3077136.3080790. [166](#), [167](#)
- [253] Michael Heilman and Noah A. Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*, pages 1011–1019, 2010. [167](#)
- [254] Aliaksei Severyn, Alessandro Moschitti, Manos Tsagkias, Richard Berendsen, and Maarten de Rijke. A syntax-aware re-ranker for microblog retrieval. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, pages 1067–1070, 2014. doi: 10.1145/2600428.2609511.
- [255] Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. Answer extraction as sequence tagging with tree edit distance. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 858–867, 2013. [167](#)
- [256] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014. [167](#)
- [257] Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. Deep learning for answer sentence selection. *CoRR*, abs/1412.1632, 2014. [167](#)
- [258] Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. Question answering using enhanced lexical semantic models. In *Proceedings*

- of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 1744–1753, 2013. [167](#)
- [259] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, pages 109–126, 1994. [167](#)
- [260] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2042–2050, 2014. [167](#)
- [261] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011. [168](#)
- [262] Laurens van der Maaten. Learning a parametric embedding by preserving local structure. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*, pages 384–391, 2009. [173](#)
- [263] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018. [178](#), [191](#)
- [264] Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, 2016. [178](#), [179](#), [188](#)
- [265] Dario Pavllo, David Grangier, and Michael Auli. Quaternet: A quaternion-based recurrent model for human motion. *arXiv preprint arXiv:1805.06485*, 2018. [179](#)
- [266] Chase Gaudet and Anthony Maida. Deep quaternion networks. *arXiv preprint arXiv:1712.04604*, 2017. [179](#)

- [267] Jiwei Li and Dan Jurafsky. Do multi-sense embeddings improve natural language understanding? *arXiv preprint arXiv:1506.01070*, 2015. [179](#)
- [268] Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654*, 2015.
- [269] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers- Volume 1*, pages 873–882. Association for Computational Linguistics, 2012. [179](#)
- [270] Titouan Parcollet, Mirco Ravanelli, Mohamed Mørchid, Georges Linarès, Chiheb Trabelsi, Renato De Mori, and Yoshua Bengio. Quaternion recurrent neural networks. *arXiv preprint arXiv:1806.04418*, 2018. [183](#), [187](#)
- [271] Tushar Khot, Ashish Sabharwal, and Peter Clark. Scitail: A textual entailment dataset from science question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. [188](#)
- [272] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017. [188](#)
- [273] Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. A continuously growing dataset of sentential paraphrases. *arXiv preprint arXiv:1708.00391*, 2017. [188](#)
- [274] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016. [188](#)
- [275] 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. [188](#)
- [276] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In

- Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/P11-1015>. 189
- [277] Artit Wangperawong. Attending to mathematical language with transformers. *CoRR*, abs/1812.02825, 2018. URL <http://arxiv.org/abs/1812.02825>. 191
- [278] Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535, 2016. 192
- [279] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018. 196
- [280] Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing datasets for multi-hop reading comprehension across documents. *arXiv preprint arXiv:1710.06481*, 2017. 196