Memory-Augmented Neural Networks for Better Discourse Understanding

Fei Liu

School of Computing and Information Systems

The University of Melbourne

Submitted in total fulfilment of the requirements of the degree of

Doctor of Philosophy

November 2019

Abstract

Natural language understanding is a long-standing goal of artificial intelligence and a vast number of approaches have been proposed. While much effort has been focused on word- or sentence-level phenomena, language often consists of collections of structured sentences, expressing a coherent message. Such groups of sentences are commonly known as discourse. The multi-sentence nature of discourse requires consideration of contexts beyond the sentence, making the application of conventional models difficult, especially those with limited memory capacity. Another demanding aspect of discourse understanding is the relationships between entities. Particularly, the prevalence of long range dependencies between distant entities in discourse necessitates the ability to not only store such entities stably over long timescales but also have access to and update the states of them regardless of the distance between each mention. Despite the successful application of recurrent neural network (RNNs) to a range of tasks, RNN-based models — in particular long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) — suffer from the locality bias of their memory components, highly influenced by immediately neighbouring elements, rendering them unstable over long timescales.

In this thesis, we focus on memory-augmented neural networks, a class of neural architectures with a global memory and easy access, both in terms of storing and updating knowledge, and show these models are well suited to a wide range of discourse tasks. We broadly classify memory-augmented models into two categories: static and dynamic memory networks and limit our focus to two models, one from each category: memory networks (MemN2Ns) (Weston et al., 2015, Sukhbaatar et al., 2015) and recurrent entity networks (EntNets) (Henaff et al., 2017). The three main research questions we

address in this thesis are: (1) How can we increase memory capacity in existing (static) memory architectures while not causing significant overfitting, thereby making them more generalisable to a wide range of discourse tasks? (2) How can we enhance modelling capacity by memory augmentation to better capture long-range contextual dependencies in discourse? (3) How can we relax limitations of (dynamic) memory networks to better capture entities and their interactions?

We address the first research question by introducing a unified weight tying mechanism to MEMN2Ns based on the observation that previous attempts to avoid overfitting by sharing model parameters often result in uneven performance with no clear winning strategy. This is implemented by letting the model dynamically choose the appropriate type of weight tying based on the input. The proposed model compares favourably to a collection of strong baselines over three discourse tasks.

Next, we propose to enhance modelling capacity by integrating external static memory into a widely popular machine learning model: conditional random fields (CRFs) (Lafferty et al., 2001). This is achieved by allowing the model to have access to the entire sequence via an attention mechanism, enabling it to look far beyond the immediately neighbouring elements and thereby capturing long-range contextual dependencies. Not only can the memory-enhanced model outperform an array of competitive baselines, it is also capable of generating predictions with improved consistency, further boosting its performance on a dialog state tracking task.

Lastly, we present two methods for relaxing limitations of dynamic memory networks, thereby improving their applicability to discourse tasks. While both build on ENTNETS, they focus on different aspects of discourse processing and are used in different settings and scenarios. The first method is concerned with fine-grained word-level information extraction where the difference in input granularity (words rather than sentences as originally designed) demands modelling of the delay in the activation of memory update. The second technique, on the other hand, considers the incorporation of semantic supervision into the training objective, allowing memory cells to have unambiguous distribution of responsibility, thereby reducing the amount of confusion caused by end-to-end training

only. Experimental results on two tasks verify the validity of the proposed approaches, superior to previous state of the art methods.

In summary, this thesis presents a number of methods for improving memory-augmented models to better understand discourse. Together, they demonstrate the utility of memory augmentation and serve as a stepping stone for future research.

Dedicated to my loving parents and grandparents.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 100,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Fei Liu

November 2019

Preface

Large portions of Chapter 3 have appeared in the following paper:

> Fei Liu, Trevor Cohn and Timothy Baldwin. 2017. Improving End-to-End Memory Networks with Unified Weight Tying. In *Proceedings of the Australasian Language Technology Association Workshop 2017 (ALTW 2017)*, pages 16–24, Brisbane, Australia.

Large portions of Chapter 4 have appeared in the following paper:

> Fei Liu, Timothy Baldwin and Trevor Cohn. 2017. Capturing Long-range Contextual Dependencies with Memory-enhanced Conditional Random Fields. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (IJCNLP 2017)*, pages 555–565, Taipei, Taiwan.

Large portions of Chapter 5 have appeared in the following papers:

> Fei Liu, Trevor Cohn and Timothy Baldwin. 2018. Recurrent Entity Networks with Delayed Memory Update for Targeted Aspect-based Sentiment Analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association of Computational Linguistics: Human Language Technologies (NAACL HLT 2018)*, pages 278–283, New Orleans, USA.
>
> Fei Liu, Trevor Cohn and Timothy Baldwin. 2018. Narrative Modeling with Memory Chains and Semantic Supervision. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 278–284, Melbourne, Australia.

The following papers, although the work of the author of this thesis, are either irrelevant to the main thesis topic or based on work carried out during internships outside the PhD candidature and therefore excluded from the thesis:

Fei Liu, Alistair Moffat, Timothy Baldwin and Xiuzhen Zhang. 2016. Quit While Ahead: Evaluating Truncated Rankings. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2016)*, pages 953–956, Pisa, Italy.

Fei Liu and Julien Perez. 2017. Gated End-to-End Memory Networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 1–10, Valencia, Spain.

Fei Liu, Julien Perez and Scott Nowson. 2017. A Language-independent and Compositional Model for Personality Trait Recognition from Short Texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 754–764, Valencia, Spain.

Fei Liu, Luke Zettlemoyer and Jacob Eisenstein. 2019. The Referential Reader: A Recurrent Entity Network for Anaphora Resolution. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pages 5918–5925, Florence, Italy.

Acknowledgements

I would like to express my sincere gratitude from the bottom of my heart to my supervisors Tim Baldwin and Trevor Cohn for taking me on as a PhD student, and more importantly, always keeping me on track throughout my candidature. Their invaluable feedback has always been the compass helping me navigate through uncharted waters. They have guided me with patience, support and encouragement, turning my PhD studies into an enjoyable journey. Tim's educational approach to supervision has left a profound mark on me, allowing me to take a step back to look at the big picture and better understand what makes good research. Trevor sets a perfect example with his remarkable enthusiasm for research, always coming up with a constant stream of new ideas, which, although overwhelming at times, has certainly been a stimulating experience. Working with them has been a true pleasure and I have benefited greatly from their mentoring. The lessons I have learned from them, be it academic or otherwise, will serve as a lifelong inspiration. I consider myself extremely lucky to have them as my supervisors, and more importantly, call them my friends.

Apart from my PhD studies, the university and my supervisors allowed me enormous freedom to work with and learn from many inspiring researchers in the field. I have had the great opportunity, with Xiuzhen (Jenny) Zhang and Alistair Moffat, to work on my first publication at a major conference, which provided much needed boost in confidence while I was desperately searching for my research topic in my first year. Two fruitful internships have also greatly broadened my knowledge, allowing me to build industry experience and connections and also see the field from a different perspective. They turned out to be two highly productive periods of my research career thus far. Among

many others who helped organise and provided support in these internship programs, I would like to thank Scott Nowson and Julien Perez at Xerox Research Centre Europe (now NAVER LABS Europe) and also Jacob Eisenstein, Luke Zettlemoyer, and Omer Levy at Facebook AI Research, for the challenging projects, stimulating discussions, and, equally importantly, fun times in Grenoble, France and Seattle, WA. In particular, I am very appreciative of the mentoring from Jacob Eisenstein, with whom I worked closely during my second internship. Jacob pushed me to be my best self and collaborating with him has been a thought-provoking and rewarding experience.

I feel incredibly fortunate and proud to be part of an amazing and talented NLP research group. A special thank you to all colleague students, post-docs, and faculty members for the great company, support and fun: Li Wang, Ned Letcher, Jey Han Lau, Huizhi Liang, Joel Nothman, Zilu Liang, Julian Brooke, Meng Fang, Daniel Beck, Matthias Petri, Dat Quoc Nguyen, Afshin Rahimi, Pingping Tan, Bahar Salehi, Miji Choi, Qingyu Chen, Wenxi Wang, Seyed Mohammad Hossein Oloomi, Oliver Adams, Ekaterina Vylomova, Weihao Cheng, Diego de Uña, Zsolt Bitvai, Michal Lukasik, Steven Xu, Yuan Li, Long Duong, Doris Hoogeveen, Andrew Bennett, Vu (Cong Duy) Hoang, Yitong Li, Nitika Mathur, Aili Shen, Shivashankar Subramanian, Wei Wu, Ziad Al Bkhetan, Anirudh Joshi, Shraey Bhatia, Dalin Wang, Tatsuya Aoki, Brian Hur, Minghao Wu, Fajri Koto, Haonan Li, Zenan Zhai, Biaoyan Fang, Yuxia Wang, Kemal Kurniawan, Jinghui Liu, Chunhua Liu, Xudong Han. Thanks to them, I never felt lonely in my long PhD journey.

Indebted to many organisations who offered to cover the costs of my PhD studies and provided living allowance throughout the candidature, I would like to gratefully acknowledge the financial support of the Australian Government Research Training Program Scholarship, the Conference Travel Scholarship from the School of Engineering and School of Computing and Information Systems, and the PhD Travel Scholarship from Google.

Last but not least, I am extremely grateful to my family, in particular, my partner and parents. I could not have made it this far without the love and support from them. Words are powerless to express my dearest gratitude to their unselfish sacrifice and unconditional love.

Notation

This chapter provides a brief reference describing the mathematical notation used throughout this thesis (unless stated otherwise). We largely follow the conventions defined in the book of Goodfellow et al. (2016) and outline the most frequently used ones below:

a, a scalar

$\boldsymbol{a}$, a column vector

a_i, i-th element of vector $\boldsymbol{a}$

$\boldsymbol{a}^\top$, transpose of vector $\boldsymbol{a}$

$\boldsymbol{a} \cdot \boldsymbol{b}$, dot product of $\boldsymbol{a}$ and $\boldsymbol{b}$ (same as $\boldsymbol{a}^\top \boldsymbol{b}$)

$\boldsymbol{a} \odot \boldsymbol{b}$, element-wise (Hadamard) product of $\boldsymbol{a}$ and $\boldsymbol{b}$

$\|\boldsymbol{a}\|$, the Euclidean norm of vector $\boldsymbol{a}$

$\boldsymbol{A}$, a matrix

$A_{i,j}$, element i, j of matrix $\boldsymbol{A}$ (i-th row, j-th column)

$\boldsymbol{A}_{i,:}$, i-th row of matrix $\boldsymbol{A}$

$\boldsymbol{A}_{:,j}$, j-th column of matrix $\boldsymbol{A}$

$\boldsymbol{I}$, identity matrix

$\boldsymbol{A}^\top$, transpose of matrix $\boldsymbol{A}$

$\boldsymbol{A} \odot \boldsymbol{B}$, element-wise (Hadamard) product of $\boldsymbol{A}$ and $\boldsymbol{B}$

$\|\boldsymbol{A}\|$, the Euclidean norm of matrix $\boldsymbol{A}$

$\sigma(x) = \dfrac{1}{1+e^{-x}}$, the sigmoid activation function, applied element-wise in the case of a vector or matrix input

$\tanh(x) = \dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$, the hyperbolic tangent function, applied element-wise in the case of a vector or matrix input

$\mathrm{ReLU}(x) = \max(0, x)$, the rectifier function, applied element-wise in the case of a vector or matrix input

$\mathrm{PReLU}(x) = \max(0, x) + \alpha \min(0, x)$, the parametric rectifier function (He et al., 2015), applied element-wise in the case of a vector or matrix input, α is a trainable parameter

$\mathrm{softmax}(\boldsymbol{x}) = \dfrac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$, the softmax function and $\boldsymbol{x} \in \mathbb{R}^K$

Table of contents

List of figures **xxiii**

List of tables **xxv**

1 Introduction **1**

1.1 Background and Motivation 3

1.2 Aim and Research Questions 7

1.3 Scope 9

1.4 Thesis Structure 11

2 Literature Review **13**

2.1 Why and What is Memory? 14

2.1.1 The Psychological View 14

2.1.2 The Neuroscience View 15

2.2 Early Studies on Memory Augmentation 16

2.2.1 Recurrent Neural Networks 17

2.2.2 Attention 22

2.2.3 Pointer Networks 25

2.3 Static Memory Networks 26

2.3.1 Memory Networks 26

2.3.1.1 Framework Definition 27

2.3.1.2 Application to Text Domain 28

2.3.1.3 Limitations of Memory Networks 31

2.3.2 End-to-End Memory Networks 31
2.3.2.1 Single-hop End-to-End Memory Networks 32
2.3.2.2 Multi-hop End-to-End Memory Networks 34
2.3.2.3 Weight Tying on Embedding Weight Matrices 36
2.3.2.4 Positional and Temporal Encoding 37
2.3.2.5 End-to-End Training 38
2.3.2.6 What Can MEMN2N Learn? 38
2.3.2.7 Variants of MEMN2N 40
2.3.2.8 Comparision with RNN-based Models and MEMNN 41
2.3.3 Non-recurrent Self-Attention-based Sequence Processing 41
2.3.3.1 Model Formulation 42
2.3.3.2 What can Self-Attention Learn? 44
2.3.3.3 Variants of TRANSFORMERS 45
2.3.3.4 Comparison with RNN-based Models and MEMN2N 45
2.4 Dynamic Memory Networks 46
2.4.1 Data-driven Algorithm Learning 47
2.4.1.1 Neural Turing Machine 47
2.4.1.2 Differentiable Neural Computer 50
2.4.1.3 Comparison with Static Memory Networks 52
2.4.2 Recurrent Entity Networks 52
2.4.2.1 Related Entity-Centric Models 53
2.4.2.2 Motivation 54
2.4.2.3 Memory Architecture 55
2.4.2.4 Final Classifier 57
2.4.2.5 Motivating Example 59
2.4.2.6 Memory Chain Configuration 60
2.4.2.7 Comparison with Other Types of Memory Networks 61
2.4.3 Stack-based Memory 62
2.4.3.1 Comparison with Other Types of Memory Networks 64

2.4.4 Summary of Memory-augmented Models 64
2.5 NLP Applications of Memory-augmented Models 67
2.5.1 Machine Translation 67
2.5.2 Question Answering 68
2.5.3 Text Categorisation 70
2.6 Narrative and Discourse Comprehension 71
2.6.1 Narrative and Story Understanding 72
2.6.1.1 Related Narrative Tasks 72
2.6.1.2 Reasoning-focused Story Understanding 75
2.6.1.3 Commonsense Narrative Understanding 77
2.6.2 Discourse Comprehension 79
2.6.2.1 Related Discourse Tasks 79
2.6.2.2 Dialogue 79
2.6.2.3 Web Forum Thread Discourse Structure 82
2.7 Summary 85

3 Improving Memory Capacity with Unified Weight Tying 87
3.1 Motivation for Unified Weight Tying 88
3.2 Related Memory-augmented Models 94
3.3 Memory Networks with Unified Weight Tying 95
3.4 Experiments on Discourse Tasks 98
3.4.1 Question Answering bAbI Experiments 99
3.4.1.1 Experimental Setup 99
3.4.1.2 Results on bAbI 100
3.4.2 Dialog bAbI Experiments 102
3.4.2.1 Experimental Setup 102
3.4.2.2 Results on Dialog bAbI 104
3.4.3 Dialog State Tracking Experiments 106
3.4.3.1 Experimental Setup 106
3.4.3.2 Results on DSTC-2 108

3.4.4 Analysis and Model Visualisation . . . 109
3.5 Summary . . . 110

4 Enhancing Modelling Capacity with Static Memory Networks 113
4.1 Motivation for Memory-enhanced Conditional Random Fields . . . 114
4.2 Related Work . . . 117
4.2.1 Conditional Random Fields (CRFs) . . . 117
4.2.2 Recurrent Neural Networks (RNNs) . . . 118
4.2.3 Memory Networks (MEMN2Ns) . . . 118
4.3 Memory-enhanced Conditional Random Fields . . . 119
4.3.1 Memory Layer . . . 120
4.3.1.1 Input Memory . . . 120
4.3.1.2 Current Input . . . 121
4.3.1.3 Attention . . . 121
4.3.1.4 Output Memory . . . 121
4.3.1.5 Memory Layer Output . . . 122
4.3.1.6 Possible Extension to Incorporate Multi-hop Reasoning . . . 122
4.3.2 CRF Layer . . . 122
4.4 Experiments on Discourse Tasks . . . 124
4.4.1 Discourse-aware Named Entity Recognition . . . 124
4.4.1.1 Dataset and Task . . . 124
4.4.1.2 Experimental Setup . . . 125
4.4.1.3 Evaluation . . . 126
4.4.1.4 Results . . . 126
4.4.1.5 Visualisation . . . 127
4.4.2 Thread Discourse Structure Prediction . . . 129
4.4.2.1 Dataset and Task . . . 129
4.4.2.2 Experimental Setup . . . 130
4.4.2.3 Evaluation . . . 132
4.4.2.4 Results . . . 132

4.4.2.5 Analysis . . . 134
4.4.3 Dialog State Tracking (DSTC-2) . . . 136
4.4.3.1 Dataset and Task . . . 136
4.4.3.2 Experimental Setup . . . 137
4.4.3.3 Evaluation . . . 139
4.4.3.4 Results . . . 139
4.4.3.5 Analysis and Discussion . . . 140
4.5 Summary . . . 143

5 Relaxing Limitations of Dynamic Memory Chains 145

5.1 Brief Overview of ENTNET . . . 147
5.1.1 Motivating Example . . . 150
5.2 Delayed Memory Update for Targeted Aspect-based Sentiment Analysis . . 151
5.2.1 Motivation . . . 153
5.2.2 Task Description . . . 155
5.2.3 Delayed Memory Update . . . 155
5.2.4 Final Classifier . . . 158
5.2.5 Comparison with ENTNET . . . 158
5.2.6 Experiments on Sentihood . . . 159
5.2.6.1 Dataset . . . 159
5.2.6.2 Model Configuration . . . 159
5.2.6.3 Evaluation . . . 160
5.2.6.4 Results . . . 161
5.2.6.5 Discussion . . . 162
5.2.6.6 Sensitivity Study . . . 163
5.2.6.7 Subsequent Studies . . . 163
5.2.7 Summary . . . 164
5.3 Narrative Modelling with Memory Chains and Semantic Supervision 165
5.3.1 Motivation . . . 166
5.3.2 Task Description . . . 168

5.3.3 Semantically Motivated Memory Chains . . . 168
5.3.4 Sentence-level Representation . . . 171
5.3.5 Bi-directionality . . . 171
5.3.6 Final Classifier . . . 171
5.3.7 Training Loss . . . 172
5.3.8 Why Not Delayed Memory Update? . . . 172
5.3.9 Experiments on ROC Story Cloze Test . . . 173
5.3.9.1 Dataset . . . 173
5.3.9.2 Model Configuration . . . 173
5.3.9.3 Evaluation . . . 174
5.3.9.4 Results and Discussion . . . 174
5.3.9.5 Discussion . . . 175
5.3.9.6 Ablation Study . . . 176
5.3.9.7 Subsequent Studies . . . 177
5.4 Summary . . . 178

6 Conclusions **181**
6.1 Research Question Revisited . . . 182
6.2 Future Directions . . . 184
6.2.1 Dynamic Entity Representation . . . 184
6.2.2 Incorporating External Knowledge . . . 186
6.2.3 Identifying Connections with Cognitive Science . . . 187
6.3 Summary . . . 188

References **189**

List of figures

1.1 A NER example with long-range contextual dependencies. 5

2.1 Illustration of a MemN2N with a single memory hop. 32
2.2 Illustration of a MemN2N with multiple memory hops. 34
2.3 Illustration of an instance of EntNet with n memory chains. 58
2.4 Illustration of an EntNet recurrent unit, depicting the inner workings of a shaded recurrent cell in Figure 2.3. 58
2.5 Examples of bAbI tasks. 76
2.6 An example of the web forum thread discourse structure prediction task. . . 86

3.1 Illustration of a MemN2N with a single memory hop, reproduced from Figure 2.1. 90
3.2 Illustration of a MemN2N with multiple memory hops, reproduced from Figure 2.2. 91
3.3 Illustration of the two types of weight tying mechanisms based on a MemN2N model with 3 hops. 93
3.4 PCA scatter plot of the gating vectors $\boldsymbol{z}$ over bAbI tasks 3, 16, 17 and 19. . . . 109

4.1 A NER example with long-range contextual dependencies, reproduced from Figure 1.1. 114
4.2 An example of the web forum thread discourse structure prediction task, reproduced from Figure 2.6. 116

4.3 Illustration of ME-CRFs with a single memory hop, showing the network architecture at time step t and $t+1$. 120
4.4 NER examples showing learned attention to long-range contextual dependencies. 128
4.5 Breakdown of post-level Joint F-scores by post depth. 134
4.6 Breakdown of post-level Link and DA F-scores by post depth. 135

5.1 Illustration of an instance of ENTNET with n memory chains, reproduced from Figure 2.3. 148
5.2 Illustration of an ENTNET recurrent unit, depicting the inner workings of a shaded recurrent cell in Figure 5.1. Figure reproduced from Figure 2.4. . . . 148
5.3 Targeted aspect-based sentiment analysis examples. 152
5.4 Illustration of our model with a single memory chain at time i. 156
5.5 Example of the gate value g_i averaged across memory chains, forward and backward, in ENTNET without delay vs. our model (with delay). 162
5.6 Sensitivity study of model performance to # of memory chains n. 163
5.7 ROC Story Cloze Test examples. 166
5.8 An example of ENTNET with semantic supervision. 170
5.9 t-SNE scatter plot of aspect-triggering words (output representations $\boldsymbol{h}_i$ by BI-GRU. 176

List of tables

1.1 An example triple of story, question and answer, extracted from the bAbI dataset (Weston et al., 2016). 2

2.1 An example triple of story, question and answer, extracted from the bAbI dataset (Weston et al., 2016), reproduced from Table 1.1. 26
2.2 An example of the attention weights of a 3-hop MEMN2N trained on task 3 (3 supporting facts) of the bAbI dataset. 39
2.3 Comparison of memory-augmented models described in Chapter 2. 65
2.4 Examples of the Story Cloze Test. 77
2.5 An example of DSTC-2. 81
2.6 An example of the tasks in Dialog bAbI. 83

3.1 An example triple of story, question and answer, extracted from the bAbI dataset (Weston et al., 2016), reproduced from Table 1.1. 89
3.2 Accuracy (%) reported in Sukhbaatar et al. (2015) on a selected subset of the 20 bAbI 10k tasks. 94
3.3 Notation summary. 95
3.4 Accuracy (%) on the 20 bAbI 10k tasks for our proposed method (UMEMN2N) and various benchmark methods. 101
3.5 Per-response accuracy on the Dialog bAbI tasks. 105
3.6 Average accuracy (%) of various models on the DSTC-2 dataset over 5 runs. 109
3.7 Accuracy (%) reported in Sukhbaatar et al. (2015) on bAbI tasks 3, 16, 17, and 19. 110

4.1 NER performance on the CoNLL 2003 English NER shared task dataset. . . . 127
4.2 Post-level Link and DA F-scores. 133
4.3 Average accuracy (%) of various models on the DSTC-2 dataset over 5 runs. 139
4.4 Average discourse-level accuracy (%) on the DSTC-2 dataset over 5 runs. . . 141
4.5 Four examples of ME-CRF + LW + BI-GRU vs. ME-CRF. 142

5.1 An example triple of story, question and answer, extracted from the bAbI dataset (Weston et al., 2016). Table reproduced from Table 1.1. 151
5.2 An example of the TABSA task. 154
5.3 Performance on SENTIHOOD. 161
5.4 Performance of subsequent studies on SENTIHOOD. 164
5.5 An example of the linguistically motivated memory chain supervision binary labels. 169
5.6 Performance on the Story Cloze test set. 175
5.7 Ablation study. Average accuracy over 5 runs on the Story Cloze test set (all subject to the same model selection criterion as our model). 177
5.8 An example of the linguistically motivated memory chain supervision binary labels, reproduced from Figure 5.5. 177
5.9 Performance comparison against subsequent studies by Radford et al. (2019) and Sun et al. (2019b). 178

Chapter 1

Introduction

Natural language understanding (NLU) has been a long-standing goal of artificial intelligence and a vast number of approaches have been proposed. While many have focused on word- or sentence-level language phenomena, language is commonly more than just individual and isolated sentences, but rather structured into complex narratives (Jurafsky and Martin, 2009). Such multi-sentence units are often collectively known as discourse. The vast diversity of characterisations of discourse structure makes it difficult to find a universally competent method, rendering discourse understanding a challenging problem (Eisenstein, 2018). In the scope of this thesis, we focus on two particular aspects of discourse. The larger context in a discourse, as opposed to that in a single sentence, necessitates the ability to capture dependencies between arbitrarily distant elements, often beyond sentence boundaries. Another demanding aspect of discourse understanding is that sentences often describe multiple entities and their interactions. This makes tracking the dynamics of the states of those entities challenging, possibly requiring one to collect evidence or even perform reasoning over a collection of supporting facts. An example of this is shown in Table 1.1 with further description in Section 1.1.

While many have proposed the use of Recurrent Neural Networks (RNNs), RNN-based models — in particular in the form of the long short-term memory (Hochreiter and Schmidhuber, 1997) (LSTM), for a range of tasks (Atkeson and Schaal, 1995, Graves, 2013, Koutnik et al., 2014, Mikolov et al., 2015) — the memory component in such architectures

#	Story
1	Jeff went to the kitchen.
2	Mary travelled to the hallway.
3	Jeff picked up the milk.
4	Jeff travelled to the bedroom.
5	Jeff left the milk.
6	Jeff went to the bathroom.

Question	Answer
Where is the milk now?	bedroom
Where was Jeff before the bedroom?	kitchen

Table 1.1 An example triple of story, question and answer, extracted from the bAbI dataset (Weston et al., 2016). Key pieces of evidence for the first question are underlined, with distractors marked with dashed underlining.

can easily be influenced by immediately neighbouring elements and is therefore unstable over long timescales. As such, this family of models is incapable of learning long-range dependencies (Lai et al., 2015, Sukhbaatar et al., 2015, Linzen et al., 2016, Wang et al., 2017) (see Section 1.1) and is infeasible for high-level discourse tasks, such as text-based reasoning and commonsense story understanding.

Another line of research, focusing on utilising explicit storage and accessing it via attention, has offered a viable alternative to the challenges identified above. Such storage, often referred to as "memory", is represented in a continuous space and can be read or written to, along with other operations, all of which can be modelled as actions of a neural network working as the memory controller.

This thesis deals with memory-augmented neural networks, which refer to a class of neural architecture equipped with global memory for storing and accessing knowledge, thereby allowing the model to consider larger contexts, capture and learn long-range dependencies, and track the dynamics of entities, three key elements crucial to the success of a range of natural language processing (NLP) tasks. We show that such techniques can be applied to a wide range of discourse tasks, such as dialog state tracking, question answering, and commonsense story understanding.

In this chapter, we briefly describe the background and motivate the necessity of memory-augmentation in the context of NLP in this era of deep learning in Section 1.1. Next, we present the aim and specify the scope of this thesis and then list the contributions of our work in Section 1.2. Lastly, we finish the chapter by outlining the thesis structure in Section 1.4.

1.1 Background and Motivation

Deep neural network models (often referred to as *deep learning*) have demonstrated strong performance over a number of challenging tasks, ranging from image classification (He et al., 2015, Huang et al., 2017), to speech recognition (Graves et al., 2013), to drug molecule activity prediction (Ma et al., 2015), to even analysing particle accelerator data (Ciodaro et al., 2012, Kaggle, 2014). More recently, deep learning has been applied to various natural language processing tasks, such as text categorisation (Kim, 2014), text-based reasoning (Weston et al., 2015, Sukhbaatar et al., 2015), question answering (Wang et al., 2017, Yu et al., 2018), and machine translation (Bahdanau et al., 2015, Luong et al., 2015, Vaswani et al., 2017). The wide success of such deep neural models can largely be ascribed to their ability to learn progressively higher levels of abstractions from raw input (LeCun et al., 2015). In the context of NLP, such models are highly capable of capturing representations of text with different granularities (e.g., characters, words, sentences, and ultimately discourse), making them applicable to a wide array of tasks.

Among many alternative network architectures, Feedforward Neural Networks (FFNNs, or multi-layer perceptrons or MLPs), Convolutional Neural Networks (LeCun, 1989) (CNNs), and Recurrent Neural Networks (Rumelhart et al., 1986b) (RNNs) are the three most widely used models. Each of them is applicable to a certain type of problem. MLPs are perhaps the quintessential deep learning model and can be used to approximate arbitrary functions. While CNNs apply convolutional filters to extract local features and gradually build up higher levels of representations, RNNs are specialised for processing sequential input, such as a sentence consisting of a sequence of words.

Although deep learning has made major advances in the field of NLP, it is still rather limited in its ability to consider larger contexts, capture long-range dependencies, and track the states and model the interactions of entities, three crucial aspects in language processing. While humans perform such tasks naturally, sometimes without even knowingly doing so, deep learning models struggle to understand the relationships between entities. To make it even more difficult, many tasks require a model to keep track of entities and update their states, requiring long-term stable storage of information regarding the states of entities. Plain RNNs are unable to capture long-range dependencies (Bengio et al., 1994, Hochreiter et al., 2001) and variants such as LSTMs (Hochreiter and Schmidhuber, 1997), although more capabable of capturing non-local patterns in theory, still exhibit a significant locality bias in practice (Lai et al., 2015, Linzen et al., 2016, Wang et al., 2017). That is, LSTMs are biased towards closer elements and do not readily generalise to long-range dependencies. The ineffectiveness is mainly due to the lack of an external memory component, leading to the inability to store data stably over long timescales (Graves et al., 2016).

The concept of working memory has been thoroughly studied in many fields. From a psychological perspective, memory refers to a block of information which can be readily accessed and easily recalled (Miller, 1956) and research has extensively focused on studying the capacity and limitations of it, leading towards an understanding of how humans construct the structure of their working memory systems (Graves et al., 2014).

From a neuroscience perspective, the concept of working memory can often be interpreted as the activations (or "firing") of either a single neuron or a group of neurons in prefrontal cortex and basal ganglia (Goldman-Rakic, 1995). Research is primarily concerned with recording and observing patterns of activations of neurons while the subject is performing a certain type of task.

From a computer science perspective, in the context of computer architecture and data storage, memory is often decoupled from computation, with the processor responsible for fetching operands via their addresses in the memory. This has two major benefits: (1) it allows the use of extensible memory storage; and (2) it supports the representation of

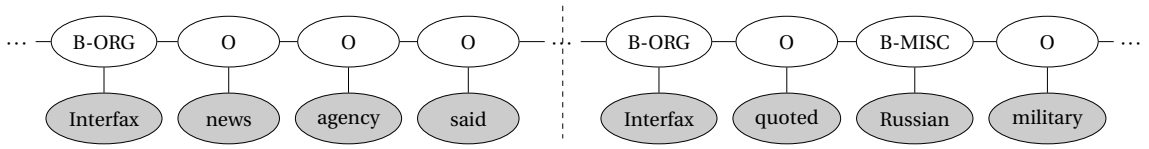

Figure 1.1 A NER example with long-range contextual dependencies. The vertical dash line indicates a sentence boundary.

memory contents as variables, thereby allowing algorithms to generalise (Graves et al., 2014).

In Figure 1.1, we present an example to demonstrate the prevalent occurrence of long-range contextual dependencies in NLP, as well as the importance and benefits of capturing them. In this example, we are interested in the identification of individual token occurrences of named entities as well as their semantic class (e.g.., `LOCATION` or `ORGANISATION`), a task commonly known as named entity recognition (NER). While it is easy to detect that *Interfax* in the second sentence is a named entity, it is much harder to determine its semantic class, as there is little contextual information. The usage in the first sentence, on the other hand, can be reliably disambiguated due to the post-modifying phase *news agency*. Ideally, we would like to be able to share such contexts across all usages (and variants) of a given named entity for reliable and consistent identification and disambiguation. This requires the modelling of long-range contextual dependencies to incorporate information far beyond neighbouring steps, which, in turn, requires external working memory since the memory in typical sequence labelling models is inherently limited to place more focus on adjacent items than distant ones, and unstable over long timescales (Sukhbaatar et al., 2015, Lai et al., 2015, Linzen et al., 2016, Seo et al., 2017).

Consider the textual reasoning example in Table 1.1, where a story consisting of a sequence of supporting facts is presented, alongside a collection of questions. The goal of this task is to find answers to such questions by keeping track of the states of entities mentioned in the story and reasoning over the collection of supporting statements. For the first question, it is easy for humans to work out the solution by tracking the state of the entity *milk*: from *in the kitchen*, to *carried by Jeff*, to *placed in the bedroom*. However,

despite the conciseness of the story, making a computer understand the narrative and capable of answering questions based on reasoning has been shown to be of great difficulty (Weston et al., 2015, 2016, Welbl et al., 2018). Had the example been processed by an RNN-based model, the final prediction may have been highly influenced by the last supporting statement, i.e., sentence #6, likely resulting in an incorrect answer, i.e., *milk in bathroom*. This issue is even more pronounced in the case of the second question, where, to be able to answer it, a model needs to go as far back as to the beginning of the story, a scenario where RNN-based models are most likely to fail.

Not only does the locality bias issue cause RNN-based models to be rather "short-sighted", the lack of an external memory component also greatly constrains the ability of a model to consider larger context. Again, we take the first question in Table 1.1 as an example. In order to find out where the *milk* is, one needs to first locate *milk* in sentence #3 (picked up by *Jeff*). Based on this, the focus should then be shifted to *Jeff* and tracking his whereabouts in sentence #4 (travelled to the *bedroom*). Lastly, sentence #5, along with information carried by previous sentences, makes it clear that the *milk* is now in the bedroom. Obviously, this question requires multiple passes of the same story, gathering new evidence each pass and shifting focus based on new clues. This requires the consideration of larger context and can be seen as multi-hop reasoning, a task difficult for RNN-based models to perform. This is further evidenced by the rather ineffective performance of LSTMs on this task reported in the work of Sukhbaatar et al. (2015), motivating the necessity of integrating an external memory.

Note that while memory-augmentation is similar to memory-based learning (also known as instance-based learning, e.g., k-nearest neighbour (Cover and Hart, 1967)) in that both of them store all (labelled) instances, they differ in how the memory components and stored information are utilised. With memory-based learning, information stored in the memory is only used for locating the closest neighbours, measured with some distance/similarity metrics, and then predicting based on the labels of such neighbours. With memory-augmentation, on the other hand, a neural network module, often referred to as the memory controller, learns to read and write to memory components, thereby

allowing it to incorporate and update knowledge stored in the memory to support a variety of NLP tasks, such as text-based reasoning and commonsense story understanding.

1.2 Aim and Research Questions

The primary aim of this thesis is to improve memory-augmented neural networks and show that such models can be applied to a range of discourse-related NLP tasks, most of which require consideration of larger contexts beyond a single sentence and modelling interactions between multiple entities. In terms of memory augmentation, we classify models into two groups: static and dynamic memory networks, according to whether the memory is actively and dynamically updated as new information arrives, but focus on two specific neural network models, one in each category: (1) memory networks (MEMNETS) (Weston et al., 2015, Sukhbaatar et al., 2015); and (2) recurrent entity networks (ENTNETS) (Henaff et al., 2017). While the idea of MEMNETS can be thought of as a static memory mechanism, ENTNETS, in contrast, underline the importance of dynamic updates, allowing the memory to evolve upon seeing new input.

We approach the aim of the thesis from the following three perspectives.

1. How can we increase memory capacity in existing (static) memory architectures while not causing significant overfitting, thereby making them more generalisable to a wide range of discourse tasks? Memory networks were originally designed for a syntactically generated dataset, making a model with limited memory capacity sufficient for such a toy dataset with unrealistic assumptions. However, this results in model configurations with uneven performance on some tasks and no clear winning strategy, hindering the application of such models to many discourse tasks. We focus on increasing memory capacity by bridging gaps in existing memory architectures and, in doing so, making (static) memory networks applicable to a diverse set of discourse tasks.

   This is a key limitation of static memory networks as they heavily rely on attention to guide the model to focus on relevant pieces of information stored in the memory.

Due to the restricted memory capacity, the attention mechanism may fail to identify the most relevant clues, resulting in inferior model performance. In contrast, the more recently introduced TRANSFORMER-based models (Vaswani et al., 2017) make use of a more sophisticated multi-head attention mechanism based on queries, and incorporating keys and values, enabling them to have more modelling capacity and therefore better performance on a wide range of tasks. Different from TRANSFORMERS, we answer this research question in Chapter 3 by trying to strike a balance between two existing weight tying methods, thereby unlocking the full potential of a static memory network without causing significant overfitting.

2. How can we enhance modelling capacity by memory augmentation to better capture long-range contextual dependencies in discourse? Observing the promising results of memory networks, we next turn our attention to traditional machine learning methods and consider ways of integrating external memory to such models, thereby enhancing modelling capacity and enabling them to capture long-range contextual dependencies, which are prevalent in discourse. Despite the common occurrence of long-range dependencies in discourse, popular sequence processing models, such as RNNs, in particular, LSTMs, struggle to capture such relationships between distant elements (Sukhbaatar et al., 2015, Lai et al., 2015, Linzen et al., 2016, Seo et al., 2017). Motivated by this, we are interested in enhancing the modelling capacity of conventional sequence processing models with memory augmentation. More specifically, the inability to capture dependencies over distance is mainly caused by the rather limited memory capacity of RNN-based models. Knowledge regarding previous steps is forced into a single dense vector, which may be inadequate to store the vast amount of information required by long-range dependencies. With the help of memory augmentation and attention, we are no longer confined by this restriction, enabling us to seek a better alternative to tackling long-range dependencies, a frequently occurring phenomenon in discourse.

3. How can we relax limitations of (dynamic) memory networks to better capture entities and their interactions?

   Entities form an important aspect of discourse, often involved in the sentences of a discourse, describing the interactions between them. Modelling entities in a discourse is a challenging task, requiring one to identify and keep track of the dynamics of such entities throughout the narrative, focusing on how they are referenced and interact with one another (Eisenstein, 2018). While many discourse models were originally designed for question answering tasks, placing more emphasis on coarse-grained input (e.g., sentences), modelling the dynamics of entities and the interactions between them often requires fine-grained word-level processing. Failing to address such changes in input granularity may result in incompatibility and inferior performance. We investigate methods to improve the applicability and performance of (dynamic) memory networks on such tasks.

1.3 Scope

The outcomes of the thesis are improved algorithms which improve over prior art at various discourse-related tasks. In terms of testing and evaluating the effectiveness of the proposed models, we focus on the following tasks:

1. bAbI, a dataset featuring 20 different natural language-based reasoning tasks in the form of: (1) a list of supporting statements; (2) a question; and (3) the answer, typically a single word or short phrase. Each task in bAbI is syntactically generated with a distinct emphasis on a specific type of reasoning. In order to predict the desired answer, a model is required to locate (or focus on) the relevant context sentences among irrelevant distractors in the memory.

2. Dialog bAbI, a collection of goal-oriented dialog tasks, all in a restaurant reservation scenario, developed by Weston et al. (2016), consisting of 6 categories each with a specific focus on testing one aspect of an end-to-end dialog system. Starting with a

request from the user, a dialog proceeds with subsequent and alternating user-agent utterances. The agent (or system) needs to figure out the user intention and answer (or react) accordingly.

3. DSTC-2, a dialog state tracking task in the form of slot filling, set in the scenario of restaurant booking (Henderson et al., 2013). Successful models are required to keep track of value changes in three slots: food, price range and area.

4. Named Entity Recognition (NER), the goal of which is to identify individual token occurrences of named entities (NEs), and tag each with its class (e.g. LOCATION or ORGANISATION).

5. Thread discourse structure analysis, given a list of posts all within the same thread on a forum, we attempt to predict which post(s) a given post directly responds to, and in what way(s) (as captured by dialogue acts).

6. Targeted aspect-based sentiment analysis, extraction of fine-grained opinion polarity with respect to a pre-defined set of aspects. Specifically, it is the task of identifying fine-grained opinion polarity towards a specific aspect associated with a given target.

7. Commonsense story understanding, given a story consisting of a sequence of context sentences, we are interested in predicting the coherent ending to the story out of two alternative ending options.

While demonstrating strong performance, some better than previous state of the art, a number of the proposed novel neural network-based models achieve so at a lower cost in terms of training data requirements, highlighting the efficiency of such memory-augmented models.

1.4 Thesis Structure

Chapter 2 reviews the literature on memory-augmented models, focusing on neural network-based models and their application to the text domain. We broadly classify such memory-augmented models into two categories: static and dynamic memory networks, depending on whether their memory components are mainly kept unchanged throughout the processing of each instance (static) or directly updated (written to) upon seeing relevant input (dynamic). We pay particular attention to two models, one in each category: (end-to-end) memory networks (MEMN2Ns) (Weston et al., 2015, Sukhbaatar et al., 2015) and recurrent entity networks (ENTNETS) (Henaff et al., 2017), both of which are critical to this thesis as they establish a firm base for us to build on. Additionally, we outline a number of discourse datasets, focusing on those relevant to this thesis, ones that we use to test the effectiveness of the methods proposed in Chapters 3, 4 and 5.

In Chapter 3, we present a unified weight tying mechanism (UMEMN2N) to bridge the gap between existing attempts to ameliorate overfitting by allowing model parameters in MEMN2Ns to be shared across memory hops, resulting in uneven performance with no clear winning strategy. In contrast, with UMEMN2N, we let the model choose the appropriate type of weight tying based on the input. The proposed method achieves uniformly high performance, outperforming strong baselines on a range of discourse tasks. Further analysis reveals that UMEMN2N indeed favours the more suitable type of weight tying, confirming our hypothesis.

In Chapter 4, we propose a novel method for capturing long-range dependencies which are prevalent in discourse. More specifically, we integrate external neural memory to a widely used sequence labelling model: conditional random fields (CRFs) (Lafferty et al., 2001), thereby allowing each step to have access to the entire sequence via attention as opposed to being heavily influenced by immediately neighbouring elements as in a recurrent neural network (RNN), resulting in significant locality bias (Lai et al., 2015, Linzen et al., 2016). The addition of CRFs also enables static memory networks to generate predictions with improved consistency by taking into account the transitions between

labels. Evaluated on a diverse list of discourse tasks, the proposed method outperforms a number of strong baselines, demonstrating the utility of memory integration.

In Chapter 5, we turn our attention to dynamic memory networks, in particular, ENTNETs, and present two methods, both aimed at widening the applicability of the model and making it better suited to discourse tasks. The first method addresses the issue of the difference in input granularity: word for fine-grained word-level information extraction as opposed to sentence-level input units originally designed for coarse-grained question answering tasks. The second method incorporates semantic supervision into the training objective to encourage each memory to focus on a specific aspect of discourse, making the distribution of responsibility unambiguous among memory chains. They both substantially improve model performance on discourse tasks, advancing state of the art.[1] Further analysis provides insights as to how these proposed methods help boost accuracy, allowing us to better understand the source of the performance gains.

Finally, Chapter 6 summarises the findings of the thesis and discusses avenues for future work.

[1] At the time of publication.

Chapter 2

Literature Review

This chapter reviews the literature on memory-augmented models in the context of neural networks and narrative and discourse comprehension. We start with a brief overview of the studies on memory in psychology and neuroscience in Section 2.1. Next, we review early studies on memory augmentation and a number of key concepts in this regard in Section 2.2. Then, we discuss two types of memory-augmented models, static and dynamic memory networks, and detail crucial components and representative models in each category, along with their application to the text domain, in Sections 2.3 and 2.4. This is then followed by an overview of the applications of memory-augmented models to NLP tasks in Section 2.5. Lastly, we present tasks and datasets relevant to the topic of narrative and discourse comprehension in Section 2.6.

It should be noted that the scope of mathematical notation is limited to its own section describing a particular model and that some may be overloaded across different subsections. The boundaries, while not explicitly specified, should be self-evident and straightforward to identify given the context.

2.1 Why and What is Memory?

The first attempt to understand human memory can be traced as far back as Aristotle (Baddeley et al., 2015). Although not reaching any conclusions, it forms one classic philosophical question and highlights the importance of the study of memory.

In the machine learning literature, the term "memory" is often vaguely defined, commonly referring to knowledge stored in the form of dense continuous vectors or matrices as well as the means to access them (mostly via attention). In this section, to motivate discussion in the rest of the chapter, we briefly review the concept of memory from the psychological and neuroscience views, bridging the gap between different disciplines, thereby better connecting them to the machine learning community.

2.1.1 The Psychological View

From the psychological perspective, evidence suggests that there is not a single, simple, global memory but rather a complex system involving many types of memory (Baddeley et al., 2015). While it remains controversial as to how many types of memory there are, we follow Baddeley et al. (2015) in classifying memory into three categories: sensory memory, short-term memory and long-term memory. This view is similar to the modal model (Atkinson and Shiffrin, 1968), which was widely accepted in the 1960s, but differs in that it does not assume the unidirectional information flow from sensory memory to short-term memory and then to long-term memory.

Of particular interest to this thesis and the deep learning community in general is the concept of short-term memory, which, in turn, is closed related to "working memory". While the former refers to the temporary storage of information over short periods time, mostly in a matter of several seconds, the latter is mainly concerned with the systematic way of managing and manipulating information so that it can be helpful in the process of performing complex tasks, or more plainly "keep things in mind" much like a mental sketch pad (Baddeley et al., 2015).

There has been some debate over whether memory should be perceived as the storage of information or the processes that operate on it (Nairne, 1990, 2002, Neath and Surprenant, 2003). The modern view is to take both into account. This is analogous to the mechanism in most memory-augmented models in that information is encoded and stored in its representational form, and later accessed or updated via attention.

2.1.2 The Neuroscience View

As the study of memory has developed, it has become increasingly evident that parallels can be drawn in regards to concepts, methods and findings between the psychological and neuroscience views of memory (Baddeley et al., 2015). The difference is that the neuroscience view of memory is more concerned with studying the activations of neurons in the prefrontal cortex and their patterns of activation.

The term working memory is generally used to describe the type of memory that stays relevant only for a short period of time, often a matter of seconds. The significance of working memory was not fully appreciated until recently, mostly because it is perceived to be less important due to its transient nature, as opposed to the more permanent archival nature of long-term memory. The human brain's working memory function should perhaps be considered "its evolutionarily most significant achievement" (Goldman-Rakic, 1995). The link between working memory and neurons was first identified in studies of human cognition (Norman, 1970, Baddeley, 1986). This is further supported by the findings of Goldman-Rakic (1987) who demonstrated strong connections between working memory and the prefrontal cortex. With the help of modern neurobiological methods, the activations of neurons can now be narrowed down to a specific region of neurons to provide in-depth analysis of the patterns of their circuitry at the molecule level (Goldman-Rakic, 1995).

Rather than "out of sight, out of mind", working memory is the bridge connecting past experience and present actions, providing temporal and spatial continuity (Goldman-Rakic, 1987). This essentially allows one to predict an outcome based on prior experience, similar to memory augmentation in the context of neural networks, and plays an important

role in the comprehension and construction of sentences as well as many other tasks, such as mental arithmetic and playing chess.

2.2 Early Studies on Memory Augmentation

Early work on memory augmentation can be traced to as far back as the 90s. Das et al. (1992) propose to learn Deterministic Context-free (DCF) grammars with a neural network, augmented with stack memory, named Recurrent Neural Network Pushdown Automaton (NNPDA). This approach relies on stack memory and allows the model to perform stack-related actions: push, pop or no-op so that the model, once trained, is capable of leveraging external memory via these actions to infer a DCF.

Schmidhuber (1992) introduces Fast-Weight Memories, an architecture consisting of two feedforward neural networks for sequential learning problems typically reserved for RNNs. The two feedforward neural networks used in this approach differ in the rate they change their internal weights, with one much faster than the other, hence the name "fast-weight" memories. The key idea is to learn to manipulate the weights of the fast network with the slow one. Here, the slow network can be thought of as the "memory controller" whereas the fast one maintains the actual short-term memory, and can preserve information over arbitrary periods of time unless explicitly modified by the controller.

Later work explored the idea of a "self-referential" weight matrix, the addressing mechanism of which is similar to memory addressing, however it is only presented as a thought experiment to show the theoretical possibility (Schmidhuber, 1993).

While paving the way for future studies on memory augmentation, they either come at the expense of high computational cost (Schmidhuber, 1993) or tend to focus on rather small-scale or toy problems with limited memory size (Das et al., 1992, Schmidhuber, 1992), a limitation likely caused by the hardware capacity at the time.

In this section, we review a number of key concepts in regards to memory augmentation, which all play a crucial role in the development of the idea of explicitly leveraging external neural memory to help perform complex tasks.

2.2.1 Recurrent Neural Networks

Perhaps one of the most widely used neural network architectures with memory augmentation, especially in the field of NLP, is Recurrent Neural Networks (RNNs), which take advantage of the idea of parameter sharing where the same set of parameter weight matrices is shared across different parts of a model. In the context of NLP, because of the variable-length nature of the input (e.g., word sequences, sentences), parameter sharing is mostly applied to the temporal axis, that is, the same set of weight matrices is used repeatedly to process the input sequence at different time steps. Without parameter sharing, a model would have to create new parameter matrices upon seeing a new input, making the number of trainable parameters grow linearly with the length of the input and the model infeasible to generalise to sequence lengths unseen at training time.

Because of the invariant nature of RNNs at every step, they are forced to learn to filter out information deemed irrelevant and remember important parts with the help of their "hidden states", which can be seen as the memory component. In fact, improved versions of RNNs explicitly frame this as their "memory cells", which will be discussed later in this section.

Given an input sequence $\mathbf{x}$ consisting of $|\mathbf{x}|$ vectorial elements: $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{|\mathbf{x}|}\}$, a typical RNN takes the form of:

$$\boldsymbol{h}_t = f(\boldsymbol{W}\boldsymbol{h}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{b}) \tag{2.1}$$

$$\boldsymbol{o}_t = \boldsymbol{V}\boldsymbol{h}_t + \boldsymbol{c} \tag{2.2}$$

where $\boldsymbol{h}_t$ and $\boldsymbol{o}_t$ are the current hidden state and output at time t, $\boldsymbol{W}$, $\boldsymbol{U}$, and $\boldsymbol{V}$ are trainable parameter matrices, $\boldsymbol{b}$ and $\boldsymbol{c}$ are bias terms, and f is a non-linear activation function (most commonly, the sigmoid σ or hyperbolic tangent function tanh). For brevity,

an RNN (Equation (2.1)) is sometimes simplified to:

$$\boldsymbol{h}_t = \text{RNN}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t). \tag{2.3}$$

In some variants of the plain RNN, the output at time t is identical to the hidden state $\boldsymbol{o}_t = \boldsymbol{h}_t$ with no linear transformation applied.

With this formulation, when trained to perform a task to predict the future based on past steps, an RNN normally learns to push relevant information from previous steps into the hidden state $\boldsymbol{h}_t$ and carry it over multiple time steps, allowing signals seen earlier in the sequence $\mathbf{x}$ to influence the output at a much later step, forming the "memory" of the RNN. Depending on the nature of the task and training criterion, an RNN may selectively choose to store some aspects, deemed to be of particular importance to the task, in its memory, leaving irrelevant bits out to reduce noise.

One issue with RNNs is that the runtime is $\mathcal{O}(T)$, linear to the length of the sequence and cannot be parallelised due to the dependency of the current hidden state $\boldsymbol{h}_t$ on that of the previous step $\boldsymbol{h}_{t-1}$. Another, and perhaps even more serious, issue is the problem of vanishing and exploding gradient (Hochreiter, 1991, Bengio et al., 1993, 1994). This is caused by repeatedly multiplying the same weight matrix $\boldsymbol{W}$ in Equation (2.1). Returning to the recurrent connection in Equation (2.1), the partial derivative of $\boldsymbol{h}_t$ with respect to $\boldsymbol{h}_{t-k}$ is:

$$\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-k}} = \prod_{t-k<i\leqslant t} \frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{h}_{i-1}} = \prod_{t-k<i\leqslant t} \boldsymbol{W}^\top \text{diag}(f'(\boldsymbol{h}_{i-1})) \tag{2.4}$$

taking the form of a product of k matrices. For a matrix with the eigendecomposition $\boldsymbol{A} = \boldsymbol{V}\text{diag}(\boldsymbol{\lambda})\boldsymbol{V}^{-1}$, multiplying the same matrix k times results in:

$$\boldsymbol{A}^k = (\boldsymbol{V}\text{diag}(\boldsymbol{\lambda})\boldsymbol{V}^{-1})^k = \boldsymbol{V}\text{diag}(\boldsymbol{\lambda})^k\boldsymbol{V}^{-1}. \tag{2.5}$$

If there is much difference between any eigenvalue λ_i and 1, then it will, through the above multiplication, end up exploding in magnitude or decaying to 0. However, for robustness against small perturbations, RNNs are mostly trained to stay in the space likely causing

the gradients to vanish (Bengio et al., 1993, 1994). Also, consider the derivative of the non-linear activation function in Equation (2.1), either the sigmoid $\sigma(x)$ or hyperbolic tangent $\tanh(x)$ function:

$$\frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x)), \qquad \frac{d}{dx}\tanh(x) = 1-\tanh^2(x). \tag{2.6}$$

While both are always positive, the derivative of $\tanh(x)$ is bounded to be $\leqslant 1$ and $\frac{d}{dx}\sigma(x)$ reaches a maximum of 0.25 when $\sigma(x) = 0.5$, making gradients more likely to vanish.

In cases where capturing long-range dependencies is a priority, gradients may become exponentially smaller due to the interactions among distant elements in a sequence than those of nearby steps, making learning difficult. Bengio et al. (1994) show that RNNs tend to fail to capture such long-range dependencies once the span is greater than 10 or 20.

To remedy the issue of vanishing and exploding gradient, Hochreiter and Schmidhuber (1997) propose the Long Short-Term Memory (LSTM) model with a self-connection link between "memory cells" at neighbouring steps. Formally, an LSTM recurrent unit is defined as:

$$\tilde{\boldsymbol{c}}_t = \sigma(\boldsymbol{W}_c \boldsymbol{h}_{t-1} + \boldsymbol{U}_c \boldsymbol{x}_t + \boldsymbol{b}_c) \tag{2.7}$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f \boldsymbol{h}_{t-1} + \boldsymbol{U}_f \boldsymbol{x}_t + \boldsymbol{b}_f) \tag{2.8}$$

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i \boldsymbol{h}_{t-1} + \boldsymbol{U}_i \boldsymbol{x}_t + \boldsymbol{b}_i) \tag{2.9}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o \boldsymbol{h}_{t-1} + \boldsymbol{U}_o \boldsymbol{x}_t + \boldsymbol{b}_o) \tag{2.10}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \tag{2.11}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \tag{2.12}$$

where $\odot$ denotes the Hadamard product, $\boldsymbol{W}_*$ and $\boldsymbol{U}_*$ are trainable weight matrices, and $\boldsymbol{b}_*$ are bias terms. Input, output and recurrent connections are all controlled by a number of "gates": (1) input gate $\boldsymbol{i}_t$; (2) output gate $\boldsymbol{o}_t$; and (3) forget gate $\boldsymbol{f}_t$. Of particular interest to the vanishing gradients problem is the forget gate $\boldsymbol{f}_t$ which controls the self-connection link by taking a value in the range of $(0,1)$. Information is now added, as opposed to

multiplied as in Equation (2.11), to the memory cells in Equation (2.11), which works as a "shortcut" and helps the gradients flow over long distances, thereby allowing the model to capture long-range dependencies to some extent (Levy et al., 2018). For the exploding gradients problem, a simple yet effective way to deal with it is to use gradient clipping (Mikolov, 2012, Pascanu et al., 2013).

A similar, yet simpler variant named Gated Recurrent Units is proposed by Cho et al. (2014b):

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r \boldsymbol{h}_{t-1} + \boldsymbol{U}_r \boldsymbol{x}_t + \boldsymbol{b}_r) \tag{2.13}$$

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_z \boldsymbol{h}_{t-1} + \boldsymbol{U}_z \boldsymbol{x}_t + \boldsymbol{b}_z) \tag{2.14}$$

$$\tilde{\boldsymbol{h}}_t = \sigma(\boldsymbol{W}_h (\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}) + \boldsymbol{U}_h \boldsymbol{x}_t + \boldsymbol{b}_h) \tag{2.15}$$

$$\boldsymbol{h}_t = \boldsymbol{z}_t \odot \boldsymbol{h}_t + (\mathbf{1} - \boldsymbol{z}_t) \odot \tilde{\boldsymbol{h}}_t \tag{2.16}$$

where $\boldsymbol{W}_*$ and $\boldsymbol{U}_*$ are trainable weight matrices, and $\boldsymbol{b}_*$ are bias terms. This formulation simplifies the LSTM model by relying on a single gating unit to control the forget and update operations to the memory, and achieves equally competitive performance on a number of tasks (Chung et al., 2014).

Despite the theoretical claim that LSTMs and GRUs are capable of capturing relationships among distant elements, numerous studies have demonstrated the strong locality bias of such models in practice, rendering them incompetent when coping with long-range dependencies (Lai et al., 2015, Linzen et al., 2016). In particular, Linzen et al. (2016) show that, on a verb-subject agreement task, the prediction output of an LSTM model is often influenced by the noun closest to the verb and that error rates increase sharply, almost by an order of magnitude, if the last intervening noun is not of the same number as the true subject to which the verb attaches.

Mathematically, Levy et al. (2018) show that the current memory cell in an LSTM, $\boldsymbol{c}_t$, can be seen as an element-wise weighted sum of all previous memory cells:

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \tag{2.17}$$

$$= \sum_{j=0}^{t} \left(\boldsymbol{i}_j \odot \prod_{k=j+1}^{t} \boldsymbol{f}_k \right) \odot \tilde{\boldsymbol{c}}_j \tag{2.18}$$

$$= \sum_{j=0}^{t} \boldsymbol{w}_j^t \odot \tilde{\boldsymbol{c}}_j \tag{2.19}$$

where the weighting term $\boldsymbol{w}_j^t = \boldsymbol{i}_t \odot \prod_{k=j+1}^{t} \boldsymbol{f}_k$ is the product of the input gate $\boldsymbol{i}_j$ and all subsequent forget gates $\boldsymbol{f}_k$. Here, $\boldsymbol{i}_0 = \mathbf{1}$ and $\tilde{\boldsymbol{c}}_0$ initialises the memory cell $\boldsymbol{c}$. Similarly, for GRUs, the hidden state $\boldsymbol{h}_t$ can also be expanded in terms of previous hidden states:

$$\boldsymbol{h}_t = \boldsymbol{z}_t \odot \boldsymbol{h}_{t-1} + (\mathbf{1} - \boldsymbol{z}_t) \odot \tilde{\boldsymbol{h}}_t \tag{2.20}$$

$$= \sum_{j=0}^{t} \left((\mathbf{1} - \boldsymbol{z}_j) \odot \prod_{k=j+1}^{t} \boldsymbol{z}_k \right) \odot \tilde{\boldsymbol{h}}_j \tag{2.21}$$

$$= \sum_{j=0}^{t} \boldsymbol{w}_j^t \odot \tilde{\boldsymbol{h}}_j \tag{2.22}$$

where the weighting term $\boldsymbol{w}_j^t$ is computed analogously to its LSTM counterpart.

The above derivations suggest that, while it is easier for LSTMs and GRUs to backpropagate through time because of the additive memory cells as opposed to the multiplicative ones in RNNs (causing exponential decay or explosion), the distance as to how far gradients can flow back is now influenced by other multiplicative terms, namely the weighting terms $\boldsymbol{w}_j^t$, which can in turn be represented by a pair of gates (input $\boldsymbol{i}_j$ and forget $\boldsymbol{f}_k$ gates in the case of LSTM and their GRU counterparts $\mathbf{1} - \boldsymbol{z}_j$ and $\boldsymbol{z}_k$). Note that all the gates in an LSTM or GRU are bounded to be $\in [0, 1]$. In practice, if any of the values of such gates, forget gates in particular, are less than 1, then the flow of gradients may become seriously restricted for future steps to have any impact during backpropagation, thereby preventing the model from capturing long-range dependencies.

2.2.2 Attention

Attention, in the context of neural networks, is another concept closed related to memory. As identified by Baddeley et al. (2015), an attentional controller, also known as the "central executive", rather than a memory system, is assumed to be responsible for directing the working memory.

Sequence learning is among the first attempts at applying attention to NLP problems (Sutskever et al., 2014, Cho et al., 2014a,b), in particular machine translation because of the sequence-to-sequence nature of the task. A typical model for machine translation consists of an encoder and a decoder, both most commonly based on LSTMs. The encoder is responsible for processing the source sentence whereas the decoder generates the output of the predicted target sentence. Together, they model the probability of the output target sequence $\mathbf{y} = \{y_1, \ldots, y_{|\mathbf{y}|}\}$ given the input source sequence $\mathbf{x} = \{x_1, \ldots, x_{|\mathbf{x}|}\}$:

$$p(y_1, \ldots, y_{|\mathbf{y}|}) = \prod_{t=1}^{|\mathbf{y}|} p(y_t | y_1, \ldots, y_{t-1}, \boldsymbol{c}) \tag{2.23}$$

where $|\mathbf{x}|$ and $|\mathbf{y}|$ indicate the sequence length of $\mathbf{x}$ and $\mathbf{y}$ respectively, and $\boldsymbol{c}$ is a context vector from the encoder to represent the input source sequence. Depending on the model design, the context vector can take different forms, varying in complexity, with the simplest setting being the hidden state of the final step of the encoder, that is, $\boldsymbol{c} = \boldsymbol{h}_{|\mathbf{x}|}$. In fact, Sutskever et al. (2014) propose the use of two deep, 4-layer stacked LSTMs, one as encoder and the other as decoder, with the decoder LSTM taking as input the final hidden state of the encoder, to estimate the conditional probability in Equation (2.23). One key finding in this work of Sutskever et al. (2014) is that the performance of the model on machine translation can be substantially improved by reversing the input source sequence ($\{x_1, x_2, \ldots, x_{|\mathbf{x}|-1}, x_{|\mathbf{x}|}\}$ becomes $\{x_{|\mathbf{x}|}, x_{|\mathbf{x}|-1}, \ldots, x_2, x_1\}$, the target sequence remains intact) so as to reduce long term dependencies at the beginning of a setence pair. Moreover, rapid performance degradation can be observed as the length of input sequences increases (Cho et al., 2014a). This provides further evidence that RNN-based

models are better at preserving key information over short periods of time rather than over distant elements.

One potential problem with this architecture is that knowledge regarding the source sequence has to be compressed into a fixed-length vector, which may not have enough capacity to store the amount of information required to perform decoding well on the target side, making it particularly difficult for sentences longer than those in the training set (Cho et al., 2014a, Bahdanau et al., 2015).

Observing this, Bahdanau et al. (2015) introduce RNNSearch, an encoder-decoder architecture with an attention mechanism, lifting the restriction of relying on a single, fixed-length vector, regardless of the input length, to pass on the encoding of the input to the decoder, allowing the decoder to have unconstrained access to the entire input sequence. Specifically, the encoder in RNNSearch is comprised of a bi-directional RNN, scanning the input sentence both forwards and backward:

$$\overrightarrow{\boldsymbol{h}}_t = \overrightarrow{\text{GRU}}_x(\overrightarrow{\boldsymbol{h}}_{t-1}, \boldsymbol{x}_t) \tag{2.24}$$

$$\overleftarrow{\boldsymbol{h}}_t = \overleftarrow{\text{GRU}}_x(\overleftarrow{\boldsymbol{h}}_{t+1}, \boldsymbol{x}_t) \tag{2.25}$$

$$\boldsymbol{h}_t = \left[\overrightarrow{\boldsymbol{h}}_{t-1}^\top; \overleftarrow{\boldsymbol{h}}_t^\top\right]^\top \tag{2.26}$$

where GRU_x denotes the processing of an GRU unit at a single step with the overhead arrow indicating processing direction (each direction takes a distinctive set of trainable weight matrices), $\left[\boldsymbol{a}^\top; \boldsymbol{b}^\top\right]$ is the concatenation of the two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, and $\boldsymbol{x}_t$ is the embedding of x_t. Note that the $\overrightarrow{\text{GRU}}$ and $\overleftarrow{\text{GRU}}$ are the same as in Section 2.2.1 with the exception of Equation (2.15) where the non-linear activation function of σ is now replaced with tanh:

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_h(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}) + \boldsymbol{U}_h \boldsymbol{x}_t + \boldsymbol{b}_h). \tag{2.27}$$

The decoder, on the other hand, is uni-directional and takes the form of a GRU unit to generate the output prediction of the target sequence:

$$\boldsymbol{s}_t = \text{GRU}_y(\boldsymbol{s}_{t-1}, y_{t-1}, \boldsymbol{c}_t) \tag{2.28}$$

where $\boldsymbol{s}_{t-1}$ is the previous hidden state, y_{t-1} is the predicted target output at the previous step, and $\boldsymbol{c}_t$ is the attention-based contextualised representation of the input sequence. The context vector $\boldsymbol{c}_t$ is the result of an attention-weighted sum of the representations of the input sequence:

$$\boldsymbol{c}_t = \sum_{i=1}^{|\mathbf{x}|} \alpha_{ti} \boldsymbol{h}_i \tag{2.29}$$

where the attention value α_{ti} is in turn defined as:

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^{|\mathbf{x}|} \exp(e_{tj})} \tag{2.30}$$

$$e_{ti} = a(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i) \tag{2.31}$$

where α_{ti} is the score of the model's belief of how much x_i should align to (or rather translate into) y_t, and the function a is jointly trained to learn the unnormalised soft alignment of the source to the target, parameterised by a feedforward neural network. This way, RNNSearch is trained to learn alignment and translation jointly. Note that, here, the GRU_y takes a slightly different form than presented in Section 2.2.1. When computing $\tilde{\boldsymbol{s}}_t$, with the addition of $\boldsymbol{y}_{t-1}$, GRU_y now becomes:

$$\tilde{\boldsymbol{s}}_t = \tanh(\boldsymbol{W}_s(\boldsymbol{r}_t \odot \boldsymbol{s}_{t-1}) + \boldsymbol{W}_y y_{t-1} + \boldsymbol{C}\boldsymbol{c}_t) \tag{2.32}$$

where $\boldsymbol{W}_s$, $\boldsymbol{W}_y$ and $\boldsymbol{C}$ are all trainable weight matrices, in particular $\boldsymbol{W}_y$ is the embedding matrix mapping words in the target language into their dense vectorial representations.

Subsequent work of Xu et al. (2015) demonstrates the power of attention on the task of image caption generation with visualised analysis showing that the model is trained to attend to a specific region of an image highly relevant to the decision of which next word to generate.

This paves the way for the adoption of attention in NLP. In memory-augmented models, attention is primarily used as to make a model to focus on a select few elements, according to their relevance, among many candidates stored in the memory.

Note that while we have only described a few forms of attention, those most relevant to NLP problems, many types exist in other literature. For a more comprehensive overview of attention, see the works of Hu (2018) and Chaudhari et al. (2019).

2.2.3 Pointer Networks

A closely related topic to attention is Point Networks (PTRNETS) introduced by Vinyals et al. (2015) to address problems with a varying number of output classes (e.g., the number of output classes may depend on the length of the input sequence). Specifically, Vinyals et al. (2015) propose to regard attention as a pointer, identifying a member of the input as the output. For a pair $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{|(x)|}\}$ and $\mathbf{y} = \{y_1, \ldots, y_{|\mathbf{y}|}\}$, a PTRNET estimates the conditional probability:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{|\mathbf{y}|} p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) \tag{2.33}$$

where the term on the right is modelled with an LSTM and attention over the input sequence:

$$u_{ij} = \boldsymbol{v}^\top \tanh(\boldsymbol{W}\boldsymbol{h}_{i-1} + \boldsymbol{U}\boldsymbol{x}_j), \quad \text{for } j \in \left[1, |\mathbf{x}|\right] \tag{2.34}$$

$$p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = \frac{\exp(u_{ij})}{\sum_{k=1}^{|\mathbf{x}|} \exp(u_{ik})} \tag{2.35}$$

where $\boldsymbol{h}_{i-1}$ is the LSTM hidden state at the $(i-1)$-th step, and $\boldsymbol{W}$, $\boldsymbol{U}$ and $\boldsymbol{v}$ are trainable weights. With this formulation, the distribution output of PTRNET at every step (Equation (2.35)) is not over a pre-defined set of classes, but rather depends on the size of the input sequence $\mathbf{x}$.

Perhaps one of the most notable applications of PTRNETs to the field of NLP is the work of Gu et al. (2016), in which the decoder in a sequence-to-sequence framework can not only generate words but also point to and copy certain input subsequences, making them part of the output sequence.

#	Story
1	Jeff went to the kitchen.
2	Mary travelled to the hallway.
3	Jeff picked up the milk.
4	Jeff travelled to the bedroom.
5	Jeff left the milk.
6	Jeff went to the bathroom.

Question	Answer
Where is the milk now?	bedroom
Where was Jeff before the bedroom?	kitchen

Table 2.1 An example triple of story, question and answer, extracted from the bAbI dataset (Weston et al., 2016). Key pieces of evidence for the first question are underlined, with distractors marked with dashed underline. This table is reproduced from Table 1.1

In Section 4, we will demonstrate how we can leverage the concept of PTRNETs in training Memory-enhanced Conditional Random Fields.

2.3 Static Memory Networks

While there are many ways to categorise memory augmented models into different families depending on the perspective, here we simply classify them into two categories: static and dynamic memory networks with the former only extracting (reading) information from the memory and not explicitly updating (writing to) its contents, and the latter capable of both reading and writing to the memory.

In this section, we focus on two influential pieces of work, namely Memory Networks (Weston et al., 2015) and its cousin End-to-End Memory Networks (Sukhbaatar et al., 2015), both of which can be perceived as static memory networks.

2.3.1 Memory Networks

Perhaps the first in recent years to explore the idea of integrating external memory components into neural network-based models is the effort by Weston et al. (2015) on Memory Networks (MEMNETS), a new class of learning models, specifically designed to cope

with reasoning and inference tasks requiring long-term memory storage. For a natural language-based reasoning problem, such as the one shown in Table 2.1, although it is possible to train an RNN-based language model to predict the next couple of words based on what the model has read previously, the "memory" in such models is often not capable of encoding the vast amount of information carried by the support statements in the story. The problem is particularly severe on tasks requiring memorisation over long sequences, as shown by Zaremba and Sutskever (2014) that even with curriculum learning the performance on a simple copying task is not perfect and deteriorates as the length of the sequence increases.

2.3.1.1 Framework Definition

Weston et al. (2015) define MEMNETs to be a framework consisting of four major components, along with a memory matrix $\boldsymbol{M}$. Here, we use the example shown in Table 2.1 to facilitate discussion:

1. I: the input module responsible for converting incoming data into its feature representation, for example, encoding the support statements s_i in the story with $I(s_i)$ and the questions q at the bottom with $I(q)$; this can be as simple as transforming the input into its dense representation (e.g., word embeddings) or as sophisticated as also including other types of pre-processing (e.g., parsing, coreference resolution);

2. G: the generalisation module updating the memory based on information carried by the new input: $\boldsymbol{m}_i = G(\boldsymbol{m}_i, I(x), \boldsymbol{M})$ where, in the context of the example, $\boldsymbol{M}$ stores the encoding of the entire story with $\boldsymbol{m}_i$ being the encoding of the i-th support statement; a more complex memory indexing mechanism, either carefully designed or trained, can also be explored to deal with a large-scale memory bank where G may be limited to a select number of memory locations and ignore the rest, allowing efficient retrieval and update of memory;

3. O: the output module in charge of constructing the output feature representation based on the current input and and its relevance to each of the memory locations: $o = O(I(x), \boldsymbol{M})$;

4. R: the response module converting the output feature representation into the desired response format: $\hat{r} = R(o)$; this transforms o into either a distribution or an array of scores $\hat{\boldsymbol{r}}$ over all possible answers and then selects the argmax as the answer, in the example in Table 2.1, a single word in the vocabulary.

2.3.1.2 Application to Text Domain

Based on the MEMNET framework, Weston et al. (2015) go on to specify a particular implementation of Memory Neural Networks (MEMNNs) for the text domain. Again, we use Table 2.1 as a motivating example to facilitate discussion. Here, we formalise the task as one of narrative comprehension with the input coming in the form of a tuple: $(\{\mathbf{s}_1, \ldots, \mathbf{s}_m\}, \mathbf{q})$ where m is the total number of sentences in the story, $\mathbf{s}_i$ is the i-th sentence, and $\mathbf{q}$ is the question. Each sentence or question is in turn a sequence of words such that $\mathbf{s}_i = \{s_{i1}, \ldots, s_{i|\mathbf{s}_i|}\}$ and $\mathbf{q} = \{q_1, \ldots, q_{|\mathbf{q}|}\}$. Additionally, associated with each training example is a true answer a, typically a single word in the vocabulary V.

The input module I first converts the incoming message to its internal representation, which can be as simple as a bag of words representation:

$$I(\mathbf{s}_i) = \sum_{j=1}^{|\mathbf{s}_i|} \Phi(s_{ij}) \tag{2.36}$$

$$I(\mathbf{q}) = \sum_{j=1}^{|\mathbf{q}|} \Phi(q_j) \tag{2.37}$$

where Φ is a function mapping individual input words into a one-hot representation. This results in an n-hot vector of vocabulary size $|V|$, with 1 at the i-th location in the vector indicating the occurrence of the i-th word in the vocabulary V, and 0 otherwise.

The generalisation module G then simply stores the sentences into the memory bank, one support statement at each location:

$$\boldsymbol{m}_i = G(\boldsymbol{m}_i, I(\mathbf{s}_i), \boldsymbol{M}) = I(\mathbf{s}_i). \tag{2.38}$$

Note that with this formulation, old memories are not updated upon seeing new input data. While more sophisticated memory updating mechanisms can be explored, for simplicity, we keep the memory bank unaltered for now.

The output module O is crucial for performing inference, which is based on identifying the k most important clues among the given support statements with a scoring function. While this procedure can be generalised to include multiple supporting memory locations with much larger k values, for illustration purposes, we pick $k = 2$. The first step of the output module is to find the most relevant memory location given the input, which can be the question:

$$o_1 = O_1(\mathbf{q}, \boldsymbol{M}) = \underset{i=1,\ldots,m}{\arg\max}\, s_O(I(\mathbf{q}), \boldsymbol{m}_i) \tag{2.39}$$

where s_O is the scoring function measuring the relevance, and m is the total number of memory locations. Once the output module has identified the first supporting memory candidate, it moves on to locating the second candidate by taking into account what it has found so far:

$$o_2 = O_2(\mathbf{q}, \boldsymbol{M}, \boldsymbol{m}_{o_1}) = \underset{i=1,\ldots,m}{\arg\max}\, s_O([I(\mathbf{q}), \boldsymbol{m}_{o_1}], \boldsymbol{m}_i). \tag{2.40}$$

For $k = 2$, the output of this module is $o = [\mathbf{q}, \boldsymbol{m}_{o_1}, \boldsymbol{m}_{o_2}]$.

Lastly, the response module R determines the output, that is, in the case of the motivating example, the answer to the question q:

$$\hat{r} = \underset{w \in V}{\arg\max}\, \hat{\boldsymbol{r}} = \underset{w \in V}{\arg\max}\, s_R([\mathbf{q}, \boldsymbol{m}_{o_1}, \boldsymbol{m}_{o_2}], w) \tag{2.41}$$

where w is the answer candidate and V is the vocabulary.

A key component in this implementation is the scoring functions s_O and s_R. Suppose that the input to these two functions is already processed by the input module I, which

may take the form of converting the input to a bag of words representation, that is, mapping each word in a sequence into its index, resulting in an n-hot vector for an input sequence. To measure the relevance of the two inputs to s_O and s_R, we simply transform them into their embedding space and compute the similarity with a dot product of the two:

$$s(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{U}^\top \boldsymbol{U} \boldsymbol{y} \tag{2.42}$$

where $\boldsymbol{U}$ is a trainable embedding weight matrix. Note that the $\boldsymbol{U}$ is not shared across s_O and s_R, but rather each function has its own embedding weight matrix. Also note that while it is straightforward to map the same word to the same index regardless of the source it comes from (either support statements or questions), it allows for greater flexibility for the model to treat words in support statements differently from those in questions. This is reflected in the performance boost when the model takes the source of each word into account (Weston et al., 2015). To accommondate this, instead of mapping words into a single embedding space, Φ now maps them into separate embedding spaces, each taking the form of a vector of size $|V|$. This essentially allows each word to have multiple embeddings depending on the type of the input source.

Training of MEMNN is carried out with stochastic gradient descent and a margin ranking loss function taking as input the score of each memory location and answer candidate computed by s_O and s_R, and ranking them by assigning a higher score to the true supporting memory locations and answer candidate, and a lower score to the rest:

$$\sum_{\bar{f}_1 \neq f_1} \max(0, \gamma - s_O(I(\mathbf{q}), \boldsymbol{m}_{\boldsymbol{f_1}}) + s_O(I(\mathbf{q}), \boldsymbol{m}_{\bar{f}_1})) \tag{2.43}$$

$$+ \sum_{\bar{f}_2 \neq f_2} \max(0, \gamma - s_O([I(\mathbf{q}), \boldsymbol{m}_{f_1}], \boldsymbol{m}_{f_2}) + s_O([I(\mathbf{q}), \boldsymbol{m}_{f_1}], \boldsymbol{m}_{\bar{f}_2}) \tag{2.44}$$

$$+ \sum_{\bar{r} \neq r} \max(0, \gamma - s_R([I(\mathbf{q}), \boldsymbol{m}_{f_1}, \boldsymbol{m}_{f_2}], r) + s_R([I(\mathbf{q}), \boldsymbol{m}_{f_1}, \boldsymbol{m}_{f_2}], \bar{r})) \tag{2.45}$$

where f_1, f_2 and r denote the indices of the true supporting memory locations and answer candidate, $\bar{f}_1$, $\bar{f}_2$ and r are their false counterparts, and γ is the score margin between true and false.

When applied to real-world problems, MEMNN can be further extended by ideas such as more sophisticated word tokenisation and efficient memory access via hashing, to enhance model performance (Weston et al., 2015).

2.3.1.3 Limitations of Memory Networks

Despite its success on a range of QA tasks, the applicability of MEMNN is severely limited by the requirement of knowing the locations of the true supporting statements. That is, in addition to the final output, the model also needs to have access to the location of each relevant clue in the memory during training (see Equations (2.43) and (2.44)). Not only does this call for additional annotations, it is sometimes an unrealistic demand for some datasets, making it impossible to be applied to those without such annotations.

While the introduction of MEMNN lays the groundwork for memory mechanisms, setting a firm foundation and paving the way for futher research on memory augmentation, the drawbacks identified above have severely hindered its application to NLP problems. Building on MEMNN, Sukhbaatar et al. (2015) introduce End-to-End Memory Networks (MEMN2N), removing the requirement of additional annotations, thereby making it possible to train the model end-to-end. This significantly increases the applicability of MEMNN and for the remainder of this thesis, we focus on MEMN2N and describe it in the next subsection Section 2.3.2.

2.3.2 End-to-End Memory Networks

To address the issues of MEMNN, most notably the issue of requiring additional memory location supervision, Sukhbaatar et al. (2015) introduce End-to-End Memory Networks (MEMN2N). Focusing on the text domain, Sukhbaatar et al. (2015) substantially simplify the formulation of the model. As a result, not only is MEMN2N simpler than MEMNN, it is also fully end-to-end trainable without the need for additional annotation other than the true gold label. Illustrations of MEMN2N with a single memory hop and multiple hops are shown in Figures 2.1 and 2.2, respectively.

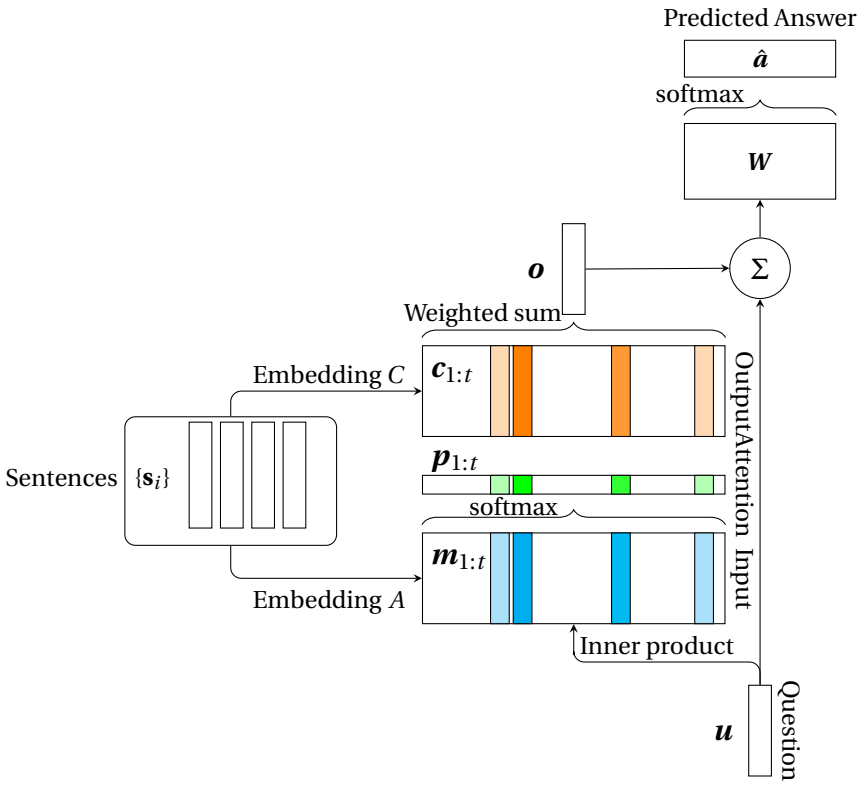

Figure 2.1 Illustration of a MEMN2N with a single memory hop.

2.3.2.1 Single-hop End-to-End Memory Networks

More formally, MEMN2N is comprised of two main components: (1) a supporting memory bank and (2) a memory controller. The supporting memory bank is used to store relevant information, for example, support statements in a story. At each hop of the memory network, MEMN2N first encodes sentences in a story into two types of representations: input and output, each serving a different purpose. The input memory representation is responsible for guiding the focus of the attention mechanism to select relevant memory locations given the input question **q** by measuring the similarity between **q** and each sentence encoded in the input memory space. In contrast, the output memory representation is more concerned with how to present the selected memory elements based on the attention weighting to form the output of the current memory hop. The input and output memory representations, along with the attention mechanism, constitute the core

of MEMN2N, reflecting the design of the output module in MEMNET. Specifically, this is achieved by first converting each input word in a sentence into its one-hot representation with a function Φ mapping an individual word to its one-hot vector of vocabulary size $|V|$. Next, MEMN2N transforms each input word s_{ij}, the j-th word in the i-th sentence, in its one-hot form into the embedding space with two distinct embedding matrices:

$$\boldsymbol{m}_i = \sum_{j=1}^{|\mathbf{s}_i|} \boldsymbol{A}\Phi(s_{ij}) \tag{2.46}$$

$$\boldsymbol{c}_i = \sum_{j=1}^{|\mathbf{s}_i|} \boldsymbol{C}\Phi(s_{ij}) \tag{2.47}$$

where $\boldsymbol{A}, \boldsymbol{C} \in \mathbb{R}^{d\times|V|}$ denote the input and output embedding matrices respectively, and $\Phi(s_{ij}) \in \mathbb{R}^{|V|}$.

To determine the relevance of each memory elements in the bank, we measure the similarity between the input question $\mathbf{q}$ and each memory location. To this end, the input question needs to be encoded first:

$$\boldsymbol{u} = \sum_{j=1}^{|\mathbf{q}|} \boldsymbol{B}\Phi(q_j) \tag{2.48}$$

where q_j denotes the j-th word in $\mathbf{q}$, Φ is the same function as in Equations (2.46) and (2.47), and $\boldsymbol{B}$ is an embedding matrix for questions. Then the relevance is measured by taking the dot product between $\boldsymbol{u}$ and each of the input memory locations $\boldsymbol{m}_i$:

$$p_i = \text{softmax}(\boldsymbol{u} \cdot \boldsymbol{m}_i). \tag{2.49}$$

The output of the current memory hop takes the form of a weighted sum of the output memory representation:

$$\boldsymbol{o} = \sum_{i=1}^{m} p_i \boldsymbol{c}_i. \tag{2.50}$$

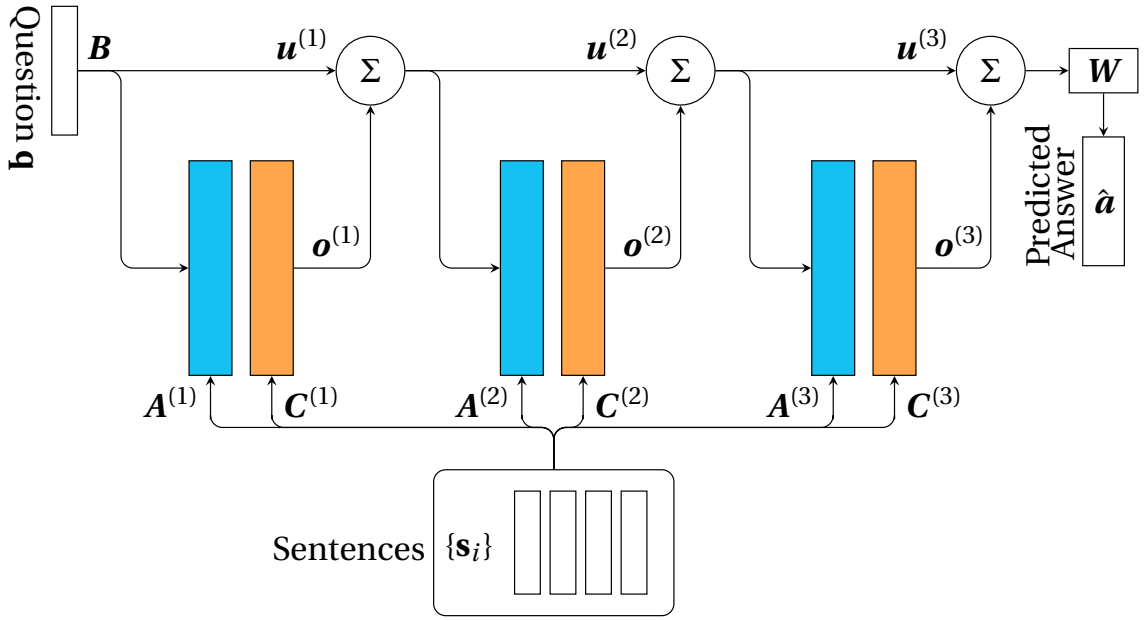

Figure 2.2 Illustration of a MEMN2N with multiple memory hops.

Lastly, the output, together with the question representation, is used to predict the final answer:

$$\hat{\boldsymbol{a}} = \text{softmax}(\boldsymbol{W}(\boldsymbol{o} + \boldsymbol{u})) \tag{2.51}$$

where $\boldsymbol{W} \in \mathbb{R}^{|V| \times d}$ is a trainable weight matrix and this results in a distribution over all vocabulary words. In this case, we assume that the response is a single word in the vocabulary V and to get the final predicted answer, one can simply take:

$$\hat{a} = \arg\max \hat{\boldsymbol{a}}. \tag{2.52}$$

2.3.2.2 Multi-hop End-to-End Memory Networks

For more demanding tasks requiring reasoning capabilities over mutiple supporting facts in the memory bank, MEMN2N can be easily extended to include more memory hops. In this setting, several layers of input/output memory representations are stacked with the output of the previous hop taken as input to the next one. More formally, with the superscript in parentheses denoting the index of a memory hop, a multi-hop MEMN2N is

defined analogously to its single-hop counterpart:

$$\boldsymbol{u}^{(1)} = \sum_{j=1}^{|\mathbf{q}|} \boldsymbol{B}\Phi(q_j) \tag{2.53}$$

$$\boldsymbol{m}_i^{(k)} = \sum_{j=1}^{|\mathbf{s}_i|} \boldsymbol{A}^{(k)}\Phi(s_{ij}) \tag{2.54}$$

$$\boldsymbol{c}_i^{(k)} = \sum_{j=1}^{|\mathbf{s}_i|} \boldsymbol{C}^{(k)}\Phi(s_{ij}) \tag{2.55}$$

$$p_i^{(k)} = \text{softmax}(\boldsymbol{u}^{(k)} \cdot \boldsymbol{m}_i^{(k)}) \tag{2.56}$$

$$\boldsymbol{o}^{(k)} = \sum_{i=1}^{m} p_i^{(k)} \boldsymbol{c}_i^{(k)} \tag{2.57}$$

with the addition of the connection between hops:

$$\boldsymbol{u}^{(k+1)} = \boldsymbol{o}^{(k)} + \boldsymbol{u}^{(k)} \tag{2.58}$$

where the input question to the next hop $\boldsymbol{u}^{(k+1)}$ is comprised of the output of the previous hop $\boldsymbol{o}^{(k)}$ and the previous input question $\boldsymbol{u}^{(k)}$ via a residual connection. Lastly, similar to the single-hop setting, the response is predicted as:

$$\hat{\boldsymbol{a}} = \text{softmax}(\boldsymbol{W}(\boldsymbol{o}^{(K)} + \boldsymbol{u}^{(K)})) \tag{2.59}$$

where K is the total number of layers.

Notice that in this formulation, each memory hop now has its own pair of input and output embeddings $\boldsymbol{A}^{(k)}$ and $\boldsymbol{C}^{(k)}$. With the addition of the embedding matrix for the input question $\boldsymbol{B}$ (Equation (2.53)) and the weight matrix $\boldsymbol{W}$ in the final classifier (Equation (2.59)), this results in $2K+2$ weight matrices each consisting of $d \times |V|$ parameters. While it is possible to keep all of them independent, doing so may cause overfitting and it is therefore beneficial to make some of them shared across different hops. More details on weight tying of embedding matrices will be discussed in the next section (Section 2.3.2.3).

Intuitively, multi-hop MemN2N can be understood as incorporating newly observed relevant information from the previous memory $\boldsymbol{o}^{(k)}$ by adding it back to the original

question $\boldsymbol{u}^{(k)}$, thereby re-formulating the input question to $\boldsymbol{u}^{(k+1)}$ for the next hop. Consider the example in Table 2.1. With $k = 2$ and the incoming question *Where is the milk now?*, a MEMN2N first scores all support statements in the memory bank and will likely place the majority of its attention on the fifth sentence *Jeff left the milk*, which becomes the dominating piece of information of the weighted sum forming the output of the first memory hop $\boldsymbol{o}^{(1)}$. The output, carrying this newly found supporting evidence, is then incorporated into the question with Equation (2.58), potentially forming a new question in the embedding space $\boldsymbol{u}^{(2)}$ possibly expressing the question: *Where was Jeff before he left the milk?*. The focus of the search is then shifted from looking for the location of the milk to the last place Jeff visited while carrying the milk. At the second memory hop, with the new question $\boldsymbol{u}^{(2)}$, the model now focuses on the fourth sentence *Jeff travelled to the bedroom* and propagates this new piece of evidence to the final classifier in Equation (2.59) to produce the predicted answer to the original question.

2.3.2.3 Weight Tying on Embedding Weight Matrices

An important concept in the multi-hop setting is weight tying, the goal of which, as discussed in the previous section (Section 2.3.2.2), is mainly to prevent overfitting. Specifically, we focus on different weight-tying schemes applied to the embedding weight matrices $\boldsymbol{A}^{(i)}$ and $\boldsymbol{C}^{(i)}$.

There are two types of weight tying in the context of memory networks:

1. layer-wise: the same input and output embedding matrices are shared across different memory hops, that is, $\boldsymbol{A}^{(1)} = \boldsymbol{A}^{(2)} = \ldots = \boldsymbol{A}^{(K)}$ and $\boldsymbol{C}^{(1)} = \boldsymbol{C}^{(2)} = \ldots = \boldsymbol{C}^{(K)}$, resembling RNNs in that the same set of weight matrices is applied at every step (hop);

2. adjacent: the output embedding matrix at the current memory hop is shared with the input embedding matrix one hop above, more formally: $\boldsymbol{C}^{(k)} = \boldsymbol{A}^{(k+1)}$. In addition to the constraints on the input/output embedding matrices at each hop, we further put constraints on: (1) the weight matrix $\boldsymbol{W}$ in the final classifier (Equa-

tion (2.59)) to share weight parameters with the output embedding matrix $\boldsymbol{C}^{(K)}$ at the last memory hop, i.e., $\boldsymbol{W}^\top = \boldsymbol{C}^{(K)}$; (2) the embedding matrix $\boldsymbol{B}$ for the input question (Equation (2.53)) to be the same as the input embedding matrix $\boldsymbol{A}^{(1)}$ at the first hop, i.e., $\boldsymbol{B} = \boldsymbol{A}^{(1)}$.

Additionally, with the layer-wise weight tying scheme, Sukhbaatar et al. (2015) find it beneficial to place a linear transformation between memory hops so that Equation (2.58) becomes:

$$\boldsymbol{u}^{(k+1)} = \boldsymbol{H}\boldsymbol{u}^{(k)} + \boldsymbol{o}^{(k)} \tag{2.60}$$

where $\boldsymbol{H} \in \mathbb{R}^{d \times d}$ is a trainable weight matrix. This can be seen as a way to increase model capacity, allowing more flexibility in the mapping functions between memory hops.

There is no definitive answer as to which weight tying scheme performs better but rather they each have their own strengths and weaknesses when applied to different problems.

2.3.2.4 Positional and Temporal Encoding

To account for word- and sentence-level positional information, MEMN2N further incorporates two types of encoding. For word-level processing, positional encoding is applied with element-wise multiplication with a column vector $\boldsymbol{l}_j$ of size d whose i-th element takes the value of: $l_{ij} = (1 - j/J) - (i/d)(1 - 2j/J)$ where J is the total length of the sentence with 1-based indexing. Word-level positional encoding is incorporated:

$$\boldsymbol{m}_i^{(k)} = \sum_{j=1}^{|\mathbf{s}_i|} \boldsymbol{l}_j \odot \boldsymbol{A}^{(k)} \Phi(s_{ij}) \tag{2.61}$$

$$\boldsymbol{c}_i^{(k)} = \sum_{j=1}^{|\mathbf{s}_i|} \boldsymbol{l}_j \odot \boldsymbol{C}^{(k)} \Phi(s_{ij}) \tag{2.62}$$

where $\odot$ denotes the Hadamard product.

For sentence-level processing, a temporal encoding is learned with the help of temporal embedding matrices, helping the model incorporate sequence ordering information in which sentences in a story appear. These temporal embedding matrices are formu-

lated similarly to their word embedding counterparts in that they are subject to the same weight-tying constraints. They, however, differ in the way they are applied:

$$\boldsymbol{m}_i^{(k)} = \boldsymbol{T}_{Ai}^{(k)} + \sum_{j=1}^{|\mathbf{s}_i|} \boldsymbol{l}_j \odot \boldsymbol{A}^{(k)} \Phi(s_{ij}) \tag{2.63}$$

$$\boldsymbol{c}_i^{(k)} = \boldsymbol{T}_{Ci}^{(k)} + \sum_{j=1}^{|\mathbf{s}_i|} \boldsymbol{l}_j \odot \boldsymbol{C}^{(k)} \Phi(s_{ij}) \tag{2.64}$$

where $\boldsymbol{T}_{Ai}, \boldsymbol{T}_{Ci} \in \mathbb{R}^d$ are trainable parameters drawn from their corresponding temporal encoding weight matrices $\boldsymbol{T}_A$ and $\boldsymbol{T}_C$. Note that to prevent overfitting, Sukhbaatar et al. (2015) propose to regularise temporal encoding weight matrices by injecting random noise, i.e., adding 10% empty memories to work as dummies, thereby encouraging learning time invariant representations.

2.3.2.5 End-to-End Training

One of the major advantages of MEMN2N over MEMNN is that the model can be trained fully end-to-end. Specifically, training is carried with a cross entropy loss:

$$\text{CrossEntropy}(\boldsymbol{a}, \hat{\boldsymbol{a}}) = -\sum_{i=1}^{|\boldsymbol{a}|} a_i \log(\hat{a}_i) \tag{2.65}$$

where $|\boldsymbol{a}|$ is the dimension of the binary one-hot gold label vector (equivalent to the number of possible classes), and log is the natural logarithm function. Notice that there are no extra terms other than the cross entropy between the true and predicted answer distribution, eliminating the dependency on memory location supervision, and thereby significantly reducing the annotation requirements, making MEMN2N more applicable to a wide range of tasks.

2.3.2.6 What Can MEMN2N Learn?

To better understand what MEMN2N can learn from data, in Table 2.2, we show an example of the model trained on task 3 of the bAbI dataset. This requires a model to

Story	Support	MemNN		
		Hop 1	Hop 2	Hop 3
John moved to the hallway.		0.00	0.00	0.00
John grabbed the football.	yes	0.00	1.00	0.00
John journeyed to the garden.		0.35	0.00	0.00
Sandra moved to the hallway.		0.00	0.00	0.00
John went back to the hallway.	yes	0.00	0.00	1.00
John journeyed to the garden.	yes	0.62	0.00	0.00
Question: Where was the football before the garden? Answer: hallway, MemN2N prediction: hallway				

Table 2.2 An example of the attention weights of a 3-hop MemN2N trained on task 3 (3 supporting facts) of the bAbI dataset, also showing the true answer as well as the prediction result (example taken from (Sukhbaatar et al., 2015)). True supporting sentences are marked "yes" in the support column. Colour intensity indicates the value of the attention weight on each sentence at each memory hop.

find the answer to a question (given the story) which, in turn, requires gathering relevant clues from 3 supporting facts. Note that while the supporting sentences in the story have also been annotated, MemN2N does not receive this signal in any format at either training or test time, making the training of MemN2N fully end-to-end. In this example, this particular trained instance of MemN2N first tries to locate sentences containing the mention of *garden*, namely the third and last sentence. Next, the model looks for the second key entity in the question *football* and focuses on the second sentence, allowing it to realise that the *football* is now carried by *John*. Lastly, based on what it has learned so far, MemN2N pinpoints the answer by placing all its attention on the second last sentence, providing crucial evidence to answering the question correctly. The process of gradually gathering information from the collection of sentences, with the help of 3 memory hops, roughly mimics how humans would do when trying to answer the same question given this short story fragment.

While there are other types of tasks, 20 in total in the bAbI dataset with varying degrees of difficulties, a fair number of them can be solved (with an accuracy greater than 95%) by a vanilla MemN2N trained on each task independently. As demonstrated by Sukhbaatar

et al. (2015), on most tasks, MEMN2N learns to make sensible use of the memory hops, showing its potentials to be generalised to other tasks.

Note that the use of bAbI can be seen as a sanity check, often followed by further validation on real-world tasks as in the works of Graves et al. (2016), Henaff et al. (2017), and Banino et al. (2020). We next review a number of variants of MEMN2N with a detailed discussion on state of the art memory-augmented models presented in Section 2.5.

2.3.2.7 Variants of MEMN2N

Variants of MEMN2N have been successfully applied to many tasks in NLP, ranging from sentence-level analysis, e.g., sentiment detection and polarity classification of tweets (Tang et al., 2016b, Chen et al., 2017), to discourse-level analysis (Bordes and Weston, 2017, Xiong et al., 2016), potentially over large-scale document collections, e.g., question answering over Wikipedia documents (Miller et al., 2016).

While the above works are based on similar model architectures, they differ in how they adopt MEMN2N to their particular problems. For aspect-based sentiment analysis, Tang et al. (2016b) and Chen et al. (2017) store the entire sequence of words in a sentence in the memory where one word occupies a memory cell, and frame aspect terms as questions to compute attention weights over memory cells, allowing the model to focus more on the context most relevant to the input aspect term. At the other end of the spectrum, for question answering over large-scale knowledge bases, Miller et al. (2016) extend MEMN2N by representing each memory entry with a key-value pair and store information from an external knowledge source in its memory. Given that the knowledge source can be potentially large, leading to expensive computational cost in calculating the attention weights over all memory slots, Miller et al. (2016) introduced a key hashing mechanism, based on an inverted index, to pre-select a subset of all stored entries for further computation.

One major advantage of MEMN2N, and memory-augmented architectures with attention addressing mechanisms in general, over traditional RNN-based models is that the processing of an input sequence does not rely on recurrency. That is, the output at

step t does not depend on that at step $t-1$, making the processing of an input sequence highly parallelisable and therefore significantly reducing runtime as shown by Tang et al. (2016b). This property was further exploited by Vaswani et al. (2017) in their work on TRANSFORMERs to accelerate training speed, removing the dependency on recurrency while maintaining equally competitive, if not superior, performance compared to prior art.

In this thesis, taking inspiration from the above works, we build and improve upon existing static memory network-based architectures and demonstrate strong performance on a range of discourse tasks.

2.3.2.8 Comparision with RNN-based Models and MEMNN

MEMN2N differs from RNN-based models in that the memory is global and can be accessed with attention. This is in contrast to the memory in LSTMs and GRUs, which also works as the internal states of the network, making it unstable over long distances.

The main difference between MEMNN and MEMN2N is the removal of the requirement of the memory location supervision, allowing the model to be trained fully end-to-end. Apart from this, the argmax function used in MEMNN, which causes training difficulties via backpropagation, is now replaced with a softmax function, making MEMN2N a continuous form of MEMNN.

2.3.3 Non-recurrent Self-Attention-based Sequence Processing

More recently, Vaswani et al. (2017) introduce TRANSFORMERs, a deep model with multi-head self-attention mechanisms and residual connections under the encoder-decoder framework for the task of machine translation. It can be seen as a more sophisticated extension to MEMN2Ns, both equipped with multiple memory hops which are progressively constructed based on previous layers, positional encoding, and residual connections. However, TRANSFORMERs differ from MEMN2Ns in the more complex multi-head self-attention mechanism, layer normalisation and many other carefully designed components

to make the model perform competitively on real-world tasks, as opposed to the synthetic setting with MEMN2Ns.

The key motivation for TRANSFORMERs is to process input sequences with attention alone, removing the dependency on recurrency and therefore making the model highly parallelisable. In addition to the improved runtime, relying solely on attention for sequence processing also enables the model to capture long-range contextual dependencies as the attention mechanism is now free to attend to any position in the sequence, in contrast to RNN-based models where information regarding prior steps is forced into a fixed-size encoding.

The original TRANSFORMER model was proposed for transduction tasks, such as machine translation where both the input and output are sequences of varying lengths, typically consisting of an encoder and a decoder. In this thesis, without loss of generality, we focus on the encoder due to the similarity it shares with the decoder, and the broad applicability of the encoder of TRANSFORMERs to a wide number of downstream tasks.

2.3.3.1 Model Formulation

Consider an input sequence $\mathbf{x} = \{\boldsymbol{x}_1^{(0)}, \ldots, \boldsymbol{x}_{|\mathbf{x}|}^{(0)}\}$ with each element $\boldsymbol{x}_i^{(0)} \in \mathbb{R}^d$ being the embedding representation of the i-th input x_i. For simplicity, we represent the entire input sequence as a matrix $\boldsymbol{X}^{(0)} \in \mathbb{R}^{|\mathbf{x}| \times d}$ with the i-th row taking the value of $(\boldsymbol{x}_i^{(0)})^\top$. Here, we index encoder layers with a superscript in parentheses.

Instead of reading one element at a time, the encoder of a TRANSFORMER consumes the entire sequence at once. The encoder is comprised of multiple encoder blocks and each encoder block consists of two main components, a multi-head self-attention mechanism and a position-wise fully connected feed-forward network.

Multi-head Self-Attention

Specifically, the input to the l-th encoder is first projected into three different spaces: query, key and value with three distinct linear transformations:

$$\boldsymbol{Q}_i^{(l)} = \boldsymbol{X}^{(l-1)}\boldsymbol{W}_{qi}^{(l)} \tag{2.66}$$

$$\boldsymbol{K}_i^{(l)} = \boldsymbol{X}^{(l-1)}\boldsymbol{W}_{ki}^{(l)} \tag{2.67}$$

$$\boldsymbol{V}_i^{(l)} = \boldsymbol{X}^{(l-1)}\boldsymbol{W}_{vi}^{(l)} \tag{2.68}$$

where $\boldsymbol{W}_{qi}^{(l)} \in \mathbb{R}^{d\times d_k}$, $\boldsymbol{W}_{ki}^{(l)} \in \mathbb{R}^{d\times d_k}$ and $\boldsymbol{W}_{vi}^{(l)} \in \mathbb{R}^{d\times d_v}$ are trainable weight matrices, and i is the index of the attention head. A typical choice of d_k and d_v is $\frac{d}{h}$ where h is the number of attention heads.

Next, the model computes the self-attention weights for the i-th head by measuring the similarity between the projected queries and keys. The self-attention weights are then used to obtain a weighted sum of the projected values. For the entire sequence in the matrix form, the above two steps can be formulated as:

$$\tilde{\boldsymbol{X}}_i^{(l)} = \text{softmax}(\frac{\boldsymbol{Q}_i^{(l)}(\boldsymbol{K}_i^{(l)})^\top}{\sqrt{d_k}})\boldsymbol{V}_i^{(l)} \tag{2.69}$$

where $\tilde{\boldsymbol{X}}_i^{(l)} \in \mathbb{R}^{|\mathbf{x}|\times d_v}$. This generalises to each of the h self-attention heads, resulting in:

$$\tilde{\boldsymbol{X}}^{(l)} = [\tilde{\boldsymbol{X}}_1^{(l)};\ldots;\tilde{\boldsymbol{X}}_h^{(l)}]\boldsymbol{W}_o^{(l)} \tag{2.70}$$

where $[;]$ denotes concatenation and $\boldsymbol{W}_o^{(l)} \in \mathbb{R}^{hd_v\times d}$ is a trainable weight matrix. This results in a matrix $\tilde{\boldsymbol{X}}^{(l)} \in \mathbb{R}^{|\mathbf{x}|\times d}$.

The output of the multi-head self-attention mechanism is formed by adding the input $\boldsymbol{X}^{(l-1)}$ element-wise with a residual connection and then applying layer normalisation (Ba et al., 2016) to the sum:

$$\check{\boldsymbol{X}}^{(l)} = \text{LayerNorm}(\boldsymbol{X}^{(l-1)} + \tilde{\boldsymbol{X}}^{(l)}). \tag{2.71}$$

Fully Connected Feed-Forward Network

The output of the multi-head self-attention mechanism is then taken as input to a fully connected feed-forward network with a ReLU activation function (Glorot et al., 2011):

$$\dot{\boldsymbol{X}}^{(l)} = \text{ReLU}(\check{\boldsymbol{X}}_{(l)}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2 \tag{2.72}$$

where $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are trainable weight matrices, and $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are trainable bias terms.

The output of the feed-forward network is formed analogously to that of the multi-head self-attention mechanism with layer normalisation applied to the element-wise summation of $\dot{\boldsymbol{X}}^{(l)}$ and the residual connection $\check{\boldsymbol{X}}^{(l)}$:

$$\boldsymbol{X}^{(l)} = \text{LayerNorm}(\dot{\boldsymbol{X}}^{(l)} + \check{\boldsymbol{X}}^{(l)}). \tag{2.73}$$

Note that the output of the fully connected feed-forward network in Equation (2.73) forms the output of the l-th encoder block, which, in turn, forms the input to the $l+1$ encoder block. This allows multiple encoder blocks to be stacked, enabling the model to learn deep and progressively incremental representations of language.

2.3.3.2 What can Self-Attention Learn?

The self-attention mechanism allows each element in a sequence to incorporate knowledge from all elements (including itself) in the same sequence by attending to them with a different weight. Consider the task of machine translation where the input to the encoder is a sequence of words. In this case, each word is first projected into the query space and then matched against words in the same sequence projected into the key space to determine the relevance. The values of words with high relevance are incorporated into the current word, forming the output to the next encoder block. Vaswani et al. (2017) show that, by inspecting words of high relevance, the multi-head self-attention mechanism of TRANSFORMERS, although not explicitly trained to do so, is capable of learning to perform anaphora resolution, capturing long-range dependencies and internal sentence structure.

2.3.3.3 Variants of TRANSFORMERS

The success of TRANSFORMERS has spurred this line of research of utilising attention to process sequences and been applied to many downstream tasks, ranging from sentence-level text categorisation, such as sentiment analysis and text entailment (Devlin et al., 2019), to document-level comprehension and inference (Yu et al., 2018). For example, the impressive performance of QANET (Yu et al., 2018) can largely be attributed to the highly parallelisable nature of TRANSFORMERS, thereby allowing it to take advantage of data augmentation techniques, such as back-translation, to substantially improve its accuracy. Perhaps even more impressively, BERT (Devlin et al., 2019), a TRANSFORMER based deep text encoder trained on large-scale corpora with a novel masked language modelling objective, has been demonstrated to have strong performance over a variety of NLP tasks, highlighting its capabilities to capture the syntactic structure and semantics of text of arbitrary length.

2.3.3.4 Comparison with RNN-based Models and MEMN2N

The TRANSFORMER architecture is significantly different from RNNs in the removal of the dependency on recurrent steps, i.e., step t depends on the previous step $t-1$, eliminating the bottleneck in runtime speed and also the locality bias towards immediately neighbouring elements. Instead, TRANSFORMERS rely on self-attention, enabling the model to capture long-range contextual dependencies.

In comparison to MEMN2N, TRANSFORMER is similar in the resemblance of the self-attention mechanism to the recurrent attention in MEMN2N. While a MEMN2N collects evidence from its memory into the controller, with a TRANSFORMER, every input element in a sequence can be perceived as an individual memory controller, starting as the embedding of a word and gradually building up its representation by incorporating information from its context via self-attention.

While more sophisticated with the addition of feedforward and layer normalisation, TRANSFORMER blocks bear a resemblance to MEMN2N in the similarity between self-attention and the interaction between the query and input/output memory in a MEMN2N.

In the formulation of a TRANSFORMER, the key/value can largely be thought of as the input/output memory encoding in a MEMN2N. In both models, the keys (or input memory blocks) are matched in similarity against the query, resulting in a distribution over all memory elements, which is then used to obtain a weighted sum of the values (or output memory blocks) to incorporate information from the entire sequence.

It remains an open question as to whether memory-augmentation is beneficial to a TRANSFORMER-based language encoder, such as BERT, to incorporate external knowledge. The findings of Zhang et al. (2019) suggest that such TRANSFORMER-based encoders, although pretrained on large-scale unstructured text, are insufficient in their ability to fully comprehend the rich structured knowledge stored in knowledge graphs, and that incorporating such information helps improve language understanding. This indicates that some forms of knowledge may be difficult to capture with unsupervised training of language models over large collections of text. In this regard, memory-augmentation may potentially be good fit for memory-augmentation and promising avenue for further investigation. Note that the incorporation of external knowledge via memory-augmentation is discussed at length in Section 6.2.2.

2.4 Dynamic Memory Networks

Besides static memory networks, the other important category is dynamic memory networks. As the name suggests, here the contents of the memory are dynamically updated as new information arrives. This is in contrast to static memory networks where memory cells are read and kept unchanged throughout the processing of each instance, mainly serving a supportive role to provide evidence to some complex task.

In this section, we analyse some of the most relevant works to this thesis, all augmented with external, dynamic neural memory.

2.4.1 Data-driven Algorithm Learning

Concurrent to the works of Weston et al. (2015) and Sukhbaatar et al. (2015), Graves et al. (2014) explored the possibility of performing algorithmic tasks, such as copying and sorting, with a model trained in a data-driven fashion. Specifically, an LSTM-based controller is responsible for manipulating the contents of a continuous memory, which is considered decoupled from the controller. The subsequent work of Graves et al. (2016) further extends the model and applies it to more challenging tasks, such as graph-based traversal problems and block puzzles.

In this section, we describe these two models and compare them to static memory networks.

2.4.1.1 Neural Turing Machine

Concurrent to the works of Weston et al. (2015) and Sukhbaatar et al. (2015) on MEMNET and MEMN2N, Graves et al. (2014) introduce the Neural Turning Machine (NTM), an analogous model to the Turing Machine (von Neumann, 1993). While RNNs are Turing-complete (Siegelmann and Sontag, 1995) and can theoretically approximate any procedure, in practice, this is not always the case (Graves et al., 2014), especially when the RNN cell is not properly configured. As mentioned before, plain RNNs suffer from the issue of vanishing and exploding gradient. The more powerful variant LSTM, on the other hand, although embedded with the "perfect integrator" (Seung, 1998) in the form of "memory cells", only alleviates the problem and has been shown to struggle when dealing with long-range dependencies (Cho et al., 2014a, Lai et al., 2015, Linzen et al., 2016).

NTMs aim to complement this with an external, addressable memory via soft attention, making the model fully differentiable and trainable with gradient descent efficiently. Specifically, an NTM consists of a controller and a memory bank. Interaction with the input and output is performed through the controller, which is also responsible for constructing the read and write heads to the memory bank.

Formally, reading is performed with a weighted sum of the memory:

$$\boldsymbol{r}_t = \boldsymbol{M}_t^\top \boldsymbol{\alpha}_t \tag{2.74}$$

where $\boldsymbol{M}_t \in \mathbb{R}^{n \times d}$ is the memory bank with n indicating the number of elements, and d the dimension of each element, $\boldsymbol{\alpha} \in \mathbb{R}^n$ is the attention weights indicating the relevance of each memory location.

Writing, on the other hand, is comprised of two steps: erase and add, each represented by a vector ($\boldsymbol{e}_t, \boldsymbol{a}_t \in \mathbb{R}^d$), also generated by the controller. To write to the memory, an NTM first needs to erase knowledge deemed irrelevant for future processing and then add new information:

$$\tilde{\boldsymbol{M}}_t = \boldsymbol{M}_{t-1} \odot [\mathbf{1} - \boldsymbol{\alpha}_t \boldsymbol{e}_t^\top] \tag{2.75}$$

$$\boldsymbol{M}_t = \tilde{\boldsymbol{M}}_t + \boldsymbol{\alpha}_t \boldsymbol{a}_t^\top \tag{2.76}$$

Here, the relevance is determined by the attention weights $\boldsymbol{\alpha}_t$.

With this, NTM can be trained to learn to focus on highly relevant memory locations, leaving irrelevant ones out, and accumulating knowledge over time by maintaining the memory bank through the read/write heads and attention weights.

A key component in NTM is the attention weights $\boldsymbol{\alpha}_t$, which is a distribution over the n memory locations, such that:

$$\sum_{i=1}^{n} \alpha_{t,i} = 1, \quad 0 \leqslant \alpha_{t,i} \leqslant 1, \forall i \tag{2.77}$$

where $\alpha_{t,i} \in \mathbb{R}$ is the i-th element in $\boldsymbol{\alpha}_t$. This works as the primary addressing mechanism for NTM to access the memory bank.

The attention weights are formed by two types of addressing mechanisms: content- and location-based addressing. Content-based addressing, a term first coined by Hopfield (1982), guides the focus to memory locations whose contents have high similarity to a key

vector $\boldsymbol{k}_t$ generated by the controller:

$$\alpha^c_{t,i} = \frac{\exp(\beta_t \text{sim}(\boldsymbol{k}_t, \boldsymbol{M}_{t,i}))}{\sum_j \exp(\beta_t \text{sim}(\boldsymbol{k}_t, \boldsymbol{M}_{t,j}))} \tag{2.78}$$

this results in a distribution over all memory locations. Here, β_t is a positive key strength parameter, also generated by the controller, to sharpen or soften the resulting distribution. This is essentially equivalent to the inverse of the temperature τ in the softmax function. The higher the value for β_t, the sharper or more confident the distribution is, and vice versa. The sim function is the cosine similarity measure:

$$\text{sim}(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\|\boldsymbol{a}\| \cdot \|\boldsymbol{b}\|}. \tag{2.79}$$

However, simply relying on content similarity-based extraction may not be powerful enough in some cases. For some tasks, such as simple sequence copying, it is beneficial for the model to be aware of the activated memory locations at the last step. Without access to previously attended memory locations, NTMs fail to generalise to sequences with length longer than those seen in the training set (Graves et al., 2014). In addition to content-based attention, NTMs further take the focus of the model at the previous step into account. Specifically, attention at the previous step is merged with interpolation:

$$\boldsymbol{\alpha}^g_t = g_t \boldsymbol{\alpha}^c_t + (1 - g_t) \boldsymbol{\alpha}_{t-1}. \tag{2.80}$$

The controller further defines a shift weighting distribution $\boldsymbol{s}_t$ indicating the likelihood over the allowed integer shifts (e.g., shifts of -1, 0, 1 from the current location). The model then performs a convolutional shift (Equation (2.81)) and lastly produces the attention weights (Equation (2.82)):

$$\tilde{\alpha}_{t,i} = \sum_{j=0}^{n-1} \alpha^g_{t,j} s_{t,i-j} \tag{2.81}$$

$$\alpha_{t,i} = \frac{\tilde{\alpha}^{\gamma_t}_{t,i}}{\sum_j \tilde{\alpha}^{\gamma_t}_{t,j}} \tag{2.82}$$

where γ_t is a positive scalar $\geqslant 1$ emitted by the controller to control the sharpness of the resulting distribution (analogous to β_t in Equation (2.78)).

Using this architecture, Graves et al. (2014) show that the model can be trained, in a data-driven fashion, to perform simple tasks, such as copying random sequences, sorting and associative recall, not only much better in performance but also requiring fewer training instances than LSTMs. The performance advantage of NTMs over LSTMs is more pronounced on longer sequences. Moreover, NTMs achieve so in an interpretable way, making sensible use of the content- and location-based addressing mechanisms, mimicking the procedures humans would follow when performing those tasks. This type of NTM-style memory has been shown by Wang et al. (2016) to be beneficial to the task of machine translation, with memory augmented to the decoder of a seq2seq model, demonstrating strong performance on Chinese-English translation (a more detailed description of this work is presented in Section 2.5.1).

It was later pointed out by Graves et al. (2016) that, despite the impressive performance on such simple tasks, the performance of NTMs suffers when applied to the artificially generated text-based reasoning tasks in bAbI (Weston et al., 2016). A likely explanation is that the location-based addressing mechanism, due to its sequential nature, forces NTMs to access consecutive memory blocks as opposed to true "random access" which is crucial to the success on bAbI given the non-linear narrative structure of those stories.

2.4.1.2 Differentiable Neural Computer

Differentiable Neural Computer (DNC), the successor to NTM, is another architecture where data storage (external memory) is decoupled from the actual processing (controller). While the former can be thought of as the RAM in a computer, the latter is analogous to the CPU, accessing data stored in the RAM via its address. While largely sharing the same model architecture, DNC differs from its predecessor in the memory addressing mechanisms. Despite having content- and location-based addressing mechanisms, NTMs are rather limited in their ability to access the memory bank freely. Due to the design of the location-based addressing mechanism, NTMs are forced to write sequential data to

consecutive memory blocks. If the heads jump to a different location not in the neighbourhood of the previous focus, the chain of operations cannot be preserved and later recovered. In other words, the memory bank in a NTM is not "random access". Also, NTMs have no mechanism to free a block of memory nor to keep track of which part of the memory is currently in use. This may cause the model to accidentally write to memory locations allocated previously to keep record of certain data for future processing, especially when the memory capacity limit is reached.

DNCs improve upon NTMs in having more flexible memory access mechanisms. In addition to content-based addressing, it employs an $n \times n$ matrix $\boldsymbol{L}$ to keep track of transitions between consecutively written locations. Each element in $\boldsymbol{L}$ is a scalar in the range $[0, 1]$ with 1 at the i-th row and j-th column indicating that the i-th memory location was written to immediately after the j-th, and 0 otherwise. With this definition, simply multiplying with the attention weights $\boldsymbol{L\alpha}$ shifts the focus forwards to the next memory location whereas $\boldsymbol{L}^\top \boldsymbol{\alpha}$ reverses the attention backwards. This allows DNCs to recover the chain of operations regardless of whether data was originally stored in consecutive blocks.

Secondly, DNCs keep track of the "usage" of each memory block and recycle those that have not been in use for a while. This is implemented by increasing the weightings on locations after each write, and gradually decreasing after each read. With this new usage weighting, it allows DNCs to re-allocate memory blocks that are deemed "no longer in use", thereby freeing up memory space to make room for the storage of newly arrived data.

Graves et al. (2016) demonstrate the wide applicability of the proposed DNC architecture to a range of tasks, ranging from natural language-based reasoning to learning from data to perform algorithmic tasks such as finding the shortest path and inferring missing links in graphs. Of particular interest to this thesis is the text-based reasoning task bAbI. Perhaps even more importantly, the work of Graves et al. (2016) underlines the significance of coupling external memory to neural networks, enabling the model to have the capacity to solve complex problems. Without an external memory component, along with properly designed access mechanisms, neural network models tend to struggle when dealing with such tasks of a complex nature.

2.4.1.3 Comparison with Static Memory Networks

The most notable difference, when comparing NTM and DNC with MEMN2N, is that the memory components in dynamic memory networks can now be updated, as opposed to read-only in static memory networks. Similar to reading, which is achieved via attention, the writing mechanism is also modelled using attention, with an erasing step followed by an adding operation.

Another key difference, while often ignored, is in the memory addressing mechanism. While all based on attention, in addition to the content-based addressing used in MEMN2N, NTM and DNC are further equipped with a location-based accessing mechanism, allowing them to either utilise consecutive memory blocks (NTM) or trace back the last written location (DNC).

Lastly, DNCs allows memory blocks to be released and recycled once they are deemed no longer in use. This is achieved by keeping track of the usage of each block.

2.4.2 Recurrent Entity Networks

A key difference between static and dynamic memory networks is whether the contents of memory can be updated as new information arrives. This property can be used to model an important element of discourse: entities, in particular, how they are mentioned and interact with one another. Entities play an important role in language (Ji et al., 2017), as they tend to evolve over time as a story unfolds, which naturally brings out the question of how they should be represented in a model. More specifically, modelling such entities requires keeping track of their changing states, a good fit for dynamic memory networks. To this end, a variety of approaches have been explored, as described in Section 2.4.2.1.[1]

Of particular interest to this thesis is the work by Henaff et al. (2017) on Recurrent Entity Networks (ENTNETS), where entities are represented and stored with a fixed size memory independently, with each cell responsible for keeping track of the state of a single entity. Each memory cell is further equipped with a key, indicating the identity of the

[1] A detailed discussion of modern entity-centric models is presented in Section 6.2.1.

entity being tracked by this memory chain. Together, these memory cells can be perceived as external "memory chains" maintaining up-to-date representations of entities. When reading through a story, an ENTNET, upon seeing relevant input, updates the states of the entities by writing to the corresponding memory chain. The write operation is controlled by a gating mechanism, which measures the relevance (similarity) between the current input and each memory chain, deciding which chain(s) to update. Note that the writing operation is not mutually exclusive in that multiple memory chains can be updated at the same time. Unlike static memory network-based models (e.g., MEMNN and MEMN2N), where narrative understanding is performed by having multiple passes of the same story to progressively gather evidence and search for the next clue, ENTNETs do this in a single pass. We first describe the motivation of ENTNET with the help of an example to facilitate discussion in Section 2.4.2.2 and then detail the model formulation in Section 2.4.2.3.

2.4.2.1 Related Entity-Centric Models

Many attempts have been made to incorporate entity-level knowledge into the model architecture. Yang et al. (2017) propose a "reference-aware" language model, taking into account entities stored in three different information sources: (1) an item list; (2) an external database; and (3) context from the same document. In the context of their work, not only are entity representations updated by taking the recurrent state of the final token in an entity expression with no long-term history of the entity maintained, entity information is also required at test time (Ji et al., 2017), making it difficult to generalize to corpus without such annotations.

Another entity representation architecture is proposed by Kobayashi et al. (2017) where the representations of the same entity across the entire sequence are processed with a GRU and used not only in the output but the input layer as well. While taking this global view of entities shows benefits on the task of language modelling, they only consider single-token entities (multi-token entities are normalisd to anonymised identifiers, making them single-token entities as well), an unrealistic assumption to make in our view. Not only is critical information regarding entities lost in the anonymisation process, a language

model trained on such anonymised identifiers is severely limited in its applicability to downstream tasks.

Ji et al. (2017) introduce a generative entity-aware language model, named EntityNLM, capable of dynamically constructing entity representations on the fly. EntityNLM differs from EntNet in that: (1) in contrast to a fixed-size memory, EntityNLM creates a new memory cell upon seeing a new entity, resulting in greater capacity of the model, but comes at the cost of memory efficiency as no memory chains can be recycled; (2) EntityNLM requires the computationally expensive importance sampling during inference; and (3) more demanding annotation requirements, with more fine-grained, low-level annotations necessary for the training process compared to EntNet.

2.4.2.2 Motivation

The key motivation of EntNet is that to understand the narrative of a story, one needs to constantly update a set of high-level concepts related to the story as new input arrives, thereby maintaining an up-to-date perception of the entities mentioned in the story.

Consider the example shown in Table 2.1. Upon reading the fourth sentence *Jeff travelled to the bedroom.*, it is straightforward to see that, in one's mind, the property of *Jeff*'s location should be updated from *kitchen* to *bedroom*. In addition to *Jeff*'s location, the whereabouts of all the objects that are on *Jeff* should also be updated accordingly. In this case, the location property of *milk* should be updated due to it being carried by *Jeff*, a fact stated in the immediately preceding sentence *Jeff picked up the milk*.

The above example demonstrates that for a memory-augmented model to fully understand the narrative of the story, two types of addressing mechanisms are necessary: location- and content-based addressing. Location-based addressing is in charge of matching entities directly mentioned in the input (e.g., *Jeff* in the example) to the memory chains tracking those entities via their keys. Content-based addressing, on the other hand, is responsible for updating the states of those entities (e.g., *milk*) that may not be directly mentioned in the current input but are still involved implicitly due to previous interactions. In other words, location-based addressing is more concerned with the identity of

an entity whereas content-based addressing places more focus on the state of an entity and its relationships and involvement with other entities.

Obviously, memory chains in EntNets are not limited to just tracking the locations of entities. They can be used to track many other properties of and relationships between entities depending on the task. In addition, the model may also learn dynamic constraints, such as an entity can not be at two different locations at the same time. Once trained, EntNets can learn to perform narrative comprehension with some basic form of reasoning, and, more importantly, given that the states of entities stored in memory chains are only updated when encountering relevant input, the model can therefore keep entities intact over long timescales, thereby capturing long-range dependencies.

2.4.2.3 Memory Architecture

Consider the input sequence $\mathbf{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{|\mathbf{x}|}\}$ of length $|\mathbf{x}|$, where each element is the vector representation of an input (e.g., a word or a sentence).

Each memory chain largely resembles an RNN or LSTM with a much simplified recurrent unit and an EntNet may consist of multiple memory chains. We once again denote the index of a memory chain with a superscript in parentheses. Specifically, at every time step i, the current input representation $\boldsymbol{x}_i$ is fed into the j-th memory chain $\boldsymbol{m}^{(j)}$ via a gating mechanism g controlling the amount of update being pushed into that memory chain:

$$g_i^{(j)} = \sigma(\boldsymbol{x}_i \cdot \boldsymbol{k}^{(j)} + \boldsymbol{x}_i \cdot \boldsymbol{m}_{i-1}^{(j)}) \tag{2.83}$$

where σ is the sigmoid activation function, $\boldsymbol{k}^{(j)}$ is the key representing the identity of the entity being tracked by the j-th memory chain, and $\boldsymbol{m}_i^{(j)}$ is the state of the j-th memory chain. While the key for the j-th memory chain $\boldsymbol{k}^{(j)}$ can also be learned and updated at training time, it is invariant regardless of time step i. That is, $\boldsymbol{k}^{(j)}$ stays constant throughout the sequence $\mathbf{x}$.

It is important to note that the value of gate $g_i^{(j)}$ is determined by two dot products: the location term $\boldsymbol{x}_i \cdot \boldsymbol{k}^{(j)}$ and the content term $\boldsymbol{x}_i \cdot \boldsymbol{m}_{i-1}^{(j)}$, reflecting the location- and content-based address mechanisms respectively. Both of them lead to the activation of $g_i^{(j)}$ but

differ in how they turn the gate on. The former, the location-based term, quantifies how related the current input $\boldsymbol{x}_i$ is to the key $\boldsymbol{k}^{(j)}$ of the entity being tracked by the j-th memory chain whereas the latter, the content-based term, measures the similarity between the input and the current state of the tracked entity $\boldsymbol{m}_{i-1}^{(k)}$. That is, the location term causes the gate to open for memory chains whose keys $\boldsymbol{k}^{(j)}$ match the current input. The content-based term, on the other hand, triggers the activation of gate $g_i^{(j)}$ when the content of the entity matches the input.

Prior to the actual memory update, an ENTNET first computes a memory update candidate $\tilde{\boldsymbol{m}}_i^{(j)}$, taking into account the key of the j-th memory chain $\boldsymbol{k}^{(j)}$, the state of the j-th tracked entity $\boldsymbol{m}_{i-1}^{(j)}$ and the current input $\boldsymbol{x}_i$:

$$\tilde{\boldsymbol{m}}_i^{(j)} = \phi(\boldsymbol{U}\boldsymbol{m}_{i-1}^{(j)} + \boldsymbol{V}\boldsymbol{k}^{(j)} + \boldsymbol{W}\boldsymbol{x}_i) \tag{2.84}$$

where $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ are trainable weight matrices, and ϕ can be any arbitrary activation function with the typical choice being the parametric ReLU (PReLU) (He et al., 2015):

$$\text{PReLU}(x_i) = \max(0, x_i) + \alpha_i \min(0, x_i) \tag{2.85}$$

where PReLU can be applied element-wise to an input vector $\boldsymbol{x}$ and its parameters $\boldsymbol{\alpha}$ can be trained jointly with the final objective function. Note that the weight matrices $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ are shared across all memory chains.

Using the update gate $g_i^{(j)}$, an ENTNET can incorporate this new information carried by the memory update candidate $\tilde{\boldsymbol{m}}_i^{(j)}$ via simple addition:

$$\check{\boldsymbol{m}}_i^{(j)} = \boldsymbol{m}_{i-1}^{(j)} + g_i^{(j)} \odot \tilde{\boldsymbol{m}}_i^{(j)} \tag{2.86}$$

where $\odot$ is the Hadamard product, resulting in an unnormalised version of the memory representation $\check{\boldsymbol{m}}_i^{(j)}$. The influence of the memory update candidate on $\check{\boldsymbol{m}}_i^{(j)}$ is here controlled by the update gate $g_i^{(j)}$.

Lastly, following the update, the model performs a normalisation step, allowing the memory to forget:

$$\boldsymbol{m}_i^{(j)} = \frac{\check{\boldsymbol{m}}_i^{(j)}}{\|\check{\boldsymbol{m}}_i^{(j)}\|} \tag{2.87}$$

where $\|\check{\boldsymbol{m}}_i^{(j)}\|$ denotes the Euclidean norm of $\check{\boldsymbol{m}}_i^{(j)}$. As all information stored in $\boldsymbol{m}_i^{(j)}$ is now contrained to be of unit length, lying on the surface of the unit sphere, adding new information carried by $\tilde{\boldsymbol{m}}_i^{(j)}$ to the existing memory $\boldsymbol{m}_{i-1}^{(j)}$ makes the resulting vector $\check{\boldsymbol{m}}_i^{(j)}$ point to a different direction than the original (unless the two operands are pointing to the same direction). Performing normalisation to $\check{\boldsymbol{m}}_i^{(j)}$ inevitably causes forgetting in that the cosine distance between the original and updated memory decreases. The exact same normalisation step is also used in the work of Ji et al. (2017)

It can be noticed that the state of an entity $\boldsymbol{m}_i^{(j)}$ is not updated unless the model sees an input deemed relevant to the tracked entity, either directly matching the identity of the entity or indirectly connected, via prior interactions, to the state of the entity. Only then does the model allow the input to interact with the involved memory chains, and thereby modifying their contents.

An overview of the model architecture is presented in Figure 2.3, with a detailed depiction of a single recurrent unit of EntNet displayed in Figure 2.4.

2.4.2.4 Final Classifier

The final classifier may take one of many potentially possible forms. To facilitate discussion, let us consider a question answering scenario where the input consists of not only the story but also a question q. A typical choice in this case is an attentional classifier over the final memory chain representations with a residual connection of the input question q.

Specifically, once the input sequence $\mathbf{x}$ has been processed by the memory architecture of an EntNet, the states of the tracked entities are measured in similarity with the question representation $\boldsymbol{q}$:

$$p^{(j)} = \text{softmax}(\boldsymbol{q} \cdot \boldsymbol{m}_{|\mathbf{x}|}^{(j)}) \tag{2.88}$$

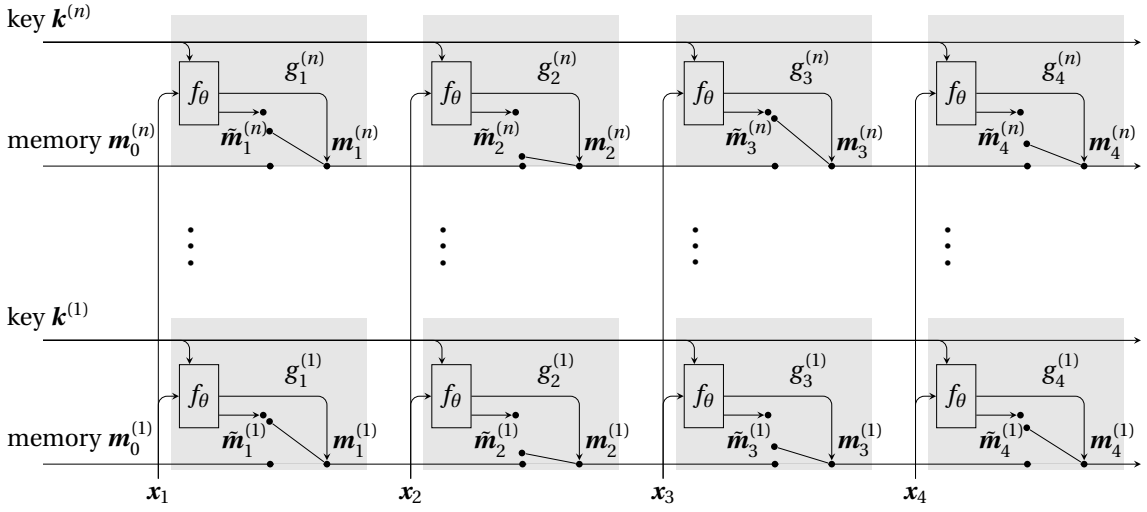

Figure 2.3 Illustration of an instance of EntNet with n memory chains.

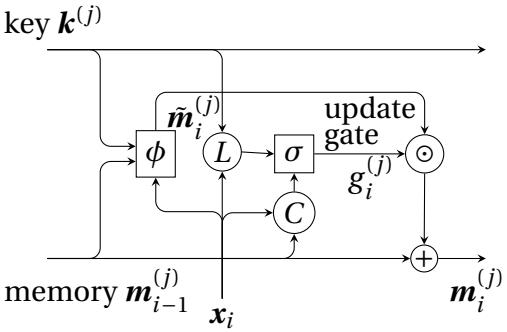

Figure 2.4 Illustration of an EntNet recurrent unit, depicting the inner workings of a shaded recurrent cell in Figure 2.3.

resulting in an attention distribution over all memory chains.

Using the attention distribution, the model summarises the memory representations with a weighted sum:

$$\boldsymbol{u} = \sum_{j=1}^{n} p^{(j)} \boldsymbol{m}_{|\mathbf{x}|}^{(j)}. \tag{2.89}$$

Lastly, the final prediction is produced by taking a non-linear transformation of the sum of weighted memory representation $\boldsymbol{u}$ and the question $\boldsymbol{q}$ with an arbitrary non-linear activation function ϕ:

$$\hat{\boldsymbol{y}} = \boldsymbol{R}\phi(\boldsymbol{q} + \boldsymbol{H}\boldsymbol{u}) \tag{2.90}$$

where $\boldsymbol{R}$ and $\boldsymbol{H}$ are trainable weight matrices.

The final classifier selects those memory chains with high relevance to the question and produces a prediction distribution over possible answer candidates for a classification task. At training time, the predicted distribution can be used to train the model with cross entropy loss against the true label.

2.4.2.5 Motivating Example

To better understand the model, we use the example shown in Table 2.1 to illustrate the processing of this instance. Let us focus on the third and fourth sentences and suppose that an EntNet works on the sentence-level, consuming one sentence at a time. Let us also assume that the EntNet makes use of its memory chains to encode entities and that the j_1-th and j_2-th memory chains track the states of the entities *Jeff* and *milk*.

Upon seeing the third sentence, *Jeff picked up the milk.*, the model is expected to activate both the j_1- and j_2-th memory chains due to the occurrence of *Jeff* and *milk* in the sentence. This can likely be achieved by the high matching values of the location addressing terms $\boldsymbol{x}_3 \cdot \boldsymbol{k}^{(j_1)}$ and $\boldsymbol{x}_3 \cdot \boldsymbol{k}^{(j_2)}$, causing the gates ($g_3^{(j_1)}$ and $g_3^{(j_2)}$) to open. The EntNet therefore edits the memory states of these two entities, adding *carrying the milk* to *Jeff* and *carried by Jeff* to *milk*.

Next, the fourth sentence, *Jeff travelled to the bedroom.*, should trigger the model to again turn on gate $g_4^{(j_1)}$ for *Jeff* because of the occurrence of *Jeff* matching the key $\boldsymbol{k}^{(j_1)}$,

leading to a high value for the location term and therefore opening the gate. The ENTNET is also supposed to update the memory states for *milk* and can be encouraged to do so with a dominating content-based addressing term. At the previous step, the model should have already merged the knowledge of the *milk* being carried by *Jeff* into the content of the j_2-th memory chain. While the input $\boldsymbol{x}_4$ may not match the key of the memory chain $\boldsymbol{k}^{(j_2)}$ due to the lack of occurrence of *milk* in the fourth sentence, the content-based addressing term should instead result in a high score because of the *carried by Jeff* state of the *milk* entity, activating gate $g_4^{(j_2)}$.

2.4.2.6 Memory Chain Configuration

A number of different modes can be used with respect to how the keys of the memory chains are initialised. One possible choice is to initialise the keys to be the entities mentioned in the story. In this case, we associate prior knowledge with the choice of memory keys by tying the weights of such keys to take the embeddings of those entities. While this encourages the model to focus on specific entities and keep track of the changes of their states throughout the story, it requires knowledge of the identities of those entities a priori.

It is also possible to randomly initialise the keys and let the model learn to decide what aspects of the story to focus on. This allows for greater flexibility at the cost of less interpretability. With this mode, entities may be represented in a distributed fashion by multiple memory chains at the same time and the focus of an ENTNET is not constrained to a number of pre-defined set of entities, eliminating the need to identify such entities. In fact, the findings of Henaff et al. (2017) suggest that tying key vectors does not always help and may instead damage model performance.

A hybrid mode can also be used with some of the memory keys tied to certain entities and others free to learn to capture aspects deemed of importance to the task. We will show in later chapters that this hybrid mode is indeed beneficial to model performance as it encourages the model to learn complementary aspects of the pre-defined set of entities in Chapter 5.

2.4.2.7 Comparison with Other Types of Memory Networks

Comparison with LSTM and GRU

ENTNETs resemble LSTMs and GRUs in that they are all controlled by gates, but they differ in how information is represented in their memories. While LSTMs and GRUs use a single vector as its memory where elements can interact with each other, ENTNETs have multiple independent memory chains, each comprised of a vector and responsible for keeping track of a single entity, with little interaction between chains. The independence of memory chains in ENTNETs allows the model to store information more stably compared to LSTMs and GRUs, thereby enabling it to capture long-range dependencies.

Comparison with MEMN2N

Although inspired by MEMN2N in its design of a softmax distribution over all memory blocks, the memory mechanism in ENTNETs is completely different from MEMN2N. While update mainly takes place in the memory controller in a MEMN2N by collecting evidence from its memory, an ENTNET directly updates the memory representation of a relevant chain with an independent gating mechanism.

Comparison with NTM and DNC

Memory cells in NTMs and DNCs are generally accessed via read, erase and write heads with a softmax distribution over all memory locations. That is, different locations compete for attention and those receiving little attention will be updated insignificantly, making it difficult to update multiple memory blocks. In contrast, each memory chain in an ENTNET is governed independently by a gate, determining whether the current input is relevant and the memory of that chain should be updated. This implies that multiple memory chains can be updated at the same time, a crucial element in discourse understanding as more than one entity state may change in a single input step (e.g., in the example in Section 2.4.2.5, the sentence *Jeff travelled to the bedroom* should trigger updates for both *Jeff* and *milk* given the *carried by Jeff* state of the *milk* entity).

2.4.3 Stack-based Memory

In contrast to the random access memory-augmented models, another line of research, inspired by the work of Sun et al. (1993) on neural pushdown automaton, has investigated the feasibility of incorporating a more structured type of memory, stack-based memory, into neural network architectures. To this end, Joulin and Mikolov (2015) propose to train an Elman RNN (Elman, 1990) augmented with stack memory to perform simple algorithmic tasks. Similar and concurrent to Joulin and Mikolov (2015) is the work by Grefenstette et al. (2015) on utilising stack memory for transduction problems. Both let their memory controllers learn to leverage external stack memory by performing either push, pop or stay (keep stack memory unchanged) at any given time step.

Stack-based algorithms are a good fit for text processing due to the hierarchical structure often observed in natural language (Grefenstette et al., 2015, Yogatama et al., 2018). In the text domain, this type of memory augmentation has been shown to be particularly effective on tasks requiring non-local syntactic information. Dyer et al. (2015) introduce stack LSTMs to tackle the task of transition-based dependency parsing. In a stack LSTM, there are three stacks: one for storing the input buffer (where the input sequence is pushed into in reverse order), one for representing partial syntactic trees (containing both individual tokens and partial syntactic units which are computed compositionally with recursive neural networks similarly to the work of Socher et al. (2013)), one for tracking the history of parsing actions (which is only ever pushed to but framed as a stack for implementational convenience). While always appending new inputs to the right-most position, a stack LSTM utilises a "stack pointer" to point to the LSTM cell providing the previous memory states. At each step, a controller determines the best action to take (e.g., shift, reduce-left), based on the top elements of the three stacks. At test time, a greedy heuristic is employed for decoding.

More recently, Yogatama et al. (2018) further propose a multipop mechanism, allowing a stack memory controller to perform multiple pop operations at a time. This is particularly effective when dealing with complex nested syntactic structures. Consider the case of a sentence with multiple nested relative clauses, each of which may be a potential source

of distraction as the intervening noun, often the subject of the relative clause, may be of the opposite number to the subject in the main clause. For example, the sentence, *the students who volunteered in the event organised by the university* ***are*** *given a day off*, has two intervening nouns marked with dashed underlining, *event* and *university*. Both differ in number to the subject of the main clause marked with solid underlining *students*. While it is straightforward to see the subject-verb agreement when given a syntactic parse of the sentence, it is highly confusing to a structure-insentitive model, such as an LSTM, tasked to predict the number of the verb (Linzen et al., 2016). This type of problem leads to what are known as agreement attraction errors (Bock and Miller, 1991).

To tackle this problem, Yogatama et al. (2018) propose to integrate stack memory into LSTMs and equip it with a multipop mechanism, allowing for greater flexibility. Traditionally, a stack memory controller can choose to either push, pop or stay (keep the stack memory unaltered). With the multipop mechanism, the controller still selects from the above three possible options, but learns to execute the chosen operation after performing k pops. Specifically, at each time step, the states of a stack with capacity K after k pops are denoted PUSH_k, POP_k and STAY_k, with probabilities defined as:

$$p(\text{STAY}_k|\boldsymbol{x}_t, \boldsymbol{M}) = p(\text{POP}_{k-1}|\boldsymbol{x}_t, \boldsymbol{m}_{k-1,0}, \boldsymbol{m}_{k-1,1}) \times p(\text{STAY}_k|\boldsymbol{x}_t, \boldsymbol{m}_{k,0}, \boldsymbol{m}_{k,1}) \tag{2.91}$$

$$p(\text{PUSH}_k|\boldsymbol{x}_t, \boldsymbol{M}) = p(\text{POP}_{k-1}|\boldsymbol{x}_t, \boldsymbol{m}_{k-1,0}, \boldsymbol{m}_{k-1,1}) \times p(\text{PUSH}_k|\boldsymbol{x}_t, \boldsymbol{m}_{k,0}, \boldsymbol{m}_{k,1}) \tag{2.92}$$

$$p(\text{POP}_k|\boldsymbol{x}_t, \boldsymbol{M}) = p(\text{POP}_{k-1}|\boldsymbol{x}_t, \boldsymbol{m}_{k-1,0}, \boldsymbol{m}_{k-1,1}) \times p(\text{POP}_k|\boldsymbol{x}_t, \boldsymbol{m}_{k,0}, \boldsymbol{m}_{k,1}) \tag{2.93}$$

where $p(\text{POP}_{-1}|\boldsymbol{x}_t, \boldsymbol{m}_{k-1,0}, \boldsymbol{m}_{k-1,1}) = 1$ and $p(\text{POP}_{K+1}|\boldsymbol{x}_t, \boldsymbol{m}_{K,0}, \boldsymbol{m}_{K,1}) = 0$, $\boldsymbol{m}_{k,0}$ and $\boldsymbol{m}_{k,1}$ are the top two elements of the stack after k pops. Note that in contrast to previous works where the stack operations are determined by the controller solely based on the top element in the stack, this model takes into account the top two elements, making it aware of the future state the stack would likely be in.

The final representation of the stack memory can be computed as:

$$\boldsymbol{M} = \sum_{k=0}^{K} p(\text{STAY}_k|\boldsymbol{x}_t, \boldsymbol{M})\boldsymbol{M}_{\text{STAY}_k} + p(\text{PUSH}_k|\boldsymbol{x}_t, \boldsymbol{M})\boldsymbol{M}_{\text{PUSH}_k} \tag{2.94}$$

where $\boldsymbol{M}_{\text{STAY}_k}$ and $\boldsymbol{M}_{\text{PUSH}_k}$ are the stack states after k pops.

The output representation is computed as a linear transformation of the LSTM hidden state $\boldsymbol{h}_t$ and the top element in the stack $\boldsymbol{m}_t$:

$$\tilde{\boldsymbol{h}}_t = \boldsymbol{W}_h \boldsymbol{h}_t + \boldsymbol{W}_m \boldsymbol{m}_t \tag{2.95}$$

where $\boldsymbol{W}_h$ and $\boldsymbol{W}_m$ are trainable weight matrices. This output representation $\tilde{\boldsymbol{h}}_t$ can then be used to compute the probability of the next word for language model training.

2.4.3.1 Comparison with Other Types of Memory Networks

The main difference of this class of stack-based memory models, as compared to other types of memory networks, lies in the way memory is accessed and manipulated. The controller in a stack-based memory model is trained to perform a series of stack-related operations (i.e., push, pop, stay). In contrast, memory controllers in MEMN2Ns and ENTNETs operate based on the content- and/or location-based addressing.

2.4.4 Summary of Memory-augmented Models

In Table 2.3, we list the models described in this chapter and compare them across a few different dimensions: memory type, addressing mechanisms, training, and controller duty. From the memory type perspective, we classify the models into two major categories: static and dynamic memory, according to whether the memory is read-only or writable. Looking from a different angle, the key difference is where the update takes place, i.e., whether new information is written to the memory itself or to the controller. In the case of MEMNN and MEMN2N, the memory is used as the storage. The memory controller, which, in the context of reading comprehension, takes the value of the embedding of a question, gets updated by collecting relevant evidence from the memory. In this sense, every element in a sequence in a TRANSFORMER can be viewed as an individual controller, collecting relevant information from its surrounding context with self-attention and progressively building up its representation by updating the controller value. In contrast, NTM, DNC,

Model	Memory Type	Addressing Mechanism	Training	Controller Duty
MEMNN	static	content-based	require additional memory location supervision	arg max, collect relevant evidence from memory
MEMN2N	static	content-based	end-to-end	softmax over memory locations collect relevant evidence from memory
TRANSFORMER	static	self-attention	self-supervision then fine-tuning	query-key-value tuple, softmax over sequence elements build up representation from context
NTM	dynamic	content- and location-based	end-to-end	softmax over memory locations with read, erase, and add heads
DNC	dynamic	content- and location-based	end-to-end	softmax over memory locations with read, erase, and add heads
ENTNET	dynamic	content- and location-based	end-to-end (with semantic supervision)	sigmoidal gates, activated at relevant input
stack-based memory	dynamic	N/A	end-to-end, can also be trained with REINFORCE (Williams, 1992)	stack operations push, pop, stay

Table 2.3 Comparison of memory-augmented models described in Chapter 2.

ENTNET and stack-based memory models all update their storage by directly writing to the their memory components, either with a softmax distribution over all memory locations, or a sigmoid gating function, or stack-related operations.

There are two major categories of memory addressing mechanism: content- and location-based addressing. While content-based addressing is based on measuring the similarity between the contents of two elements (e.g., the dot product between the vector representations of two elements), location-based addressing is more concerned with the memory locations of consecutive accesses (either to contiguous memory blocks, e.g., NTM or shifting between locations, e.g., DNC). Another type of location-based addressing is implemented in ENTNET where the location term reflects the identity of the entity tracked by the current memory chain. In contrast, the content term in a ENTNET tracks the state, as opposed to the identity, of the entity. The self-attention mechanism in TRANSFORMERS is similar to content-based addressing in that it measures the similarity between the vector representations of a query and key, both of which are projected from the input.

Apart from MEMNN, the other models in Table 2.3 can all be trained end-to-end. BERT, an encoder based on TRANSFORMER, can be trained in an unsupervised fashion using a masked language modelling objective, which can be viewed as self-supervision. With further fine-tuning, BERT demonstrates strong performance on a wide range of downstream NLP tasks (Devlin et al., 2019). In Chapter 5, we show the benefits of incorporating semantic supervision with ENTNET, allowing the model to achieve better accuracy on the task of commonsense story understanding.

The last dimension of Table 2.3 focuses on how the controller in each model works on its memory. A number of the models (MEMN2N, TRANSFORMER, NTM, and DNC) rely on the softmax function to compute attention weights over all memory locations according to their relevance. In contrast, ENTNETS makes use of sigmoidal gates as each memory chain is independent and stack-based memory models learn to perform stack-related operations.

2.5 NLP Applications of Memory-augmented Models

Many attempts have been made to apply memory-augmented models to the text domain, ranging from language modelling (Tran et al., 2016, Cheng et al., 2016), to summarisation (Jiang and Bansal, 2018, Kim et al., 2019), to even poetry generation (Zhang et al., 2017b). Among a broad array of potential applications, many have focused on three tasks: (1) machine translation, (2) question answering, and (3) text categorisation (in particular, sentiment analysis). In this section, we briefly review recent studies on these tasks.

2.5.1 Machine Translation

The task of machine translation (MT) can be cast as a sequence-to-sequence (seq2seq) learning problem where the goal is to transform an input sequence in a source language to its translation in a target language. MT systems typically take the form of an encoder and a decoder, responsible for the input and output sequence respectively. This encoder-decoder framework may be further equipped with attention, enabling the decoder to collect information across the entire input sequence while generating the next target word in the output sequence. The most popular model representative of this framework is RNNSEARCH by Bahdanau et al. (2015). While this has clear benefits over a vanilla encoder-decoder model (without attention) (Cho et al., 2014b, Sutskever et al., 2014), the memory capacity is still somewhat limited (Jiang and Bansal, 2018). To this end, many have made progress in enhancing modelling capacity by augmenting memory to RNNSEARCH-based models.

Wang et al. (2016) propose to integrate memory buffer with content-based addressing into RNNSEARCH. This memory buffer is designed to be fixed in size and accessed by three attention heads: read/erase/add, inspired by the work of Graves et al. (2014). At test time, the decoder incorporates information from both the input sequence via attention as well as the memory buffer via read heads. This model demonstrates strong performance on the task of Chinese-English translation, outperforming a number of competitive baselines.

Feng et al. (2017) address the limitations of MT systems when dealing with infrequent words and word pairs. More specifically, memory is used as storage of mappings between words in the source and target language, which can be learned by conventional statistical machine translation tools such as GIZA++ (Och and Ney, 2003). The proposed method improves performance on the Chinese-English translation task with particular gains in cases involving out-of-vocabulary words.

Maruf and Haffari (2018) consider document-level translation and frame the task as a structured prediction problem. This approach takes into account the interdependencies between the current sentence and all previous sentences (in both the source and target languages) in the same document by storing them into its memory. Experimental results on document-level translation on three language pairs show significant improvements over previous work.

A closely related task to machine translation is generation with applications of memory-augmented models to Chinese poetry generation (Zhang et al., 2017b), conversational response generation (Tian et al., 2019), answer generation (Fu and Feng, 2018), and task-oriented dialog response generation (Chen et al., 2019a, Lin et al., 2019).

2.5.2 Question Answering

Question answering (QA) is another task to which memory-augmented models have been widely applied. To accommodate the large contexts (a single or multiple documents) typically involved in a QA task or the vast amount of world knowledge needed to answer questions, many have proposed the use of memory to store such information.

Das et al. (2017) present a universal schema to merge knowledge from heterogeneous sources: triples from knowledge bases and unstructured web text. This allows the model to exploit the benefits of both knowledge sources and results in much improved performance on SPADES (Bisk et al., 2016), a real-world fill-in-the-blank cloze-styled question answering dataset, outperforming previous state of the art approaches by a large margin.

Back et al. (2018) introduce MEMOREADER for large-scale reading comprehension (RC), particularly focusing on lengthy documents with long-range dependencies. MEM-

OREADER borrows ideas from the work of Graves et al. (2016) in its implementation of the memory controller, enabling it to reason over long documents. Also equipped with self-attention and co-attention, the proposed method achieves substantial improvements over a number of competitive baselines on three RC datasets, with performance gains particularly pronounced on tasks involving lengthy documents, such as TriviaQA (Joshi et al., 2017) and QUASAR-T (Dhingra et al., 2017).

Lai et al. (2019) present a gated self-attention memory network for better answer selection. Memory in each hop is progressively constructed from the previous hop through the gated self-attention mechanism. The model can be trained using transfer learning, allowing it to exploit large-scale web-crawled data for pre-training and then to be further fine-tuned on the dataset of interest. Experimental results demonstrate the utility of the proposed method, establishing a new state of the art on TrecQA (Wang et al., 2007) and WikiQA (Yang et al., 2015).

Han et al. (2019) consider a different setting of question answering where the input takes the form of streaming data of arbitrary length. This requires a model to manage external memory storage by discarding and replacing memory entries to make room for new input with information deemed of value to the task. The proposed memory-augmented architecture can be trained using reinforcement learning with either A3C (Mnih et al., 2016) or REINFORCE (Williams, 1992). The authors report significant improvements over rule-based memory scheduling policies.

More recently, there has been increasing interest in applying key-value memory networks (Miller et al., 2016) to question answering tasks. Chen et al. (2019b) make use of a key-value memory network to store triples in a knowledge base and design a novel query updating mechanism to incorporate both the query in the previous memory hop and the weighted sum of the attended keys and values. Xu et al. (2019) propose the use of a key-value memory network to store answer candidates whose entities are close to the main topic entity in the question (including all connected entities within h hops). This is further coupled with bi-directional attention, making the model aware of the interrelationships between questions and knowledge base triples. Both models are tested on

the WebQuestions dataset (Berant et al., 2013) and achieve competitive results over a number of strong baselines.

2.5.3 Text Categorisation

Many text categorisation tasks focus on classifying the sentiment polarity of a given piece of text, a task commonly known as sentiment analysis (SA). An even more fine-grained variant of SA is the task of aspect-based sentiment analysis (ABSA) where the goal is to identify sentiment polarity towards specific aspects. For example, in the sentence *tasty food but terrible service*, the *food* aspect receives a *positive* rating while the *service* aspect is *negative*. In this section, we review a number of recent memory-augmented approaches applied to these two tasks.

Sentiment Analysis

Liu et al. (2016) augment LSTMs with external memory and NTM-style control mechanisms. More precisely, this is achieved by three memory accessing heads responsible for reading, erasing, and writing. Their model is further coupled with either shared global or local-global hybrid memory, making it compatible with multi-task learning, which has been shown to be effective in boosting the performance on the primary task with the help of auxiliary tasks.

Another application of MEMN2Ns to SA is presented in the work of Dou (2017). To better model the sentiment of a given review of product p written by user u, their model takes into account the influence of: (1) reviews (written by other users) of the same product p, and (2) reviews (of other products) written by the same user u. This is achieved by storing such reviews in a MEMN2N and then performing multi-hop reasoning over the memory bank with attention. This model compares favourably with existing approaches on a number of SA datasets.

To solve the data sparsity problem in many short text classification tasks, Zeng et al. (2018) propose a novel topic memory mechanism to store latent topic representations which are indicative of class labels. This is achieved through the use of a variational

auto-encoder-based neural topic model for inducing latent topics and then projecting such latent representations into the key and value space of a memory network to support the final classification decision. While not directly tested on any SA datasets, the proposed model shows impressive results on a number of short text classification tasks and can be potentially adopted to SA.

Aspect-based Sentiment Analysis

Tang et al. (2016b) apply MEMN2Ns to the task of ABSA where the word embeddings of a context sentence are stored in the memory with the question being the aspect term. A similar approach by Chen et al. (2017) improves this model by processing a context sentence with a BI-LSTM and storing the contextualised representation in the memory. Both methods are tested on the SemEval-2014 ABSA dataset (Pontiki et al., 2014), advancing the state of the art on this task.

Wang et al. (2018) identify an important problem of MEMN2Ns when applied to ABSA, pointing out simply improving attention will not fix it. Instead, they propose six alternative target-sensitive memory networks, each with its own characteristics and evaluate their effectiveness also on the SemEval-2014 ABSA dataset.

Majumder et al. (2018) take this one step further by storing inter-aspect dependent sentence representations in a MEMN2N. The inter-aspect dependent representation of a sentence is obtained by processing its aspect-aware representation with a GRU. The aspect-aware representation is, in turn, produced by the output of another GRU taking as input the concatenation of word and aspect embeddings. This method achieves impressive in-domain as well as cross-domain results on the SemEval-2014 ABSA task.

2.6 Narrative and Discourse Comprehension

The understanding and comprehension of narrative and discourse has long been a challenging task in artificial intelligence (Winograd, 1972, Turner, 1994, Schubert and Hwang, 2000). The difficulties arise from a number of perspectives, requiring a collection of

demanding capabilities of a model, such as reasoning capabilities (Weston et al., 2016), commonsense knowledge (Mostafazadeh et al., 2016) and capturing relationships between distant entities (Hermann et al., 2015). To this end, many have constructed datasets, focusing on various aspects, to facilitate the development of narrative and discourse comprehension.

In this section, we first review a number of related datasets in each subsection then limit our focus to the tasks most relevant to and actually used in this thesis. We divide this section into two major parts: (1) narrative and story understanding (Section 2.6.1), and (2) discourse comprehension (Section 2.6.2).

2.6.1 Narrative and Story Understanding

2.6.1.1 Related Narrative Tasks

Reading Comprehension

A large body of work has focused on the problem of reading comprehension, focusing on answering questions in regards to a given document. To this end, Richardson et al. (2013) construct an open-domain reading comprehension task, named MCTest. Although this corpus demands various degrees of reasoning capabilities from multiple sentences, its rather limited size (660 paragraphs, each associated with 4 questions) renders training statistical models infeasible (Chen et al., 2016).

CNN/Daily Mail (Hermann et al., 2015) has been used for the task of machine reading formalised as a problem of text extraction from a source conditioned on a given question. However, as pointed out by Chen et al. (2016), this dataset is not only noisy but also requires little reasoning and inference, which is evidenced by a manual analysis of a randomly selected subset of the questions, showing that only 2% of the examples call for multi-sentence inference.

The Children's Book Test (CBT), constructed by Hill et al. (2015), is a dataset extracted from Project Gutenberg.[2] The use of children's books ensures that the narrative is straight-

[2]https://www.gutenberg.org/

forward without being too complex, making context important. Each example in this dataset consists of a context of 20 consecutive sentences in the original story and a question built from the 21st sentence with one word removed. The task is then defined as filling the slot of the missing word, among 10 possible candidates, in the question given the context. Sukhbaatar et al. (2015) claimed that increasing the number of hops is crucial for performance improvements on some tasks, which can be seen as enabling MEMN2N to accommodate more supporting facts, making any such performance boost more pronounced on tasks requiring complex reasoning. However, Hill et al. (2015) reported little improvement in performance by stacking more hops, and ultimately chose a single-hop MEMN2N. This suggests that multi-sentence based reasoning is not essential for this dataset.

Perhaps the most popular datasets in the reading comprehension community are SQuAD (Rajpurkar et al., 2016) and SQuAD v2.0 (Rajpurkar et al., 2018). SQuAD v1.0 defines a question answering task where a model must find the answer to each question in the given passage. The improved version, SQuAD v2.0, builds on the previous work and further extends it by adding potentially unanswerable questions, to which a model is supposed to return "no answer". That is, the answer, if there is one, is a continuous span of text in the original document; otherwise, the model should abstain from answering. SQuAD v2.0 consists of over 100,000 questions with a fair portion of them being unanswerable questions written by crowdworkers. At the time of writing, most competitive systems on SQuAD make use of BERT with some types of complex attention mechanisms.[3]

Scripts and Narrative Events

Scripts played a central role in the research of natural language understanding in the 1970s and 1980s (Schank and Abelson, 1977). Structured knowledge can be represented by scripts, such as event sequences and their participants, forming "narrative chains". A number of language understanding tasks, e.g., question answering, coreference resolution and textual entailment, may benefit from such narrative chains to capture their critical

[3] https://rajpurkar.github.io/SQuAD-explorer/, accessed: May 8th, 2019.

background knowledge. To this end, Chambers and Jurafsky (2008) introduce a new narrative cloze task. Specifically, given a sequence of events where one event has been removed, the goal is to predict the missing verb and typed dependency of that removed event. While Chambers and Jurafsky (2008) admit that there are instances where it is even impossible for humans to identify the missing events, shallow models with little commonsense understanding have been making solid progress, which is problematic as they merely learn to beat the shallow test (Mostafazadeh et al., 2016).

Also related is the Spinn3r dataset by Burton et al. (2009). The dataset is built from online Weblogs consisting of 44 million blog posts. Despite personal blogs providing good sources for commonsense causal information (Gordon and Swanson, 2009, Manshadi et al., 2008), the noisy nature and discourse complexities may potentially require many irrelevant steps to get to the core task of narrative comprehension (Mostafazadeh et al., 2016), making the task unnecessarily difficult and less than ideal to serve as a benchmark to evaluate the true ability of a model to understand narrative.

Language Inference

The Winograd Schema Challenge (WSC) is another challenging task, developed by Levesque et al. (2012) to evaluate the capabilities of a model to perform pronoun disambiguation, requiring a deep semantic understanding of the text. It takes the form of a pair of almost identical sentences, with the difference being only one or two words. Despite the minor difference, it may cause the resolution of anaphora to end up in completely opposite ways. For example, consider the the sentence pair, *The woman held the girl against **her** chest/will*, where the pronoun in question is in bold and the only difference underlined (the last word in each sentence). The difference in the last word being either *chest* or *will* drastically changes the antecedent to which the pronoun **her** refers. Each pair is carefully designed so that human readers, in most cases, can easily solve it and would not even notice the ambiguity, whereas simple models are likely to struggle. While highly challenging, WSC is prohibitively small, consisting of 273 test instances only, with no training or validation set.

Recent state-of-the-art approaches show that large-scale language model pre-training is helpful, achieving a success rate of over 70% (Radford et al., 2019).

More recently, Zellers et al. (2018) introduce the SWAG dataset, a large-scale corpus for grounded commonsense inference. There are 113k instances in the dataset, all extracted from two video caption datasets: ActivityNet Captions (Heilbron et al., 2015, Krishna et al., 2017) and the Large Scale Movie Description Challenge (Rohrbach et al., 2017). Instances are constructed from adjacent video captions, guaranteeing that the follow-up event is physically plausible. Questions in this dataset come in the form of multiple choice, focusing on the follow-up event, with the gold event being the true answer and three adversarially generated counterfactuals. Each counterfactual distractor is generated using an LSTM, pretrained on BookCorpus (Zhu et al., 2015) and then further finetuned on the video caption datasets. Despite the claim of SWAG being difficult for a number of state-of-the-art natural language inference models, BERT, when pretrained on large-scale corpora with a language modelling objective and then finetuned on SWAG, achieves an impressive accuracy of 86.3%, already close to that of human performance (88.0%) (Devlin et al., 2019).

2.6.1.2 Reasoning-focused Story Understanding

Synthetic tasks are not uncommon in the machine learning community. The XOR problem, for instance, provides a perfect example to motivate the development of neural networks with non-linear activation functions (Minsky and Papert, 1969, Rumelhart et al., 1986a). To this end, Weston et al. (2016) construct an artificially generated dataset, named bAbI, to test the reasoning and inference capabilities of a model in a story understanding setting. The dataset takes the form of a natural language-based question answering task. Specifically, each instance consists of: (1) a story comprised of an ordered list of supporting statements $(x_1, \ldots, x_n)$, with each sentence describing the states of some entities, events, or interactions taking place between them; (2) a question in regards to the entities mentioned in the story; and (3) an answer to the question, typically a single word or short phrase. A total of 20 tasks are included in the dataset, each with a distinct emphasis on a particular

Mary moved to the bathroom.
Sandra journeyed to the bedroom.
Mary got the football there.
John went to the kitchen.
Mary went back to the kitchen.
Mary went back to the garden.
Where is the football?
Answer: garden

(a) bAbI task 2: two supporting facts.

Mary moved to the bathroom.
Sandra journeyed to the bedroom.
Mary went back to the bedroom.
Daniel went back to the hallway.
Sandra went to the kitchen.
Daniel went back to the bathroom.
Is Daniel in the office?
Answer: no

(b) bAbI task 6: yes/no questions

The suitcase is bigger than the container.
The container fits inside the box.
The chest is bigger than the chocolate.
The suitcase fits inside the box.
The chest fits inside the box.
Does the chocolate fit in the box?
Answer: yes

(c) bAbI task 18: size reasoning.

The office is east of the hallway.
The kitchen is north of the office.
The garden is west of the bedroom.
The office is west of the garden.
The bathroom is north of the garden.
How do you go from the kitchen to the garden? Answer: south, east

(d) bAbI task 19: path finding

Figure 2.5 Examples of bAbI tasks.

type of reasoning-focused narrative, some much more difficult than others (for example, 3 supporting facts, positional reasoning, and path finding). Each task includes a training and test set and the validation set is often randomly sampled from the training set (10%). Figure 2.5 presents examples of a few bAbI tasks where the correct answer is highlighted in red. Note that in Figure 2.5d, the answer is comprised of multiple words: *south, east*. In this case, this multi-word answer is considered as a single phrase and accounts for a single entry in the answer vocabulary. It should also be noted that not every supporting statement is necessarily relevant to the question. In fact, only a selected subset of them are truly informative to answering the question, the rest are simply irrelevant distractors.

The bAbI dataset, although synthetic, has been a challenging collection of tasks for a number of models to successfully solve (success defined as accuracy $\geqslant 95\%$, an arbitrarily chosen threshold by Weston et al. (2016)) (Sukhbaatar et al., 2015, Xiong et al., 2016, Liu

#	Story
1	Ricky fell while hiking in the woods.
2	He was terrified!
3	He thought he had fallen into a patch of poison ivy.
4	Then he used his nature guide to identify the plant.
	Correct ending: He was relieved to find out he was wrong.
	Incorrect ending: Ricky was soaking wet from falling in the pond.

#	Story
1	Sam loved his old belt.
2	He matched it with everything.
3	Unfortunately, he gained too much weight.
4	It became too small.
	Correct ending: Sam went on a diet.
	Incorrect ending: Sam was happy.

Table 2.4 Examples of the Story Cloze Test.

and Perez, 2017). It is not until recently have models become capable of solving all 20 tasks (Seo et al., 2017). More importantly, bAbI often serves as a crucial sanity-checking step, providing motivation for further validation on real-world datasets as in the works of Graves et al. (2016), Henaff et al. (2017), and Banino et al. (2020).

2.6.1.3 Commonsense Narrative Understanding

To better evaluate the effectiveness of a model's ability to understand causal and correlational relationships between events, Mostafazadeh et al. (2016) develop a dataset named Story Cloze Test. As the name suggests, the task takes the form of a cloze test but differs in that the goal is to predict the most coherent ending to a given story out of two alternative ending options. Stories in this dataset, while describing ordinary activaties, are constructed with a rich set of causal and temporal commonsense relations between daily events. With this goal in mind, Mostafazadeh et al. (2016) asked crowdworkers on Amazon Mechanical Turk to write novel stories focusing on logicality and storiness, following the definition of Forster (1927) and Bailey (1999).

Two example stories are shown in Table 2.4. The first example is about a hiking scenario where the main character, *Ricky*, fell into a patch of ivy of whose toxicity he was unsure, leading him to turn to his nature guide for help in identifying the plant. While the incorrect ending may be acceptable and seems fitting to the scenario of being "soaking wet" after *Ricky* fell, it is inappropriate because it does not constitute a conclusive ending to the story and has low continuity relating to the immediately preceding sentence, as the focus was on the question of whether the plant is poisonous. The correct ending, on the other hand, makes more sense by revealing the reaction of *Ricky* after looking the plant up in his nature guide and that he realised what he initially thought was poisonous was in fact harmless, much more consistent with the main narrative of the story.

In the second example, the story describes a sequence of daily events of *Sam* gaining weight, making the belt he loved too small for him. While *Sam* was very fond of his belt, the negation in sentiment, indicated by the use of *unfortunately* in the third sentence, along with the subsequent sentence suggesting the belt is now too small for *Sam*, implicitly implies that to be able to match the belt with everything again, Sam will need to lose some weight. For a model to be able to pick the correct ending option, not only is it required to understand that *Sam*'s recent gaining of weight is the cause for the belt becoming too small for him, the model must also be aware that *went on a diet* is equivalent to losing weight, so as to make the belt fit.

The Story Cloze Test dataset is split into a development set and a test set, each comprised of 1,871 4-sentence stories. Each story is coupled with a pair of alternative ending options, only one of which is coherent with the narrative of the story. Also included in the data release is a much larger training set, consisting of 100k 5-sentence ROC stories with coherent endings only. This can be used to train a language model of coherent stories, of which a number of recent models make use.

More recently, efforts have focused on leveraging large-scale language model pre-training with unsupervised learning (Radford et al., 2018, Sun et al., 2019b) and shown substantial improvements over traditional methods.

2.6.2 Discourse Comprehension

2.6.2.1 Related Discourse Tasks

The Ubuntu Dialogue Corpus (UDC), developed by Lowe et al. (2015), consists of a total of over 7 million uterances. In contrast to tasks focusing on structured dialogues using slot-filling representations (Singh et al., 2002, Williams et al., 2013, Henderson et al., 2014), UDC targets unstructured dialogues in the format of multi-turn conversations. The dialogues are extracted from the Ubuntu chat logs,[4] mostly two-person conversations on the topic of technical support for a variety of topics in regards to the Ubuntu operating system. Although it is possible that multiple users may all have different answers to the same initial question, such multi-party conversations are broken down into sub-conversations between the first user and each user who replied, and treated as a separate dialogoue. That is, the interactions and complex discourse structures are intentionally left out in this dataset.

Concurrent to our work on thread discourse structure analysis in Chapter 4, Zhang et al. (2017a) introduced a similar discourse dataset based on Reddit,[5] targeted at better analysing the coarse discourse structures of online discussions. The goal is to categorise the discourse acts between utterances to enable better understanding of online forum-based discussions. While made publicly available, a portion of the dataset is no longer accessible due to various reasons, such as removed posts or accounts, making it difficult to establish a reliable baseline.

2.6.2.2 Dialogue

Dialog State Tracking

One important concept in the process of understanding a dialog, a goal-oriented one in particular, is "dialog state", a term linked to the full representation of a user's intent at any point in a dialog (Henderson et al., 2013). The process of tracking the state of a dialog is

[4] http://irclogs.ubuntu.com/ from 2004 to 2015.
[5] https://www.reddit.com/

simply termed "dialog state tracking". The task plays an essential role in the success of a dialog system in detecting the user intent and informing the system's actions (Henderson et al., 2014). Effective dialog systems are often equipped with such components, enabling them to gather evidence over the sequence of utterances and update the dialog state based on new observations.

To test the effectiveness of such dialog state trackers, the research community has been organising an on-going series of dialog state tracking challenges (DSTC), dating back to 2013 when the first challenge was released (Williams et al., 2013). While originally coined to stand for "Dialog State Tracking Challenges", the term DSTC has been re-branded to "Dialog System Technology Challenges" since the sixth edition of DSTC in 2017.

Of particular interest to this thesis is the second edition of the challenge, DSTC-2. The dataset of DSTC-2 is built following the work of Williams et al. (2013) with a few key differences. First, DSTC-2 is set in a restaurant domain as opposed to a bus timetable information querying scenario in the first edition of the challenge. Also, users are now permitted to change their intents during the course of the dialog, that is, interest in a Chinese restaurant expressed at the beginning of a dialog may switch to Italian food at the end. Lastly, dialog state has been enriched to include not only pairs of slot and value attributes of user goals but also their search method and user queries (e.g., requests for the address or phone number of a restaurant).

An example of DSTC-2 is shown in Table 2.5. In this example, the system agent initiates the dialog by greeting the user, followed by a question regarding the preferred food type. The user then expresses interest in Swedish food in the moderate price range, which is reflected in the slot values on the right half of the table. Notice that the area slot is unspecified at this point and the value of it is therefore kept unchanged. Next, the agent informs the user that there is no restaurant matching the specified search criterion, at which point the values to the three slots remain intact. Lastly, the user decides to switch to Asian food and the food slot value should be updated accordingly (food value to Asian) while the others stay untouched.

User	Agent	Slots		
		Food	Price	Area
	Hello and welcome	–	–	–
	What kind of food would you like?	–	–	–
Moderately priced Swedish food		Swedish	moderate	–
	Sorry there is no Swedish restaurant in the moderate price range	Swedish	moderate	–
How about Asian food?		Asian	moderate	–

Table 2.5 An example of DSTC-2. User and agent utterances are left and right aligned on the left half of the table. Slots and values are on the right half of the table. – indicates not specified.

The data collection process involves a dialog system with two main components: a dialog manager and a speech recogniser, each chosen among a pool of alternative options (3 dialog managers and 2 speech recognisers, resulting in 6 different configurations). The data is then collected by asking paid Amazon Mechanical Turkers to call the dialog systems with the goal of finding a restaurant matching certain constraints, such as area, price range and food type. To make the task more realistic and the dialogs more complex, turkers were sometimes asked to change their minds to simulate real-world scenarios. In addition to the N-best automatic speech recognition (ASR) hypotheses, the transcript of each utterance is also provided as part of the data release. While it is common practice to take the N-best ASR hypotheses as input, in the scope of this thesis, we simplify the task and limit our focus to the transcripts as the goal is not to advance the state of the art in dialog state tracking but rather to use DSTC-2 as a testbed to evaluate the effectiveness of the proposed methods in later chapters.

Dialog bAbI

While the traditional slot-filling method for goal-oriented dialog agents has proven reliable, the amount of work required to scale to a new domain is enormous, severely limiting the applicability, calling for a paradigm shift.

To this end, Bordes and Weston (2017) develop a goal-oriented dialog dataset named Dialog bAbI (see Table 2.6 for an example). The dataset consists of a set of tasks, which

can be grouped into 6 categories, each focusing on a particular objective: (1) issuing API calls, (2) updating API calls, (3) displaying options, (4) providing extra-information, (5) conducting full dialogs (the aggregation of the first 4 tasks), and (6) Dialog State Tracking Challenge 2 corpus (DSTC-2). All tasks are set in a restaurant reservation scenario but differ in their source. While the first 5 tasks are synthetically generated, the last one is derived from the second Dialog State Tracking Challenge (Henderson et al., 2014), comprised of real human-bot conversations. Text in Dialog bAbI takes the form of conversations between a user and a bot and is artificially generated, in the case of the first 5 tasks, based on a knowledge base containing facts about restaurants along with their associated properties (7 types, such as location and price range). The goal of this dataset is to evaluate the capacity of an end-to-end trained system to conduct dialog with various objectives. A user and a dialog system take turns in the conversation with utterances alternating between a user request and a system response. In order to test how well a model can generalise to previously unseen entities in the training set, Bordes and Weston (2017) further construct a separate yet more challenging set of test tasks with the name of out-of-vocabulary (OOV) test set, indicating that a fair portion of entities do not occur in the training set. Task 6, on the other hand, while also in the restaurant booking domain, is composed of real human-bot conversations extracted from DSTC-2. It is important to notice that the answers in this dataset may no longer be a single word but can be comprised of multiple words.

2.6.2.3 Web Forum Thread Discourse Structure

While dialog, in the traditional sense, is often used to refer to conversations between two or even multiple parties, with the proliferation of the Internet, another important type of discourse takes place in the cyberspace. In this regard, online forums, such as CNET[6] and Reddit,[7] have become a crucial source of information when surfing the web. Such forums cover a wide range of subjects ranging from dialy topics commonly seen on Reddit, to

[6]https://www.cnet.com/forums/
[7]https://www.reddit.com/

		User	Agent
Task 1	Issuing API calls	Good morning	
			Hello what can I help you with today
		I'd like to book a table for two in London in a moderate price range	
			I'm on it
			Any preference on a type of cuisine
		With Spanish cuisine	
			OK let me look into some options for you
			api_call Spanish London two moderate
Task 2	Updating API calls	Instead could it be with British food	
			Sure is there anything else to update
		No	
			OK let me look into some options for you
			api_call British London two moderate
Task 3	Displaying options	*resto_london_moderate_british_5stars R_phone resto_london_moderate_british_5stars_phone* *resto_london_moderate_british_5stars R_cuisine british* *resto_london_moderate_british_5stars R_address resto_london_moderate_british_5stars_address* *resto_london_moderate_british_5stars R_location london* *resto_london_moderate_british_5stars R_number two* *resto_london_moderate_british_5stars R_price moderate* *resto_london_moderate_british_5stars R_rating 5* *resto_london_moderate_british_3stars R_phone resto_london_moderate_british_3stars_phone* ⋮	
			What do you think of this option: *resto_london_moderate_british_5stars*
		I love that	
			Great let me do the reservation
Task 4	Providing extra-info	What is the phone number of the restaurant	
			Here it is *resto_london_moderate_birtish_5stars_phone*
		Thank you	
			Is there anything I can help you with
		No thank you	
			You're welcome

Table 2.6 An example of the tasks in Dialog bAbI. Task 5 is the aggregate of the first 4 tasks. *Italics* indicate API calls made by the agent to query the knowldege base or the results returned by the knowledge base. User and agent utterances are left and right aligned, respectively.

in-depth technical discussions within small special interest groups on CNET and Ubuntu Forums.[8]

On the one hand, the increasing participation in such forums makes them valuable platforms when seeking answers to questions already solved. On the other hand, because of the popularity, the amount of data accumulated over time has also become prohibitively large, rendering the extraction of relevant information tedious. For example, one may need to go through a large volume of irrelevant posts in a thread before eventually arriving at the answer. This has motivated research in this area and previous works have constructed a number of datasets to facilitate the development of models to analyse discourse structure of web forums (Kim et al., 2010, Lowe et al., 2015, Zhang et al., 2017a).

In this thesis, we make use of the dataset introduced by Kim et al. (2010) on thread discourse structure prediction. Specifically, the task, given an ordered list of posts in a thread from an online forum, requires one to identify which posts the current one is a direct reply to, i.e., to identify the hidden linking structure between posts. Additionally, the task is also concerned with the relationship between linked posts, i.e., to classify the dialog act of each link among many possible options, e.g., whether the current post is asking for further clarification or actually answering the question. An example of this task is illustrated in Figure 2.6. In this example, posts with "in reply to" relationships are linked with arrows and the type of reply marked on the edges. Notice that it is possible for a post to be responding to multiple previous posts in the same thread, e.g., post #6 is connected to both posts #1 and #5.

A sensible modelling choice, as demonstrated in previous works of Kim et al. (2010), Wang et al. (2011),and Wang (2014), is to model this task with conditional random fields (CRFs) (Lafferty et al., 2001). This requires the task to be treated as one of sequence tagging with the labels being the combined tags of each post. The input to a CRF can be a collection of structural and semantic features to enable the sequential classifier to learn the dynamics between neighbouring posts. Another possible and effective approach is based on a constrained dependency parser where posts in a thread are perceived as

[8]https://ubuntuforums.org/

nodes in a dependency graph with the constraint that links must connect to preceding posts (Wang et al., 2011, Wang, 2014).

2.7 Summary

In this chapter, we have thoroughly reviewed a number of important memory-augmented models and their application to a wide range of tasks. A comparison of all the models described in this chapter is presented in Table 2.3. Additionally, we have also presented a comprehensive study over many discourse understanding tasks, most relevant to or directly used in this thesis.

Next, we address the three research questions presented in Section 1.2, focusing on improving memory capacity (Chapter 3), enhancing modelling capacity by memory augmentation (Chapter 4), and lastly relaxing model limitations (Chapter 5).

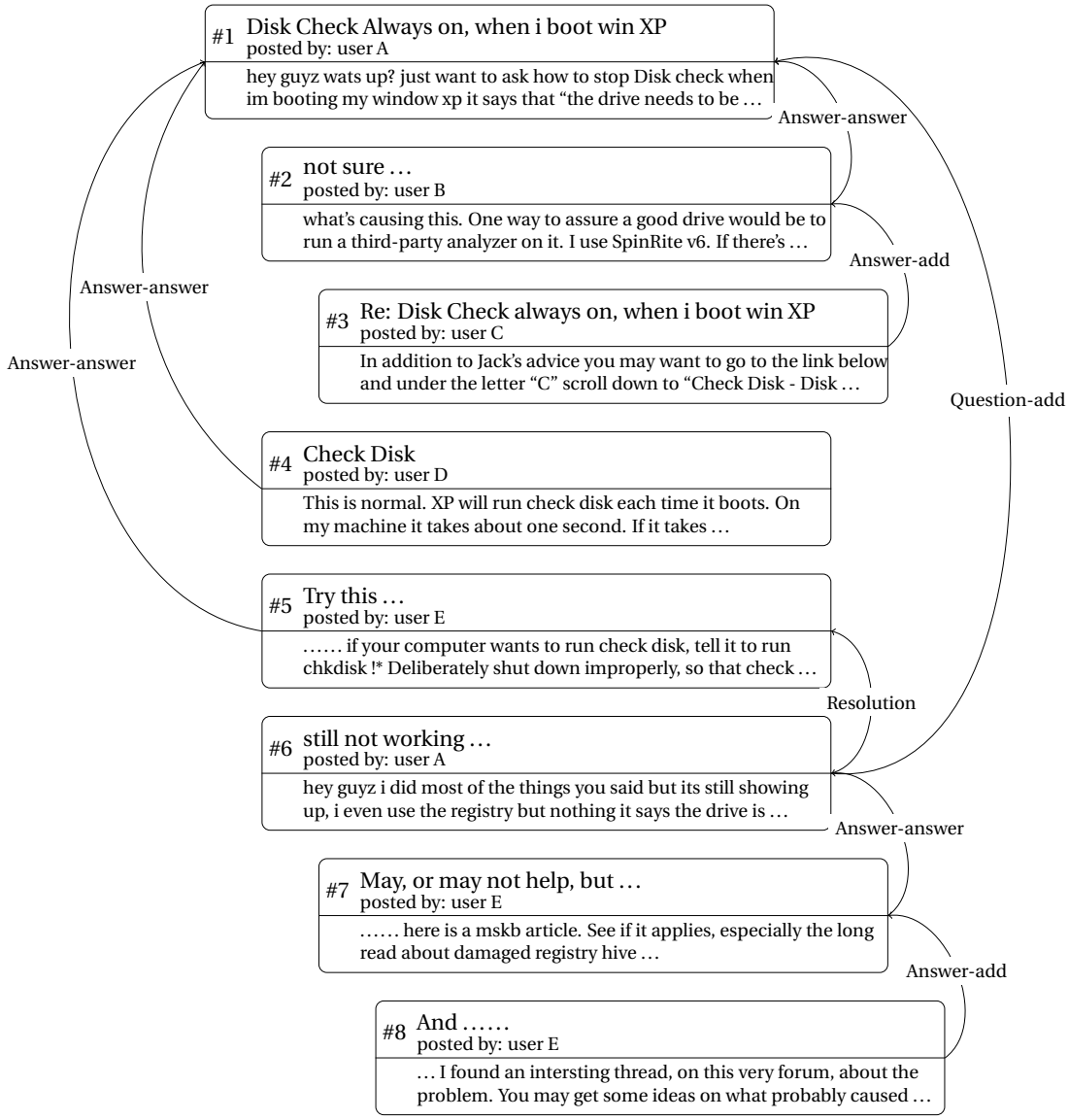

Figure 2.6 An example of the web forum thread discourse structure prediction task. The "in reply to" relationships between posts are implied by the links with arrows starting from the reply and pointing to the post it answers. Captions on edges indicate the type of reply.

Chapter 3

Improving Memory Capacity with Unified Weight Tying

To understand a discourse, it is sometimes required to answer questions while reasoning over multiple supporting facts, a long-standing goal of artificial intelligence. To this end, remarkable advances have been made, focusing on reasoning over language-based stories. In particular, end-to-end memory networks (MEMN2N) have achieved state-of-the-art results over such tasks. A MEMN2N stores the entire sequence of sentences of a story in its memory and relies on a memory controller to progressively gather evidence from the memory, allowing it to reason over multiple supporting facts. In the development of MEMN2Ns, two types of weight tying mechanisms were introduced to ameliorate overfitting, namely adjacent (ADJ) and layer-wise (LW) weight tying (Section 2.3.2.3), requiring one to choose between the two. This decision is made even more difficult by their uneven performance on the bAbI dataset, that is, neither of them performs consistently well over all tasks. Additionally, while attempting to reduce overfitting by forcing a number of parameter weight matrices to be shared across memory hops, both weight tying methods come at the cost of limiting memory capacity.

In this chapter, we address the first research question of how we can increase memory capacity in existing (static) memory architectures while not leading to significant overfitting, thereby making them more generalisable to a wide range of discourse tasks. In

this regard, we propose a unified model generalising weight tying to increase memory capacity and in doing so, make the model more expressive. The proposed model achieves uniformly high performance, improving on the best results for memory network-based models on the bAbI dataset, and competitive results on Dialog bAbI. More importantly, on the second dialog state tracking challenge corpus (DSTC-2), a corpus set in the real world scenario of restaurant booking, we demonstrate the superiority of the proposed method to a collection of baselines, highlighting the benefits of unified weight tying to discourse understanding.

This chapter is based on and extends the following paper:

> Fei Liu, Trevor Cohn and Timothy Baldwin. 2017. Improving End-to-End Memory Networks with Unified Weight Tying. In *Proceedings of the Australasian Language Technology Association Workshop 2017 (ALTW 2017)*, pages 16–24, Brisbane, Australia.

3.1 Motivation for Unified Weight Tying

Deep neural network models have demonstrated strong performance on a number of challenging tasks, such as image classification (He et al., 2015), speech recognition (Graves et al., 2013), and various natural language processing tasks (Kim, 2014, Bahdanau et al., 2015, Xiong et al., 2016). Recently, the augmentation of neural networks with external memory components has been shown to be a powerful means of capturing context of different types (Graves et al., 2014, 2016, Rae et al., 2016). Of particular interest to this chapter is the work by Sukhbaatar et al. (2015), on end-to-end memory networks (MemN2Ns, as outlined in Section 2.3.2), which exhibit remarkable reasoning capabilities, e.g. for reasoning (Weston et al., 2016) and goal-oriented dialogue tasks (Bordes and Weston, 2017). Typically, such tasks consist of three key components: a sequence of supporting facts (the story), a question, and its answer. An example task is given in Table 5.1. Given the first two, a story and question, as input, it is the model's job to reason over the supporting facts and predict the answer to the question.

#	Story
1	Jeff went to the kitchen.
2	Mary travelled to the hallway.
3	Jeff picked up the milk.
4	Jeff travelled to the bedroom.
5	Jeff left the milk.
6	Jeff went to the bathroom.

Question	Answer
Where is the milk now?	bedroom
Where was Jeff before the bedroom?	kitchen

Table 3.1 An example triple of story, question and answer, extracted from the bAbI dataset (Weston et al., 2016). Key pieces of evidence for the first question are underlined, with distractors marked with dashed underline. This table is reproduced from Table 1.1

Brief Overview of MemN2N

Let us briefly review the architecture of MemN2N (described in Section 2.3.2; for those already familiar with MemN2N, feel free to skip this section). Building on top of memory networks (Weston et al., 2015), Sukhbaatar et al. (2015) introduced MemN2N, discarding the memory position supervision and making the model trainable in an end-to-end fashion, through the advent of supporting memories and a memory access controller. An illustration of a single-hop MemN2N is shown in Figure 3.1.

Suppose that a story consists of a sequence of n sentences: $\{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$, each sentence is in turn comprised of a sequence of words: $\mathbf{s}_i = \{s_{i1}, \ldots, s_{i|\mathbf{s}_i|}\}$ where $|\mathbf{s}_i|$ denotes the length of the sentence. Also, a question is formed by a sequence of words: $\mathbf{q} = \{q_1, \ldots, q_{|\mathbf{q}|}\}$ where $|\mathbf{q}|$ is the number of words in the question.

Representations of the context sentences $\{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ in the story are encoded using two sets of embedding matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ (both of size $d \times |V|$ where d is the embedding size and $|V|$ the vocabulary size), and stored in the input and output memory cells $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n$ and $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$, each of which is obtained via $\boldsymbol{m}_i = \sum_{j=1}^{|\mathbf{s}_i|} \boldsymbol{A}\Phi(s_{ij})$ and $\boldsymbol{c}_i = \sum_{j=1}^{|\mathbf{s}_i|} \boldsymbol{C}\Phi(s_{ij})$, where $\Phi(\cdot)$ is a function that maps an input word into a one-hot representation of dimension $|V|$. The input question $\mathbf{q}$ is encoded with another embedding matrix $\boldsymbol{B} \in \mathbb{R}^{d \times |V|}$ such

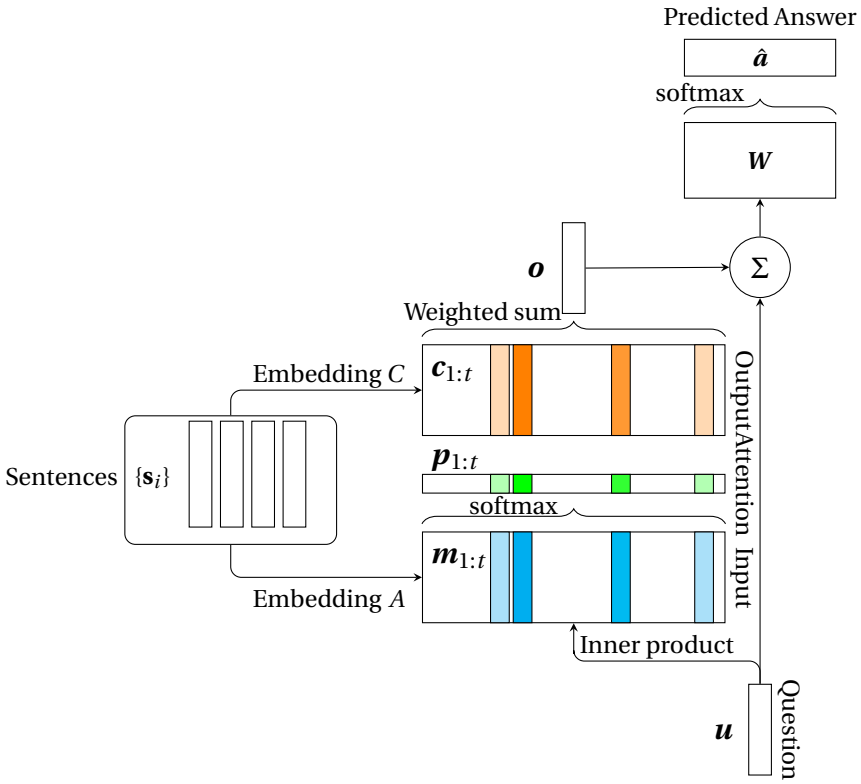

Figure 3.1 Illustration of a MemN2N with a single memory hop, reproduced from Figure 2.1.

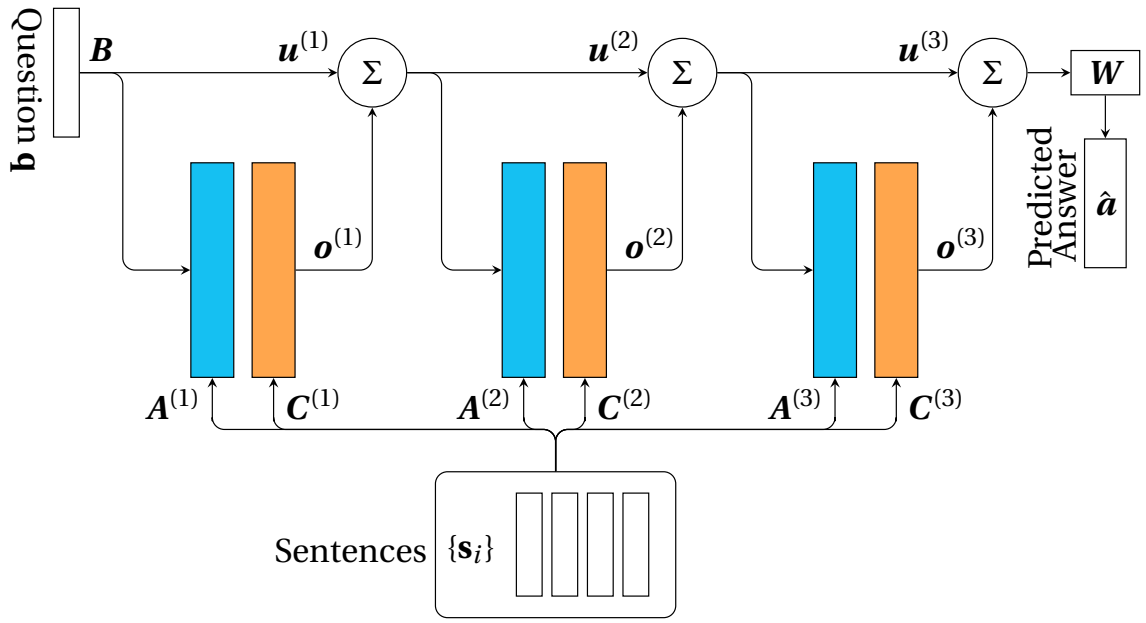

Figure 3.2 Illustration of a MEMN2N with multiple memory hops, reproduced from Figure 2.2.

that $\boldsymbol{u} = \sum_{j=1}^{|\mathbf{q}|} \boldsymbol{B}\Phi(q_j)$. MEMN2N uses the question embedding $\boldsymbol{u}$ and the input memory representations $\boldsymbol{m}_i$ to measure the relevance between the question and each supporting context sentence, resulting in a vector of attention weights:

$$p_i = \text{softmax}(\boldsymbol{u}^\top \boldsymbol{m}_i). \tag{3.1}$$

Once the attention weights have been computed, the memory access controller receives the response $\boldsymbol{o}$ in the form of a weighted sum over the output memory representations:

$$\boldsymbol{o} = \sum_{i=1}^{n} p_i \boldsymbol{c}_i. \tag{3.2}$$

To enhance the model's ability to cope with more challenging tasks requiring multiple supporting facts from the memory, Sukhbaatar et al. (2015) further extended the model by stacking multiple memory layers (also known as "hops"), in which case the output of the

k-th hop is taken as input to the $(k+1)$-th hop:

$$\boldsymbol{u}^{(k+1)} = \boldsymbol{o}^{(k)} + \boldsymbol{u}^{(k)}. \tag{3.3}$$

An illustrate of a multi-hop MEMN2N is shown in Figure 3.2.

Lastly, MEMN2N predicts the answer to question **q** using a softmax function:

$$\hat{\boldsymbol{a}} = \text{softmax}(\boldsymbol{W}(\boldsymbol{o}^{(K)} + \boldsymbol{u}^{(K)})) \tag{3.4}$$

where $\hat{\boldsymbol{a}}$ is the predicted answer distribution, $\boldsymbol{W} \in \mathbb{R}^{|V| \times d}$ is a parameter matrix for the model to learn (note that in the context of bAbI tasks, answers are single words), and K is the total number of hops.

Issues and Motivation

One drawback of MEMN2N is the necessity to choose between two types of weight tying, namely adjacent (ADJ) and layer-wise (LW) weight tying. Although, in most cases, MEMN2N works well with both of them, as reported in Sukhbaatar et al. (2015), the performance is uneven on some difficult tasks. On such tasks, we observe that while MEMN2N with ADJ may attain a perfect accuracy, LW falls far short behind. The reverse case can can also be observed. That is, ADJ and LW are almost complementary on such tasks.

This chapter focuses on improving static memory networks, namely End-to-End Memory Networks (MEMN2N). While the in-depth review of MEMN2N can be found in Section 2.3.2, here we dive into the weight tying mechanism (Section 2.3.2.3). In Sukhbaatar et al. (2015), two types of weight tying were explored for MEMN2N, namely adjacent ("ADJ") and layer-wise ("LW"). With LW, the input and output embedding matrices are shared across different hops (i.e., $\boldsymbol{A}^{(1)} = \boldsymbol{A}^{(2)} = \ldots = \boldsymbol{A}^{(K)}$ and $\boldsymbol{C}^{(1)} = \boldsymbol{C}^{(2)} = \ldots = \boldsymbol{C}^{(K)}$, all of size $\mathbb{R}^{d \times |V|}$), resembling RNNs. With ADJ, on the other hand, not only is the output embedding for a given layer shared with the corresponding input embedding

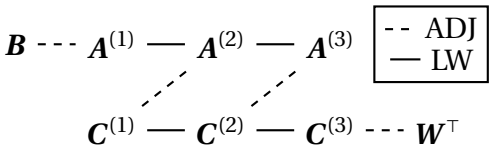

Figure 3.3 Illustration of the two types of weight tying mechanisms based on a MEMN2N model with 3 hops. Lines and dashed lines indicate parameter sharing relationships between embedding matrices.

(i.e., $\boldsymbol{A}^{(k+1)} = \boldsymbol{C}^{(k)}$), the answer prediction matrix $\boldsymbol{W}$ and question embedding matrix $\boldsymbol{B}$ are also constrained such that $\boldsymbol{W}^\top = \boldsymbol{C}^{(K)}$ and $\boldsymbol{B} = \boldsymbol{A}^{(1)}$.

While both ADJ and LW work well, achieving comparable overall performance in terms of mean error over the 20 bAbI tasks, their performance on a subset of the tasks (i.e., tasks 3, 16, 17, and 19, as shown in Table 3.2; for the full results on the 20 bAbI tasks, see the first 3 columns of Table 3.4) is inconsistent, with one performing very well, and the other performing poorly. While less pronounced on task 3, the performance gap between ADJ and LW is more obvious on the other 3 tasks (i.e., tasks 16, 17, and 19). The inconsistency in performance between these two weight tying schemes may likely be caused by the varying nature of such tasks, in particular, the size of the contextual stories needed to be stored in the memory. The number of sentences in a story in tasks 17 and 19 (positional reasoning and path finding) is 2 and 5, respectively. In comparison, task 16 contains stories consisting of as many as 9 sentences and requires multiple passes of a story to locate the answer. In terms of the two weight tying schemes, while ADJ can be preceived as more flexible, allowing the number of embedding weight matrices to grow as the number of memory hops increases, LW is more strict, resulting in the same set of input/output embedding matrices being shared across memory hops. While the added flexibility with ADJ may help MEMN2N in accommodating the contextual complexity presented in task 16, it likely causes overfitting when the context size is small, as in tasks 17 and 19 where LW gains superiority due to its limited size. Task 3, 3 supporting facts, requires a model to take the chronological order of events into account. Although only slightly outperforming ADJ, LW is considered to be more suitable for this task as it can be

Task	MemN2N	
	ADJ	LW
3: 3 supporting facts	90.7	**97.9**
16: basic induction	**99.6**	48.2
17: positional reasoning	59.3	**81.4**
19: path finding	33.5	**97.7**

Table 3.2 Accuracy (%) reported in Sukhbaatar et al. (2015) on a selected subset of the 20 bAbI 10k tasks (for the full results on the 20 bAbI tasks, see the first 3 columns of Table 3.4). Note that performance in the LW column is obtained with a larger embedding size $d = 100$ and ReLU non-linearity applied to the internal state after each hop.

cast as a traditional RNN, as pointed out by Sukhbaatar et al. (2015), to account for the sequential nature of the task.

Based on this observation and focusing on improving MemN2N, in this chapter, we introduce a unified weight tying mechanism exploiting the benefits of both ADJ and LW, and making the model capable of dynamically determining the best weight tying approach for a given task. This is realised through the use of a gating vector. Not only is this method free of the choice of weight tying, it also achieves the best performance for a memory network-based model on the bAbI dataset, superior to both adjacent and layer-wise weight tying, and competitive results on Dialog bAbI.

3.2 Related Memory-augmented Models

Gated End-to-End Memory Networks (GMemN2Ns) (Liu and Perez, 2017) are a variant of MemN2N with a simple gating mechanism on the connections between hops, allowing the model to dynamically regulate the information flow between the controller and the memory.[1] Dynamic Memory Networks (DMNs) (Xiong et al., 2016) and its improved version (DMN+) employ RNNs to sequentially process contextual information stored in the memory. The differential neural computer (DNC) (Graves et al., 2016) utilises an external memory matrix, which can be read and written to, resembling the random-access

[1] Although the work of the author of this thesis, the paper by Liu and Perez (2017) is based on the work carried out as part of an internship at Xerox Research Centre Europe and therefore excluded from this thesis.

Var.	Description	Dim.
d	Dimensionality of embeddings	$\mathbb{R}$
$\|V\|$	Vocabulary size	$\mathbb{R}$
$\boldsymbol{m}_i^{(k)}$	Input memory embedding of the i-th context sentence at the k-th hop	$\mathbb{R}^d$
$\boldsymbol{c}_i^{(k)}$	Output memory embedding of the i-th context sentence at the k-th hop	$\mathbb{R}^d$
$\boldsymbol{u}^{(k)}$	Question embedding at the k-th hop	$\mathbb{R}^d$
$\boldsymbol{o}^{(k)}$	Weighted output embedding at the k-th hop	$\mathbb{R}^d$
$\boldsymbol{z}$	Gating vector to dynamically determine the type of weight tying	$\mathbb{R}^d$
$\boldsymbol{h}_t$	Hidden GRU representation of processed memory at step t	$\mathbb{R}^d$
$\boldsymbol{b}_z$	Bias term for computing $\boldsymbol{z}$	$\mathbb{R}^d$
$\boldsymbol{A}^{(k)}$	Embedding matrix for input memory cell at the k-th hop	$\mathbb{R}^{d\times\|V\|}$
$\boldsymbol{C}^{(k)}$	Embedding matrix for output memory cell at the k-th hop	$\mathbb{R}^{d\times\|V\|}$
$\tilde{\boldsymbol{A}}^{(k)}$	Unconstrained embedding matrix for input memory cell at the k-th hop	$\mathbb{R}^{d\times\|V\|}$
$\tilde{\boldsymbol{C}}^{(k)}$	Unconstrained embedding matrix for output memory cell at the k-th hop	$\mathbb{R}^{d\times\|V\|}$
$\boldsymbol{B}$	Embedding for question at the 1^{st} hop	$\mathbb{R}^{d\times\|V\|}$
$\boldsymbol{W}$	Weight matrix for the classifier at the last (K-th) hop	$\mathbb{R}^{\|V\|\times d}$
$\boldsymbol{W}_z$	Weight matrix for computing $\boldsymbol{z}$	$\mathbb{R}^{d\times 2d}$
$\boldsymbol{H}$	Weight matrix for linear transformation between hops	$\mathbb{R}^{d\times d}$

Table 3.3 Notation summary.

memory mechanism in a conventional computer. All these models have been shown to have competent reasoning capabilities over the bAbI dataset (Weston et al., 2016). For a more comprehensive review of memory-augmented models, see Chapter 2.

3.3 Memory Networks with Unified Weight Tying

The key idea in this chapter is to let the model determine dynamically which type of weight tying mechanism it should rely on. Depending on the input story, unified weight tying can favour one over the other, allowing it to accommodate the added complexity presented in more challenging tasks as in ADJ or revert back to the simpler LW to ameliorate overfitting. For better readability, a summary table of notations used in this chapter is provided in Table 3.3.

Recall that there are two types: ADJ and LW. With ADJ, the input embedding $\boldsymbol{A}^{(k)}$ of the k-th hop is constrained to share the same parameters with the output embedding $\boldsymbol{C}^{(k-1)}$

of the $(k-1)$-th hop (i.e., $\boldsymbol{A}^{(k)} = \boldsymbol{C}^{(k-1)}$). In contrast, with LW, the same input/output embedding matrices are shared across different hops (i.e., $\boldsymbol{A}^{(1)} = \boldsymbol{A}^{(2)} = \ldots = \boldsymbol{A}^{(K)}$ and $\boldsymbol{C}^{(1)} = \boldsymbol{C}^{(2)} = \ldots = \boldsymbol{C}^{(K)}$). To this end, we design a dynamic, unified weight tying mechanism, named UMEMN2N, allowing the model to decide on the preferred type of weight tying based on the input. Specifically, the key element in UMEMN2N is that embedding matrices are constructed dynamically for each instance. This is in contrast to MEMN2N and GMEMN2N where the same embedding matrices are used for every input. UMEMN2N uses a gating vector $\boldsymbol{z}$ (described in Equation (3.8)) to construct the embedding matrices (i.e., $\boldsymbol{A}^{(k)}, \boldsymbol{C}^{(k)}, \boldsymbol{B}$ and $\boldsymbol{W}$) on the fly, based on the information carried by $\boldsymbol{z}$ regarding the input question $\boldsymbol{u}^{(1)}$ as well as the context sentences in the story $\boldsymbol{m}_i$:

$$\boldsymbol{A}^{(k+1)} = \boldsymbol{A}^{(k)} \odot (\boldsymbol{1} - \boldsymbol{z}) + \boldsymbol{C}^{(k)} \odot \boldsymbol{z} \tag{3.5}$$

$$\boldsymbol{C}^{(k+1)} = \boldsymbol{C}^{(k)} \odot (\boldsymbol{1} - \boldsymbol{z}) + \tilde{\boldsymbol{C}}^{(k+1)} \odot \boldsymbol{z} \tag{3.6}$$

where $\odot$ is the column element-wise multiplication operation, and $\tilde{\boldsymbol{C}}^{(k+1)}$ the unconstrained embedding matrix. We further define $\boldsymbol{A}^{(1)} = \tilde{\boldsymbol{A}}^{(1)}$ and $\boldsymbol{C}^{(1)} = \tilde{\boldsymbol{C}}^{(1)}$, where $\tilde{\boldsymbol{A}}^{(1)}$ and $\tilde{\boldsymbol{C}}^{(1)}$ are the unconstrained embedding matrices for hop 1. As shown in Equation (3.5), the input embedding matrix $\boldsymbol{A}^{(k+1)}$ for the $(k+1)$-th hop is composed of a weighted sum of the input and output embedding matrix $\boldsymbol{A}^{(k)}$ and $\boldsymbol{C}^{(k)}$ for the k-th hop, resembling LW and ADJ, respectively. The summation is weighted by the gating vector $\boldsymbol{z}$. $\boldsymbol{C}^{(k+1)}$ is constructed in a similar fashion. Ultimately, the larger the values of the elements of $\boldsymbol{z}$, the more UMEMN2N favours ADJ. Conversely, smaller $\boldsymbol{z}$ values indicate an inclination for LW. This results in not only simpler models with fewer design choices but also better interpretability, as shown in Section 3.4.4.

Another key to this approach is the gating vector $\boldsymbol{z}$, which is formulated to incorporate knowledge informative to the choice of the weight tying scheme. Here, we take inspiration from gated end-to-end memory networks ("GMEMN2N": Liu and Perez (2017)), where a gating mechanism is learned in an end-to-end fashion to regulate the information flow between memory hops. Concretely, similarly to DMN and DMN+, we encode the story by

first reading the memory one sentence at a time with a GRU:

$$\boldsymbol{h}_{t+1} = \text{GRU}(\boldsymbol{m}_t, \boldsymbol{h}_t) \tag{3.7}$$

where t is the current time step and $\boldsymbol{m}_t$ is the context sentence in the story at time t. Then, the last hidden state $\boldsymbol{h}_T$ of the GRU is taken to be the representation of the story.[2] Next, $\boldsymbol{z}$ is defined to be a vector of dimension d:

$$\boldsymbol{z} = \sigma\left(\boldsymbol{W}_z \begin{bmatrix} \boldsymbol{u}^{(1)} \\ \boldsymbol{h}_T \end{bmatrix} + \boldsymbol{b}_z\right) \tag{3.8}$$

where $\boldsymbol{W}_z$ is a weight matrix, $\boldsymbol{b}_z$ a bias term, σ the sigmoid function and $\begin{bmatrix} \boldsymbol{u}^{(1)} \\ \boldsymbol{h}_T \end{bmatrix}$ the concatenation of $\boldsymbol{u}^{(1)}$ and $\boldsymbol{h}_T$. Essentially, the gating vector $\boldsymbol{z}$ is now dependent on not only the question $\boldsymbol{u}^{(1)}$, but also the context sentences in the memory encoded in $\boldsymbol{h}_T$. Note that the gating vector $\boldsymbol{z}$ can be replaced by a gating scalar z, but we choose to use a vector for more fine-grained control as in LSTMs (Hochreiter and Schmidhuber, 1997) and GRUs (Cho et al., 2014b).

To simplify the model, we constrain $\boldsymbol{B}$ and $\boldsymbol{W}^\top$ to share the same parameters as $\boldsymbol{A}^{(1)}$ and $\boldsymbol{C}^{(K)}$ respectively. Moreover, following Sukhbaatar et al. (2015), we add a linear mapping $\boldsymbol{H} \in \mathbb{R}^{d \times d}$ to the update connection between memory hops, but in our case, down-weight it by $\mathbf{1} - \boldsymbol{z}$, resulting in:

$$\boldsymbol{u}^{(k+1)} = \boldsymbol{o}^{(k)} + (\boldsymbol{H} \odot (\mathbf{1} - \boldsymbol{z}))\boldsymbol{u}^{(k)} \tag{3.9}$$

where $\boldsymbol{H}$ is a trainable weight matrix, not the concatenation of $\left[\boldsymbol{h}_t\right]$ for $t \in [1, T]$.

Lastly, the model predicts the answer to question q using a softmax function:

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{W}(\boldsymbol{o}^{(K)} + \boldsymbol{u}^{(K)})) \tag{3.10}$$

[2] We also performed additional experiments with a bi-directional GRU over the memory as in Xiong et al. (2016), but did not observe any performance gain.

where K is the total number of memory hops and $\hat{\boldsymbol{y}}$ is the predicted probability distribution over all possible answer candidates.

Regularisation. In order to prevent the input and output embedding matrices $\boldsymbol{A}^{(k)}$ and $\boldsymbol{C}^{(k)}$ from being dominated by the unconstrained embedding matrices, it is necessary to restrain the magnitude of the values in $\tilde{\boldsymbol{A}}^{(1)}$ and $\tilde{\boldsymbol{C}}^{(k)}$; otherwise the model degenerates to MEMN2N without weight tying. Therefore, in addition to the categorical cross entropy loss over N training instances:

$$\mathscr{L} = \sum_{N} \text{CrossEntropy}(\boldsymbol{y}, \hat{\boldsymbol{y}}) \tag{3.11}$$

where $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ (Equation (3.10)) are the true and predicted answer, we enforce a regularisation penalty and formulate the new objective function as:

$$\mathscr{L}' = \mathscr{L} + \lambda \cdot \left(\| \tilde{\boldsymbol{A}}^{(1)} \|_2^2 + \sum_{k} \| \tilde{\boldsymbol{C}}^{(k)} \|_2^2 \right) \tag{3.12}$$

3.4 Experiments on Discourse Tasks

To test the effectiveness of the proposed method, in this section, we apply UMEMN2N to a range of discourse tasks, ranging from reasoning-focused question answering (Section 3.4.1), to goal-oriented dialog completion (Section 3.4.2), to dialog state tracking (Section 3.4.3).[3] The wide variety of tasks, while all focusing on discourse, demonstrates the broad applicability of UMEMN2N. Experimental results further show that unified weight tying substantially outperforms a collection of baselines, superior to both ADJ and LW, thereby enabling memory network-based models to better understand discourse.

[3]Note that the experiments on the reasoning-focused question answering bAbI dataset (Section 3.4.1) only serve as a sanity check to verify the validity of the proposed methods. Further validation on real-world datasets (Section 3.4.2 and Section 3.4.3) demonstrates the effectiveness of UMEMN2N.

3.4.1 Question Answering bAbI Experiments

3.4.1.1 Experimental Setup

Dataset: We evaluate the proposed model over the bAbI dataset (Weston et al., 2016) (v1.2). As bAbI has been described in Section 2.6.1.2, here we only briefly review the dataset and focus more on the task and the training procedure. This dataset features 20 different synthetically constructed reasoning tasks in the form of: (1) a list of supporting statements ($\mathbf{x}_1, \ldots, \mathbf{x}_n$); (2) a question ($\mathbf{q}$); and (3) the answer ($y$, typically a single word or short phrase). The answer is available at training time and used to train the model. At test time, it is the model's job to predict the correct answer to the question. Each task in bAbI is synthetically generated with a distinct emphasis on a specific type of reasoning. Note that not every supporting statement is necessarily relevant to the question. In fact, only a selected subset of them are truly informative to answering the question, the rest are simply irrelevant distractors. In order to predict the desired answer, the model is required to locate (or focus on) the relevant context sentences among irrelevant distractors in the memory. As with MEMN2N, our model can be trained in a fully end-to-end fashion and requires only the answers themselves as the supervision signal. The dataset comes in two sizes, with either 1k or 10k training instances per task. In this work, we focus exclusively on the 10k version.[4]

Training details: Following Sukhbaatar et al. (2015), we hold out 10% of the bAbI training set to form a development set. Position encoding and temporal encoding (with 10% random noise) are also incorporated into the model. Training is performed over 100 epochs with a batch size of 32 using the Adam optimiser (Kingma and Ba, 2015) with a learning rate of 0.005. Following Sukhbaatar et al. (2015), linear start is employed in all our experiments for the first 20 epochs where we remove the softmax function in each memory hop and re-insert it back after 20 epochs. All weight parameters are initialised based on a Gaussian distribution with zero mean and $\sigma = 0.1$. Gradients with an ℓ_2 norm

[4] We also conducted experiments on the 1k dataset but observed weak performance. We suspect that this is largely due to overfitting given the added complexity of UMEMN2N compared with MEMN2N.

of 40 are divided by a scalar to have norm 40. Also following Sukhbaatar et al. (2015), we use only the most recent 50 sentences as the memory and set the number of memory hops to 3, the embedding size to 20, and λ to 0.001.

Consistent with other published results over bAbI (Sukhbaatar et al., 2015, Graves et al., 2016, Seo et al., 2017), we repeat training 30 times for each task, and select the model which performs best on the development set.[5]

3.4.1.2 Results on bAbI

The results on the 20 bAbI QA tasks are presented in Table 3.4. We benchmark against other memory network based models: (1) MEMN2N with ADJ and LW (Sukhbaatar et al., 2015); (2) DMN (Kumar et al., 2016) and its improved version DMN+ (Xiong et al., 2016); and (3) GMEMN2N (Liu and Perez, 2017).

Major improvements on the difficult tasks. The most noticeable performance gains are over tasks 16, 17 and 19 where, compared with the vanilla MEMN2N, UMEMN2N achieves much better results than the worst of ADJ and LW, surpassing both ADJ and LW in the case of tasks 17 and 18. This confirms the validity of the model.

Minor improvements on tasks 3 and 18. On tasks where both ADJ and LW achieve strong results, UMEMN2N performs better again.

UMEMN2N maintains equally competitive performance on the other tasks. Our unified weight tying scheme does not degrade performance on the less challenging tasks.

Best performing memory network on bAbI 10k. UMEMN2N achieves the best combined results over bAbI 10k, superior to the previous top model DMN+ where two GRUs

[5]Note that it is common practice to report results on bAbI by selecting the best model (chosen by validation performance) out of multiple runs (Graves et al., 2016, Seo et al., 2017, Henaff et al., 2017). While in most cases model performance is robust (with standard deviation at around ± 1.0), a large variance in accuracy can be observed for a certain subset of tasks (e.g., tasks 3: 80.0 ± 16.0, 17: 79.8 ± 4.6 and 19: 66.5 ± 12.5). Also note that Neural Turing Machine (NTM) and Differentiable Neural Computer (DNC) result in similar degrees of variance in model performance (Graves et al., 2016).

Task	MemN2N ADJ	MemN2N LW	DMN	DMN+	GMemN2N	UMemN2N
1: 1 supporting fact	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
2: 2 supporting facts	99.7	99.7	98.2	99.7	**100.0**	99.2
3: 3 supporting facts	90.7	97.9	95.2	**98.9**	95.5	95.5
4: 2 argument relations	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
5: 3 argument relations	99.4	99.2	99.3	99.5	**99.8**	99.3
6: yes/no questions	**100.0**	99.9	**100.0**	**100.0**	**100.0**	**100.0**
7: counting	96.3	98.0	96.9	97.6	98.2	**98.9**
8: lists/sets	99.2	99.1	96.5	**100.0**	99.7	99.5
9: simple negation	99.2	99.7	**100.0**	**100.0**	**100.0**	**100.0**
10: indefinite knowledge	97.6	**100.0**	97.5	**100.0**	99.8	**100.0**
11: basic coreference	**100.0**	99.9	99.9	**100.0**	**100.0**	**100.0**
12: conjunction	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
13: compound coreference	**100.0**	**100.0**	99.8	**100.0**	**100.0**	**100.0**
14: time reasoning	**100.0**	99.9	**100.0**	**100.0**	**100.0**	**100.0**
15: basic deduction	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
16: basic induction	99.6	48.2	99.4	54.7	**100.0**	99.9
17: positional reasoning	59.3	81.4	59.6	**95.8**	72.2	90.7
18: size reasoning	93.3	94.7	95.3	97.9	91.5	**99.4**
19: path finding	33.5	97.7	34.5	**100.0**	69.0	84.0
20: agent's motivation	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Average	93.4	95.8	93.6	97.2	96.3	**98.3**

Table 3.4 Accuracy (%) on the 20 bAbI 10k tasks for our proposed method (UMemN2N) and various benchmark methods. **Bold** indicates the best result for a given task.

are employed to process the memory at the sentence and hop level; there is a particularly big improvement on task 16 (UMEMN2N = 99.9 vs. DMN+ = 54.7). Note that UMEMN2N achieves this with a much smaller embedding size $d = 20$ compared to 80 in DMN+.

Comparison with the state-of-the-art. Seo et al. (2017) report a higher overall score (99.3) for a deep learning model, named Query Reduction Networks. Their model is based on RNNs and updates its internal hidden states with a control gate, whose value is determined by the relevance of the current input sentence and the question, rendering it unrelated to memory networks and thus not immediately comparable with UMEMN2N. Also note that their model is based on embeddings of size $d = 200$, much larger than UMEMN2N's 20, and that when their model is restricted to an embedding size of $d = 50$, the reported result is 96.8, lower than ours with $d = 20$.

3.4.2 Dialog bAbI Experiments

In addition to the experiments on the natural language-based QA bAbI dataset presented previously in Section 3.4.1, we conduct further experiments on a goal-oriented, dialog-based dataset: Dialog bAbI (Bordes and Weston, 2017).

3.4.2.1 Experimental Setup

Dataset: We employ a collection of goal-oriented dialog tasks, Dialog bAbI, which has been covered in Section 2.6.2.2. All examples in Dialog bAbI are set in a restaurant reservation scenario, developed by Bordes and Weston (2017), consisting of 6 categories each with a specific focus on tasking on aspect of an end-to-end dialog system: 1. issuing API calls, 2. updating API calls, 3. displaying options, 4. providing extra-information, 5. conducting full dialogs (the aggregation of the first 4 tasks), 6. Dialog State Tracking Challenge 2 corpus (DSTC-2). Tasks 1-5 are generated synthetically in the form of conversation between a user and a bot with entities drawn from a knowledge base with facts defining restaurants and their associated properties (e.g., location and price range, 7 properties in total). Starting with a request from the user, a dialog proceeds with subsequent and

alternating user-bot utterances. The bot (or system) needs to figure out the user intention and react accordingly. A separate collection of test sets, with entities not occurring in the training set, has also been developed to evaluate the ability of the bot to deal with out-of-vocabulary (OOV) items. Task 6 is based on and derived from the second Dialog State Tracking Challenge (Henderson et al., 2014) with real human-bot conversations.

Training details: Following the works of Bordes and Weston (2017) and Liu and Perez (2017), we frame the task in the same fashion: at the i-th time step, the preceding sequence of utterances, $\mathbf{c}_1^u, \mathbf{c}_1^r, \mathbf{c}_2^u, \mathbf{c}_2^r, \ldots, \mathbf{c}_{i-1}^u, \mathbf{c}_{i-1}^r$,[6] is stored in the memory as $\boldsymbol{m}_i$ and $\boldsymbol{c}_i$. Taking the memory as contextual evidence, the goal of the model is to offer an answer $\mathbf{c}_i^r$ (the bot utterance at time i) to the question $\mathbf{c}_i^u$ (the user utterance at time i).

Note that the answers in this dataset may no longer be a single word but can be comprised of multiple words (in fact, almost all answers are multi-word ones). Following Bordes and Weston (2017), we replace the final prediction step in Equation (3.10) with:

$$\hat{\boldsymbol{y}} = \text{softmax}\Big((\boldsymbol{u}^{(K+1)})^\top \boldsymbol{W}' \Phi(\mathbf{y}_1), \ldots, (\boldsymbol{u}^{(K+1)})^\top \boldsymbol{W}' \Phi(\mathbf{y}_{|C|})\Big) \tag{3.13}$$

where $\boldsymbol{W}^{'} \in \mathbb{R}^{d \times |V|}$ is the weight parameter matrix for the model to learn, $\boldsymbol{u}^{(K+1)} = \boldsymbol{o}^{(K)} + \boldsymbol{u}^{(K)}$ (K is the total number of hops), $\mathbf{y}_i$ is the i-th response in the candidate set C such that $\mathbf{y}_i \in C$, $|C|$ the size of the candidate set, and $\Phi(\cdot)$ a function which maps the input text into a bag of dimension $|V|$.[7]

Additionally, we also append several key features to Φ, following Bordes and Weston (2017) and Liu and Perez (2017). First, two features marking the identity of the speaker of a particular utterance (user or model) are added to each of the memory slots. Second, we expand the feature representation function Φ of candidate responses with 7 additional features, each focusing on one of the 7 properties associated with any restaurants, indicating whether there are any exact matches between words occurring in the candidate and those in the question or memory. We refer to these 7 features as the *match* features. As

[6] Alternating between the user request, denoted $\mathbf{c}_i^u$ and the system response, denoted $\mathbf{c}_i^r$

[7] This is not a generative but rather a classification task.

shown by Bordes and Weston (2017) and in Table 3.5, the *match* features help improve model performance significantly.

In terms of the training procedure, experiments are carried out with the same configuration as described in Section 3.4.1.1. As a large variance can be observed due to how sensitive memory-based models are to parameter initialisation, following Sukhbaatar et al. (2015) and Liu and Perez (2017), we repeat each training 10 times using the same hyper-parameters and choose the best system based on validation performance.

3.4.2.2 Results on Dialog bAbI

The results on the Dialog bAbI tasks are shown in Table 3.5. In terms of baselines, we benchmark against other memory network-based models:[8] (1) MEMN2N (Sukhbaatar et al., 2015); and (2) GMEMN2N (Liu and Perez, 2017). While the results of GMEMN2N are achieved with ADJ, the type of weight tying for MEMN2N is not reported in Bordes and Weston (2017).

Improvements on task 5. It can be observed that UMEMN2N offers consistent performance boost on task 5 across all experimental settings, especially in the non-OOV group. Given that task 5 is the aggregation of the first 4 tasks, the performance increase suggests that the hybrid weight-tying mechanism in UMEMN2N is better capable of coping with tasks of various nature.

Equally competitive performance on tasks 1-4. UMEMN2N achieves comparable, if not slightly better in some cases, performance on tasks 1-4.

Performance on task 6. Compared to MEMN2N, UMEMN2N improves the performance consistently with or without the match features. In contrast to GMEMN2N, however, this is

[8] Seo et al. (2017) report a higher accuracy on Dialog bAbI with an average accuracy of 94.5/88.9/51.1 without *match* and 98.5/97.7/50.7 with *match* in the non-OOV/OOV/DSTC-2 settings. However, their model, based on RNNs, is rather different from memory networks and therefore deemed not immediately comparable and unrelated to the goal of this thesis: improving memory-augmented models for better discourse understanding.

Task	–match			+match		
	MemN2N	GMemN2N	UMemN2N	MemN2N	GMemN2N	UMemN2N
1. Issuing API calls	99.9	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
2. Updating API calls	**100.0**	**100.0**	**100.0**	98.3	**100.0**	**100.0**
3. Displaying options	**74.9**	**74.9**	**74.9**	**74.9**	**74.9**	**74.9**
4. Providing information	**59.5**	57.2	57.2	**100.0**	**100.0**	**100.0**
5. Full dialogs	96.1	96.3	**99.2**	93.4	98.0	**99.4**
Average	86.1	85.7	**86.3**	93.3	94.6	**94.9**
1. (OOV) Issuing API calls	72.3	82.4	**83.0**	96.5	**100.0**	**100.0**
2. (OOV) Updating API calls	**78.9**	**78.9**	**78.9**	**94.5**	94.2	94.4
3. (OOV) Displaying options	74.4	**75.3**	75.2	75.2	75.1	**75.3**
4. (OOV) Providing information	**57.6**	57.0	57.0	**100.0**	**100.0**	**100.0**
5. (OOV) Full dialogs	65.5	66.7	**67.8**	77.7	79.4	**79.5**
Average	69.7	72.1	**72.4**	88.8	89.7	**89.8**
6. Dialog state tracking 2	41.1	**47.4**	42.4	41.0	**48.7**	42.9

Table 3.5 Per-response accuracy on the Dialog bAbI tasks. MemN2N: Bordes and Weston (2017). GMemN2N: Liu and Perez (2017). +match suggests the use of the *match* features in Section 3.4.2.1. **Bold** indicates the best result in each group (with or without the *match* features) for a given task.

not the case. This is likely due to the input noise in the data collection process, compared to the synthetically generated nature of the other 5 tasks, and also the increased number of answer candidates (close to 2,500, as opposed to a few hundred in the other tasks with the most being roughly 1,500), rendering this task significantly more difficult than the others.

3.4.3 Dialog State Tracking Experiments

3.4.3.1 Experimental Setup

Dataset: To further evaluate the benefits of unified weight tying, we use the DSTC-2 dataset. We refer the reader to Section 2.6.2.2 for a detailed description of the dataset. The DSTC-2 dataset is split into three partitions: train, validation and test, with 1,612/ 506/1,117 instances respectively.

Note that while it is common practice to evaluate using the N-best Automatic Speech Recognition (ASR) hypotheses and their associated ASR scores, the goal of this experiment is to demonstrate the effectiveness of the proposed model over the two types of weight tying mechanisms of the vanilla MEMN2N, and we therefore simply take the dialog transcripts as input. Also note, we use the cleaned version of the dataset provided by Mrkšić et al. (2017) who manually rectified numerous spelling errors in the original release of DSTC-2.

Task definition: A full conversation between a user and a system with a total of T turns, each consisting of a pair of user and system utterances, can be broken down at each turn at time i. Given a sequence of utterances $\mathbf{c}_1^r, \mathbf{c}_1^u, \mathbf{c}_2^r, \mathbf{c}_2^u, \ldots, \mathbf{c}_i^r, \mathbf{c}_i^u$ (alternating between the user request, with denoted $\mathbf{c}_i^u$ and the system response, denoted $\mathbf{c}_i^r$), a model is responsible for predicting the values to three slots, namely area, food and price, denoted $\boldsymbol{y}_{\text{area}}$, $\boldsymbol{y}_{\text{food}}$ and $\boldsymbol{y}_{\text{price}}$ accordingly. Note that, in DSTC-2, a sequence of utterances always starts with a system response and ends with a user request, as opposed to Dialog bAbI where both the start and end of a sequence are user requests and a model is tasked to predict the next system response.

Training details: The values for the three slots are modelled jointly with a single UMEMN2N, with the identity of the slot in question provided as the question q. In this setting, a question is in the form of a vector $\boldsymbol{u}^{(1)}$, taking the value of the sum of the embeddings of the slot name, e.g., $\boldsymbol{u}^{(1)}_{\text{price range}} = \boldsymbol{B}\Phi(\text{price}) + \boldsymbol{B}\Phi(\text{range})$ where $\boldsymbol{B}$ is the embedding weight matrix for questions (sharing parameters with $\boldsymbol{A}^{(1)}$) and $\Phi(\cdot)$ is a function mapping words to their one-hot encoding.

To accommodate the added complexity of three types of slots (as opposed to one in bAbI and Dialog bAbI), we increase the number of weight matrices $\boldsymbol{W}$ to three in Equation (3.10), one for each type:

$$\hat{\boldsymbol{y}}_{\text{area}} = \text{softmax}(\boldsymbol{W}_{\text{area}}(\boldsymbol{o}^{(K)} + \boldsymbol{u}^{(K)})) \tag{3.14}$$

$$\hat{\boldsymbol{y}}_{\text{food}} = \text{softmax}(\boldsymbol{W}_{\text{food}}(\boldsymbol{o}^{(K)} + \boldsymbol{u}^{(K)})) \tag{3.15}$$

$$\hat{\boldsymbol{y}}_{\text{price}} = \text{softmax}(\boldsymbol{W}_{\text{price}}(\boldsymbol{o}^{(K)} + \boldsymbol{u}^{(K)})) \tag{3.16}$$

where $\boldsymbol{W}_{\text{area}}$, $\boldsymbol{W}_{\text{food}}$ and $\boldsymbol{W}_{\text{price}}$ are trainable weight matrices, and K is the total number of layers. Consistent with MEMN2N, as in Equation (2.59), we take as input the sum of the weighted memory representation $\boldsymbol{o}^{(K)}$ and the input question $\boldsymbol{u}^{(K)}$ at the top memory hop and feed it to three classifiers.

Training is performed jointly over the three slots and N training instances with cross entropy loss:

$$\mathscr{L} = \sum_{N} \text{CrossEntropy}(\boldsymbol{y}_{\text{area}}, \hat{\boldsymbol{y}}_{\text{area}}) + \text{CrossEntropy}(\boldsymbol{y}_{\text{food}}, \hat{\boldsymbol{y}}_{\text{food}}) + \text{CrossEntropy}(\boldsymbol{y}_{\text{price}}, \hat{\boldsymbol{y}}_{\text{price}}). \tag{3.17}$$

To further reduce training complexity and ameliorate overfitting, we simplify Equation (3.7) and replace the GRU with the max pooling operation over the temporal axis:

$$\boldsymbol{h}_T = \max\Big([\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_T]\Big) \tag{3.18}$$

where the max function is applied element-wise (dimension-wise).[9] The resulting memory representation $\boldsymbol{h}_T$ is then taken as input to Equation (3.8).

In terms of hyper-parameters, we largely re-use the same configuration as in Section 3.4.1.1 with the following exceptions. While training is still carried out with the Adam optimiser (Kingma and Ba, 2015), the learning rate is now set to 0.001. Embedding size is now 300 as opposed to 20 in previous sections. We also apply dropout to the input and output memories, $\boldsymbol{m}_i^{(k)}$ and $\boldsymbol{c}_i^{(k)}$, with a rate of 50%. Max gradient norm is set to 5.0.

The same set of hyper-parameters are also applied to the two baselines: MEMN2N + ADJ/LW with the exception of the Adam optimiser and learning rate. Here, in line with the works of Sukhbaatar et al. (2015) and Liu and Perez (2017), we instead use stochastic gradient descent with a learning rate of 0.01.[10]

For both UMEMN2N and the baselines, the best system is chosen based on its performance on the validation set of DSTC-2 with early stopping.

3.4.3.2 Results on DSTC-2

The results on the DSTC-2 are shown in Table 3.6. Following Henderson et al. (2013),[11] we report accuracy for area, food and price, three slots annotated in DSTC-2, as well as the joint accuracy where a predicted output is only considered correct if predictions for all slots are correct. In addition to the proposed method UMEMN2N, we also implement two baselines: MEMN2N + ADJ and MEMN2N + LW (Sukhbaatar et al., 2015), both trained with the same setting as UMEMN2N as described above, and measure the performance of all models by averaging over 5 runs.

Superior performance of UMEMN2N to MEMN2N with either ADJ or LW weight tying. UMEMN2N achieves consistently better results over the selected baselines, substantially

[9] We empirically observed slight performance improvements with max pooling over GRUs.

[10] We also experimented with Adam with a learning rate of 0.001, the same configuration as UMEMN2N, but observed performance degradation. This is perhaps due to the rather simple model architecture of MEMN2N with either ADJ or LW (compared to UMEMN2N), which does not require an advanced optimiser such as Adam.

[11] Under schedule 2 and label scheme A: only turns where the value to a slot has been filled are included and slot states are accumulated forwards through the dialog.

Model	Area	Food	Price	Joint
MEMN2N + ADJ	93.3	87.3	89.1	73.5
MEMN2N + LW	93.7	91.4	**90.6**	79.8
UMEMN2N	**95.8**	**93.3**	89.9	**82.1**

Table 3.6 Average accuracy (%) of various models on the DSTC-2 dataset over 5 runs. While both MEMN2N + ADJ/LW are implemented and run by us, the performance of MEMN2N + ADJ matches that reported in Perez and Liu (2017).

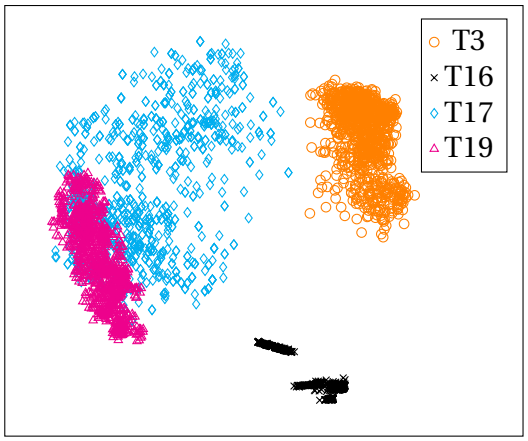

Figure 3.4 PCA scatter plot of the gating vectors z over bAbI tasks 3, 16, 17 and 19.

improving upon ADJ and LW with the exception of the price slot. While MEMN2N + LW manages to produce the best accuracy in price over the other two models in Table 3.6, UMEMN2N is only slightly inferior to LW. In terms of the joint category, the strong performance of UMEMN2N demonstrates that the proposed unified weight tying mechanism is capable of combining the strengths of both ADJ and LW with the unified weight tying mechanism, thereby generating more consistently correct output.

3.4.4 Analysis and Model Visualisation

To gain a better understanding of what the model has learned, we visualise the gating vectors z trained jointly on the difficult bAbI tasks (i.e., 3, 16, 17 and 19), in the form of a 2-D PCA scatter plot in Figure 3.4. Note that the motivation of the proposed unified weight tying mechanism is mainly based on the observation of the performance gap between ADJ and LW on the above 4 selected tasks. Four distinct clusters, representing the 4

Task	MemN2N		UMemN2N		$\bar{z}$
	ADJ	LW	single	joint	
3: 3 supporting facts	90.7	**97.9**	95.5	86.3	0.63
16: basic induction	99.6	48.2	**99.9**	99.8	0.70
17: positional reasoning	59.3	81.4	**90.7**	79.3	0.51
19: path finding	33.5	**97.7**	84.0	87.1	0.42

Table 3.7 Accuracy (%) reported in Sukhbaatar et al. (2015) on bAbI tasks 3, 16, 17, and 19. $\bar{z}$ denotes the mean of $\boldsymbol{z}$.

different tasks, are easily identifiable. Moreover, it can be observed that tasks 17 and 19, both favouring LW, are rather close whereas the black crosses representing task 16 form their own cluster, away from the other clusters, reflecting the preference for ADJ on this task. The task 3 cluster, while close to tasks 17 and 19, does not intersect them, this is likely due to the rather minor performance gap between ADJ and LW. On this task, the vanilla MemN2N demonstrates equally competitive performance with a slight inclination for LW, resulting in the orange cluster being placed close to those representing tasks 17 and 19 but with no overlap. Figure 3.4 is further evidence that our model dynamically learns the best weight tying method for a given task.

The performance of UMemN2N trained jointly on the above four tasks is shown in Table 3.7. Observe that while training on each task individually is superior, the joint model generates comparable, if not better, results on tasks where one of the weight tying schemes is a clear winner (e.g., ADJ for task 16 and LW for task 19).

To further investigate what the gating vector can learn, we show the mean value of $\boldsymbol{z}$ from a single model in Table 3.7. This follows our intuition that the greater the values of the elements of $\boldsymbol{z}$, the more UMemN2N favours ADJ. Conversely, smaller $\boldsymbol{z}$ values indicate that UMemN2N prefers LW.

3.5 Summary

In this chapter, we have addressed the first research question of increasing memory capacity while avoiding overfitting and making the model more generalisable to a wide range of

discourse tasks. Specifically, we have presented UMEMN2N, a model based on MEMN2N with a unified weight tying scheme and demonstrated the effectiveness of the proposed method on a set of discourse tasks, ranging from reasoning-focused question answering (bAbI), to goal-oriented dialog completion (Dialog bAbI), to dialog state tracking (DSTC-2). This is inspired and motivated by the observation of the performance gap between MEMN2N with ADJ and LW on various bAbI tasks. Not only does the unified weight tying mechanism improve model performance on a number of discourse level tasks, it also eliminates the necessity to choose between the two types of weight tying schemes associated with MEMN2N, further simplifying the hyper-parameter search space. With UMEMN2N, we are one step closer to the goal of this thesis, that is, creating tools for better discourse understanding.

While mainly focusing on improving static memory networks in this chapter, in the next chapter, we introduce a new model, incorporating static memory into an existing, state of the art statistical machine learning classifier, thereby significantly enhancing its modelling capacity and allowing it to capture long-range contextual dependencies.

Chapter 4

Enhancing Modelling Capacity with Static Memory Networks

While much effort has been focused on word- or sentence-level language phenomena, the multi-sentence nature of discourse often requires consideration of contexts of size much larger than a single sentence, looking beyond sentence boundaries. Due to the increased context size, capturing dependencies between distant items has become an essential yet challenging task, which is crucial for discourse understanding.

While the focus of Chapter 3 was on improving memory capacity, in this chapter, we address the second research question of how we can enhance modelling capacity by memory augmentation to better capture long-range contextual dependencies. We focus on a widely popular model, conditional random fields (CRFs, Lafferty et al. (2001), in particular the linear-chain variant), which, despite successful applications across a broad range of NLP tasks, are only able to model local features. While this has important benefits in terms of inference tractability, it limits the ability of the model to capture long-range dependencies between items. Attempts to extend CRFs to capture long-range dependencies have largely come at the cost of computational complexity and approximate inference. In this chapter, taking inspiration from static memory networks, we propose an extension to CRFs by integrating external memory, enhancing their modelling capacity and thereby allowing CRFs to incorporate information far beyond neighbouring steps. We

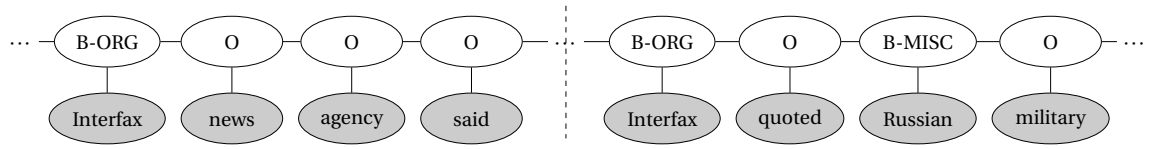

Figure 4.1 A NER example with long-range contextual dependencies, reproduced from Figure 1.1. The vertical dash line indicates a sentence boundary.

name the proposed model "memory-enhanced conditional random fields" (ME-CRFs). Experiments across three discourse tasks show substantial improvements over strong CRF and LSTM baselines, demonstrating the benefits of leveraging static memory to discourse understanding.

This chapter is based on and extends the following paper:

> Fei Liu, Timothy Baldwin and Trevor Cohn. 2017. Capturing Long-range Contextual Dependencies with Memory-enhanced Conditional Random Fields. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (IJCNLP 2017)*, pages 555–565, Taipei, Taiwan.

4.1 Motivation for Memory-enhanced Conditional Random Fields

While long-range contextual dependencies are prevalent in natural language, especially in discourse, for tractability reasons, most statistical models only use local features (Finkel et al., 2005). Take the sentences in Figure 4.1 for example. Here, while it is easy to determine that *Interfax* in the second sentence is a named entity, it is hard to determine its semantic class, as there is little context information. The usage in the first sentence, on the other hand, can be reliably disambiguated due to the post-modifying phase *news agency*. Ideally we would like to be able to share such contexts across all usages (and variants) of a given named entity for reliable and consistent identification and disambiguation.

A related example is forum thread discourse analysis, as discribed in Section 2.6.2.3 and displayed in Figure 4.2. Previous work has largely focused on linear-chain Conditional

Random Fields (CRFs) (Wang et al., 2011, Zhang et al., 2017a), framing the task as one of sequence tagging. Although CRFs are adept at capturing local structure, the problem does not naturally suit a linear sequential structure, i.e., a post may be a reply to either a neighbouring post or one posted far earlier within the same thread. In both cases, contextual dependencies can be long-range, necessitating the ability to capture dependencies between arbitrarily distant items. Identifying this key limitation, Sutton and McCallum (2004) and Finkel et al. (2005) proposed the use of CRFs with skip connections to incorporate long-range dependencies. In both cases the graph structure must be supplied a priori, rather than learned, and both techniques incur the need for costly approximate inference.

Recurrent neural networks (RNNs) have been proposed as an alternative technique for encoding sequential inputs, however plain RNNs are unable to capture long-range dependencies (Bengio et al., 1994, Hochreiter et al., 2001) and variants such as LSTMs (Hochreiter and Schmidhuber, 1997), although more capable of capturing non-local patterns, still exhibit a significant locality bias in practice (Lai et al., 2015, Linzen et al., 2016). That is, LSTMs are biased towards closer elements and do not readily capture long-range dependencies.

In this chapter, taking inspiration from the work of Weston et al. (2015) and Sukhbaatar et al. (2015) on memory networks (MemN2Ns), we propose to extend CRFs by integrating external memory mechanisms, thereby enabling the model to look beyond localised features and have access to the entire sequence. This is achieved with attention over the entire memory, more precisely, over every memory entry, which may be the representation of either a word (in the case of NER) or a sentence (in the case of forum thread discourse analysis). In addition to the enhanced capabilities of CRFs to capture long-range dependencies, from the perspective of MemN2Ns, the integration of linear-chain CRFs allows MemN2Ns to model the dynamics and transitions between neighbouring steps, thereby further improving their discourse understanding performance. We evaluate on an array of discourse tasks, ranging from forum thread parsing, to discourse-aware named entity recognition, to dialog state tracking. While the first two tasks are designed to demonstrate ME-CRF's ability to capture long-range contextual dependencies, the last task, framed as

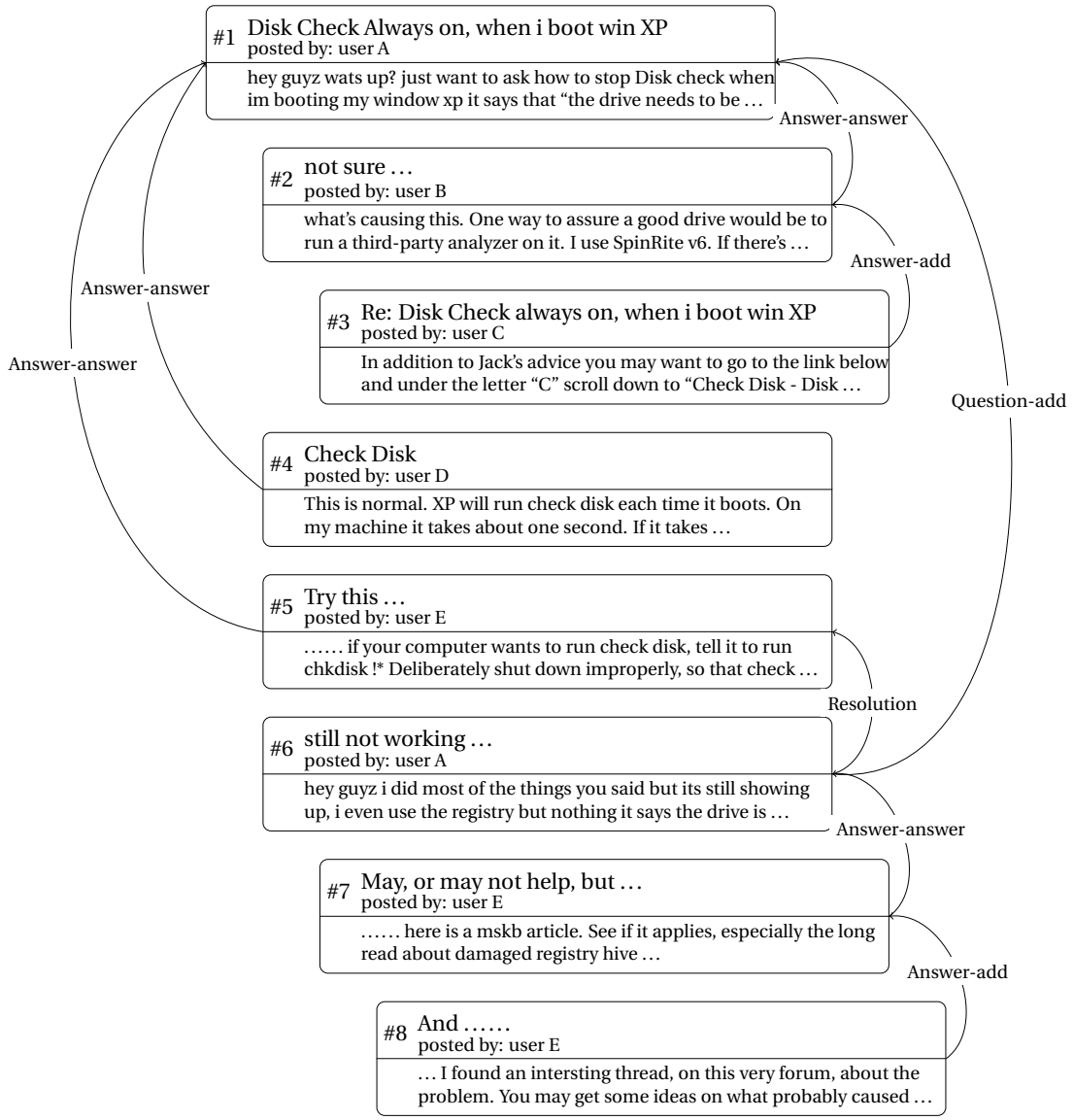

Figure 4.2 An example of the web forum thread discourse structure prediction task, reproduced from Figure 2.6. The "in reply to" relationships between posts are implied by the links with arrows starting from the reply and pointing to the post it answers. Captions on edges indicate the type of reply.

one of sequence tagging, tests the model's effectiveness to learn the evolution of dialog states. Experimental results show superior performance, outperforming a number of strong baselines, and validating the utility of memory integration.

4.2 Related Work

In this section, we review the different families of models that are relevant to this chapter, in capturing long-range contextual dependencies in different ways.

4.2.1 Conditional Random Fields (CRFs)

CRFs (Lafferty et al., 2001), in particular linear-chain CRFs, have been widely adopted and applied to sequence labelling tasks in NLP, but have the critical limitation that they only capture local structure (Sutton and McCallum, 2004, Finkel et al., 2005), despite non-local structure being common in structured language classification tasks. In the context of named entity recognition (NER), Sutton and McCallum (2004) proposed skip-chain CRFs as a means of alleviating this shortcoming, wherein distant items are connected in a sequence based on a heuristic such as string identity (to achieve label consistency across all instances of the same string). The idea of label consistency and exploiting non-local features has also been explored in the work of Finkel et al. (2005), who take long-range structure into account while maintaining tractable inference with Gibbs sampling (Geman and Geman, 1984), by performing approximate inference over factored probabilistic models. While both of these lines of work report impressive results on information extraction tasks, they come at the price of high computational cost and incompatibility with exact inference.

Similar ideas have also been explored by Krishnan and Manning (2006) for NER, where they apply two CRFs, the first of which makes predictions based on local information, and the second combines named entities identified by the first CRF in a single cluster, thereby enforcing label consistency and enabling the use of a richer set of features to capture

non-local dependencies. Liao and Grishman (2010) make a strong case for going beyond sentence boundaries and leveraging document-level information for event extraction.

While we take inspiration from these earlier studies, we do not enforce label consistency as a hard constraint, and additionally do not sacrifice inference tractability: our model is capable of incorporating non-local features, and is compatible with exact inference methods.

4.2.2 Recurrent Neural Networks (RNNs)

Recently, the broad adoption of deep learning methods in NLP has given rise to the prevalent use of RNNs (see Section 2.2.1). In particular, long short-term memories (LSTMs: Hochreiter and Schmidhuber (1997)), a variant of RNN, have become widely popular, and been successfully applied to a large number of tasks (Graves et al., 2013, Huang et al., 2015, Yang et al., 2016, Cho et al., 2014a, Bahdanau et al., 2015), as previously described in Section 2.2.1. However, as pointed out by Lai et al. (2015) and Linzen et al. (2016), RNNs — including LSTMs — are biased towards immediately preceding (or neighbouring, in the case of bi-directional RNNs) items, and perform poorly in contexts which involve long-range contextual dependencies, despite the inclusion of memory cells. This is further evidenced by the work of Cho et al. (2014a), who show that the performance of a basic encoder–decoder deteriorates as the length of the input sentence increases.

4.2.3 Memory Networks (MemN2Ns)

More recently, Weston et al. (2015) proposed memory networks and showed that the augmentation of memory is crucial to performing inference requiring long-range dependencies, especially when discourse-level reasoning between multiple supporting facts is required (see Section 2.3.2). Of particular interest to this chapter are so-called "memory hops" in memory networks, which are guided by an attention mechanism based on the relevance between a question and each supporting context sentence stored in the memory hop. Governed by the attention mechanism, the ability to access the entire sequence is

similar to the soft alignment idea proposed by Bahdanau et al. (2015) for neural machine translation. In this chapter, we borrow the concept of memory hops and integrate it into CRFs, thereby enabling the model to look beyond localised features and have access to the whole sequence via an attention mechanism.

4.3 Memory-enhanced Conditional Random Fields

In the context of sequential tagging, we assume the input is in the form of sequence pairs: $\mathscr{D} = \{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^{N}$ where $\mathbf{x}^{(n)}$ is the input of the n-th example in dataset $\mathscr{D}$ and consists of a sequence: $\{x_1^{(n)}, x_2^{(n)}, \ldots, x_T^{(n)}\}$. Similarly, $\mathbf{y}^{(n)}$ is of the same length as $\mathbf{x}^{(n)}$ ($|\mathbf{y}^{(n)}| = |\mathbf{x}^{(n)}|$) and consists of the corresponding labels $\{y_1^{(n)}, y_2^{(n)}, \ldots, y_T^{(n)}\}$. For notational convenience, hereinafter we omit the superscript denoting the n-th example.

In the case of NER (see Section 4.4.1), each x_t is a word in a sentence with y_t being the corresponding NER label. For forum thread discourse analysis (see Section 4.4.2), x_t represents the text of an entire post whereas y_t is the dialogue act label for the t-th post. Similarly, for the task of dialog state tracking (DSTC-2, see Section 4.4.3), x_t represents the text of a dialog turn, consisting of a pair of user request and system response.

The proposed model extends CRFs by integrating external memory and is therefore named a **Memory-Enhanced Conditional Random Field** ("ME-CRF"). We take inspiration from Memory Networks ("MemN2Ns": Weston et al. (2015)) and incorporate so-called memory hops into CRFs, thereby allowing the model to have unrestricted access to the whole sequence rather than localised features as captured by RNNs (Lai et al., 2015, Linzen et al., 2016).

As illustrated in Figure 4.3, ME-CRF can be divided into two major parts: (1) the memory layer; and (2) the CRF layer. The memory layer can be further broken down into three main components: (a) the input memory $\boldsymbol{m}_{1:t}$; (b) the output memory $\boldsymbol{c}_{1:t}$; and (c) the current input $\boldsymbol{u}_t$, which represents the current step (also known as the "question" in the context of MemN2N). The input and output memory representations are connected via an attention mechanism whose weights are determined by measuring the similarity

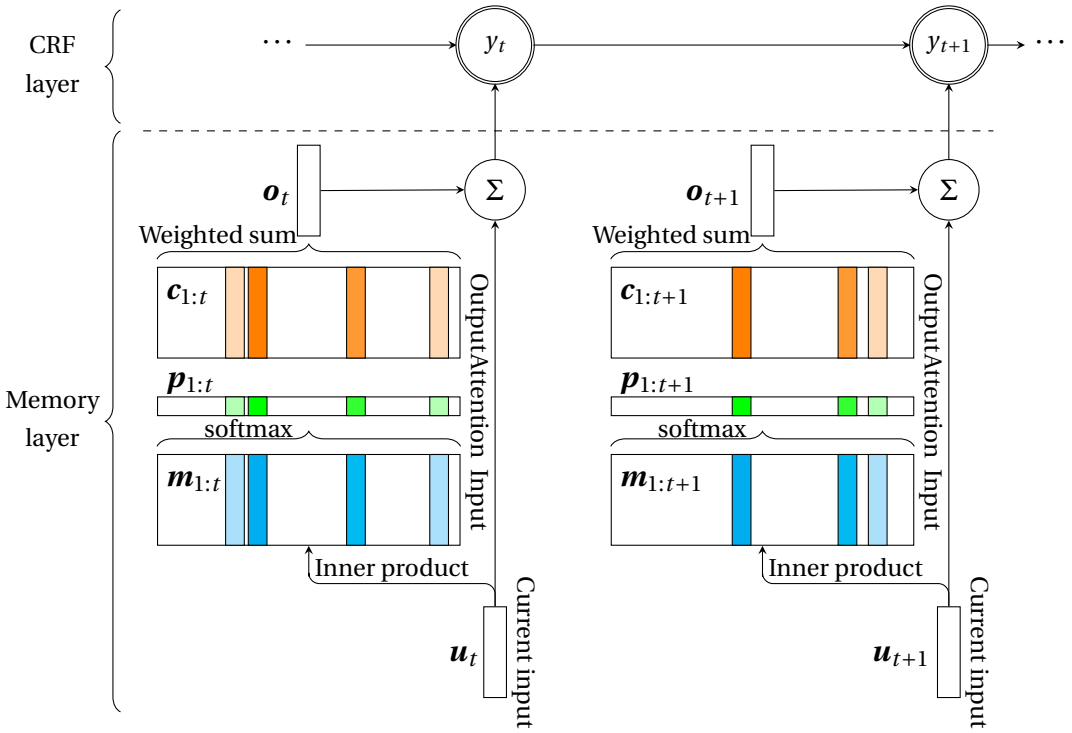

Figure 4.3 Illustration of ME-CRFs with a single memory hop, showing the network architecture at time step t and $t+1$.

between the input memory and the current input. The CRF layer, on the other hand, takes the output of the memory layer as input. In the remainder of this section, we detail the elements of ME-CRF.

4.3.1 Memory Layer

4.3.1.1 Input Memory

Every element (word/post/dialog turn) in a sequence $\mathbf{x}$ is encoded with $\boldsymbol{x}_t = \Phi(x_t)$, where $\Phi(\cdot)$ can be any encoding function mapping the input x_t into a vector $\boldsymbol{x}_t \in \mathbb{R}^d$. This results in the sequence $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$. While this new sequence can be seen as the memory in the context of MemN2Ns, one major drawback of this approach, as pointed out by Seo et al. (2017), is the insensitivity to temporal information between memory cells. We therefore follow Xiong et al. (2016) in injecting temporal signal into the memory using a

bi-directional GRU encoding (Cho et al., 2014a) (see Section 2.2.1):

$$\overrightarrow{\boldsymbol{m}}_t = \overrightarrow{\text{GRU}}(\boldsymbol{x}_t, \overrightarrow{\boldsymbol{m}}_{t-1}) \tag{4.1}$$

$$\overleftarrow{\boldsymbol{m}}_t = \overleftarrow{\text{GRU}}(\boldsymbol{x}_t, \overleftarrow{\boldsymbol{m}}_{t+1}) \tag{4.2}$$

$$\boldsymbol{m}_t = \tanh(\overrightarrow{\boldsymbol{W}}_m \overrightarrow{\boldsymbol{m}}_t + \overleftarrow{\boldsymbol{W}}_m \overleftarrow{\boldsymbol{m}}_t + \boldsymbol{b}_m) \tag{4.3}$$

where $\overrightarrow{\boldsymbol{W}}_m$, $\overleftarrow{\boldsymbol{W}}_m$ and $\boldsymbol{b}_m$ are trainable parameters. Note that $\overrightarrow{\text{GRU}}$ and $\overleftarrow{\text{GRU}}$ have their own respective sets of weight parameters.

4.3.1.2 Current Input

This is used to represent the current step x_t, be it a word or a post. As in MemN2Ns, we want to enforce the current input to be in the same space as the input memory so that we can determine the attention weight of each element in the memory by measuring the relevance between the two. We denote the current input by $\boldsymbol{u}_t = \boldsymbol{m}_t$.

4.3.1.3 Attention

To determine the attention value of each element in the memory, we measure the relevance between the current step $\boldsymbol{u}_t$ and $\boldsymbol{m}_i$ for $i \in [1, t]$ with a softmax function:

$$p_{t,i} = \text{softmax}(\boldsymbol{u}_t^\top \boldsymbol{m}_i) \tag{4.4}$$

where $\text{softmax}(a_i) = \dfrac{e^{a_i}}{\sum_j e^{a_j}}$. Note that to ensure $\boldsymbol{u}_t$ and $\boldsymbol{m}_i$ are in the same space, we also encode $\boldsymbol{u}_t$ with the same bi-directional GRU described in Equations (4.1) – (4.3).

4.3.1.4 Output Memory

Similar to $\boldsymbol{m}_t$, $\boldsymbol{c}_t$ is the output memory, and is calculated analogously but with a different set of parameters in the GRUs and tanh layers of Equations (4.1)– (4.3). The output memory is used to generate the final output of the memory layer and fed as input to the CRF layer.

4.3.1.5 Memory Layer Output

Once the attention weights have been computed, the memory access controller receives the response $\boldsymbol{o}$ in the form of a weighted sum over the output memory representations:

$$\boldsymbol{o}_t = \sum_i p_{t,i} \boldsymbol{c}_i \tag{4.5}$$

This allows the model to have unrestricted access to elements in previous steps as opposed to a single vector $\boldsymbol{h}_t$ in RNNs, thereby enabling ME-CRFs to detect and effectively incorporate long-range dependencies.

4.3.1.6 Possible Extension to Incorporate Multi-hop Reasoning

For more challenging tasks requiring complex reasoning capabilities with multiple supporting facts from the memory, the model can be further extended by stacking multiple memory hops, in which case the output of the k-th hop is taken as input to the $(k+1)$-th hop:

$$\boldsymbol{u}_t^{(k+1)} = \boldsymbol{o}_t^{(k)} + \boldsymbol{u}_t^{(k)} \tag{4.6}$$

where $\boldsymbol{u}_t^{(k+1)}$ encodes not only information at the current step ($\boldsymbol{u}_t^{(k)}$) but also relevant knowledge from the memory ($\boldsymbol{o}_t^{(k)}$).

In the scope of this chapter, we limit our focus to three tasks: (1) named entity recognition (Section 4.4.1), (2) thread discourse structure prediction (Section 4.4.2), and (3) dialog state tracking (Section 4.4.3). While all requiring capturing long-range dependencies and predicting consistent output labels to various degrees, the reasoning element in these three task is not the main concern. We therefore limit the number of hops to 1.

4.3.2 CRF Layer

Once the representation of the current step $\boldsymbol{u}_t^{(K+1)}$ is computed — incorporating relevant information from the memory (assuming the total number of memory hops is K) — it is

then fed into a CRF layer:

$$s(\mathbf{x}, \mathbf{y}) = \sum_{t=0}^{T} \boldsymbol{A}_{y_t, y_{t+1}} + \sum_{t=1}^{T} \boldsymbol{P}_{t, y_t} \tag{4.7}$$

where $\boldsymbol{A} \in \mathbb{R}^{|\mathscr{Y}| \times |\mathscr{Y}|}$ is the learned CRF transition matrix, $|\mathscr{Y}|$ is the size of the label set, and $\boldsymbol{P} \in \mathbb{R}^{T \times |\mathscr{Y}|}$ is a linearly transformed matrix from $\boldsymbol{u}_t^{(K+1)}$ such that $\boldsymbol{P}_{t,:}^\top = \boldsymbol{W}_s \boldsymbol{u}_t^{(K+1)}$ where $\boldsymbol{W}_s \in \mathbb{R}^{|\mathscr{Y}| \times h}$ with h being the size of $\boldsymbol{m}_t$ and $\boldsymbol{P}_{t,:}$ takes the value of the t-th row of $\boldsymbol{P}$ ($\boldsymbol{P}_{t,:}^\top$ is therefore a column vector of size $|\mathscr{Y}|$). Here, $\boldsymbol{A}_{i,j}$ represents the score of the transition from the i-th tag to the j-th tag whereas $\boldsymbol{P}_{i,j}$ is the score of the j-th tag at time i. Using the scoring function in Equation (4.7), we calculate the score of the sequence $\mathbf{y}$ normalised by the sum of scores of all possible sequences $\tilde{\mathbf{y}}$, and this becomes the probability of the true sequence:

$$p(\mathbf{y}|\mathbf{x}) = \frac{\exp(s(\mathbf{x}, \mathbf{y}))}{\sum_{\tilde{\mathbf{y}} \in \mathbf{Y}_\mathbf{X}} \exp(s(\mathbf{x}, \tilde{\mathbf{y}}))} \tag{4.8}$$

where $\tilde{\mathbf{y}} \in \mathbf{Y}_\mathbf{X}$ iterates through all possible sequences.

We train the model to maximise the probability of the gold label sequence with the following loss function:

$$\mathscr{L} = \log p(\mathbf{y}|\mathbf{x}) \tag{4.9}$$

where $p(\mathbf{y}|\mathbf{x})$ is calculated using the forward–backward algorithm (Baum, 1972, Lafferty et al., 2001), a dynamic programming method to compute the CRF log likelihood efficiently. Note that the model is fully end-to-end differentiable, which requires backprogatation through the forward algorithm.

At test time, the model predicts the output sequence with maximum a posteriori probability:

$$\mathbf{y}^* = \underset{\tilde{\mathbf{y}} \in \mathbf{Y}_\mathbf{x}}{\arg\max}\, p(\tilde{\mathbf{y}}|\mathbf{x}). \tag{4.10}$$

For inference, we adopt the Viterbi algorithm for decoding (Viterbi, 1967), which is similar to the forward–backward algorithm but differs in that it takes the *max* over the probabilities at the previous step as opposed to *sum* (Jurafsky and Martin, 2009).

4.4 Experiments on Discourse Tasks

In this section, we evaluate the effectiveness of ME-CRF on three discourse tasks: (1) named entity recognition (Section 4.4.1), (2) thread discourse structure prediction (Section 4.4.2), and (3) dialog state tracking (Section 4.4.3). On the first two tasks, we show that the integration of static memory to CRFs allows us to better leverage discourse-level knowledge by storing such information in the memory cells, thereby achieving superior performance to a number of strong baselines. On the third task, we demonstrate the benefits of learning the dynamics and evolution of dialog states with CRFs as opposed to modelling such states in isolation as in Section 3.4.3.

4.4.1 Discourse-aware Named Entity Recognition

In this section, we present experiments on the task of named entity recognition, over the CoNLL 2003 English NER shared task dataset (Tjong Kim Sang and De Meulder, 2003). Our interest here is in evaluating the ability of ME-CRF to capture document context, to aid in the identification and disambiguation of named entities. Here, we let the model leverage discourse knowledge. More precisely, in addition to the context presented in the current sentence, we also allow the model to have access to previous sentences within the same document.

4.4.1.1 Dataset and Task

The CoNLL 2003 NER shared task dataset consists of 14,041/3,250/3,453 sentences in the training/development/test set, respectively, extracted from 946/216/231 Reuters news articles from the period 1996–97. The goal is to identify individual token occurrences of named entities, and tag each with its class (e.g., `LOCATION` or `ORGANISATION`). Here, we use the BIO tagging scheme, resulting in a total of 9 classes (e.g., `B-LOCATION`, `I-LOCATION`). In terms of tagging schemes, while some have shown marginal improvements with the more expressive IOBES strategy (Ratinov and Roth, 2009, Dai et al., 2015),[1] we stick to

[1] The additional S and E tags are used to indicate a single-token span and the end of a multi-token span respectively.

the BIO scheme for simplicity and the observation of little improvement between these schemes by Lample et al. (2016).

4.4.1.2 Experimental Setup

We choose $\Phi(x_t)$ to be a lookup function, returning the corresponding embedding $\boldsymbol{x}_t$ of the word x_t. In addition to the word features, we employ a subset of the lexical features described in Huang et al. (2015), based on whether the word:

- starts with a capital letter;
- is composed of all capital letters;
- is composed of all lower case letters;
- contains non initial capital letters;
- contains both letters and digits;
- contains punctuation.

These features are all binary and referred to as $\Phi_l(x_t)$. Similar to the thread structure prediction experiments (see Section 4.4.2), we concatenate $\Phi_l(x_t)$ with $\boldsymbol{x}_t$ to generate the new input $\boldsymbol{x}'$ to the bi-directional GRUs in Equations (4.1) and (4.2).

In order to incorporate information in the document beyond sentence boundaries, we encode every word sequentially in a document with Φ and $\overrightarrow{\text{GRU}}$ and $\overleftarrow{\text{GRU}}$, and store them in the memory $\boldsymbol{m}_i$ and $\boldsymbol{c}_i$ for $i \in [1, t]$, where t is the index of the current word in the document.

Training uses Adam (Kingma and Ba, 2015) with 100 epochs and a batch size of 32. We use the following hyper-parameter settings: word embedding size = 50; the hidden sizes of $\overrightarrow{\text{GRU}}$ and $\overleftarrow{\text{GRU}}$ are 50; $\overrightarrow{\boldsymbol{W}}_m$ and $\overleftarrow{\boldsymbol{W}}_m \in \mathbb{R}^{50\times 50}$; and $\boldsymbol{b}_m \in \mathbb{R}^{50}$. Dropout is applied to all GRU recurrent units on the input and output connections, with a rate of 0.2. We initialise ME-CRF with pre-trained word embeddings and keep them fixed during training. While we report results only on the test set, we use early stopping based on the development set.

4.4.1.3 Evaluation

Evaluation is based on span-level named entity F-score, based on the official CoNLL evaluation script.[2]

We compare against the following baselines:

1. a CRF over hand-tuned lexical features (CRF: Huang et al. (2015))
2. an LSTM and bi-directional LSTM (LSTM and BI-LSTM, respectively: Huang et al. (2015))
3. a CRF taking features from a convolutional neural network as input (CONV-CRF: Collobert et al. (2011))
4. a CRF over the output of either a simple LSTM or bidirectional LSTM (LSTM-CRF and BI-LSTM-CRF, respectively: Huang et al. (2015))

Note that for our word embeddings, while we observe better performance with GLOVE (Pennington et al., 2014) (F-score = 90.0), for fair comparison purposes, we adopt the same SENNA embeddings (Collobert et al., 2011) as are used in the baseline methods.[3]

4.4.1.4 Results

The experimental results are presented in Table 4.1. Results for the baseline methods are based on the published results of Huang et al. (2015) and Collobert et al. (2011). Note that none of the systems in Table 4.1 use external gazetteers, to make the comparison fair. Observe that ME-CRF achieves the best performance, beating all the baselines.

Ablation study. It is important to note the benefits of the proposed memory-augmentation method ME-CRF compared to BI-LSTM-CRF, a model with no access to memory storage of previous sentences in the same document. An improvement of 0.6 in F-score can be

[2] http://www.cnts.ua.ac.be/conll2000/chunking/conlleval.txt

[3] Lample et al. (2016) report a higher result of 90.9 using a BI-LSTM-CRF architecture, but augmented with skip n-grams (Ling et al., 2015) and character embeddings. Due to the differing underlying representation, we exclude it from the comparison.

Model	F-score
CRF	86.1
LSTM	83.7
BI-LSTM	85.2
CONV-CRF	88.7
LSTM-CRF	88.4
BI-LSTM-CRF	88.8
ME-CRF	**89.5**

Table 4.1 NER performance on the CoNLL 2003 English NER shared task dataset. Baseline performance extracted from the published results of Huang et al. (2015).

observed, demonstrating the effectiveness of ME-CRF over the baseline. The following section (Section 4.4.1.5) provides more insights into what the model can learn.

4.4.1.5 Visualisation

To gain a better understanding of what the model has learned, Figure 4.4 presents five examples where ME-CRF focuses on words beyond the current sentence boundaries. In the first example, where the target word is *Juventus* (an Italian soccer team), ME-CRF directs its attention mainly to the occurrence of the same word in a previous sentence and a small fraction to *Manchester* (a UK soccer team, in this context). Note that it does not attend to the other named entity (*Europe*) in that sentence, which is of a different NER class.

In the second example, ME-CRF allocates attention to the same words as the target word in the current sentence. Note that the second occurrence of *Interfax* in the memory is the same occurrence as the first word in the current sentence. While more weight is placed on the second *Interfax*, close to one third of the attention is also assigned to the first occurrence. Given that the memory, $\boldsymbol{m}_i$ and $\boldsymbol{c}_i$, is encoded with bi-directional GRUs, the first *Interfax* should, to some degree, capture the succeeding neighbouring elements: *news agency*.

This is reminiscent of label consistency in the works of Sutton and McCallum (2004) and Finkel et al. (2005), but differs in that the consistency constraint is soft as opposed to

… Manchester United face Juventus in Europe. European champions Juventus (ORGANISATION)

… said might have been started deliberately, Interfax news agency said. Interfax (ORGANISATION)

… on Friday for their friendly against Scotland at Murrayfield more than a year after the 30-year-old wing announced he was retiring following differences over selection. Cuttitta, who trainer George Coste said was certain to play on Saturday week, was named in a 21-man squad lacking only two of the team beaten 54-21 by England at Twickenham (LOCATION)

… The draw mad on Friday pitted Juventus , who beat Dutch champions Ajax Amsterdam 4-2 on penalties in last year's final, against Alex Ferguson's European hopefuls in group C. The other two teams in the group are last season's Cup Winners' Cup runners-up Rapid Vienna and Fenerbahce of Turkey. Juventus meet United in Turin on September 11, with the return match at Old Trafford on November 20. United have dominated the premier league in the 1990s, winning three English championships in four years, but have consistently failed in Europe, crashing out of the European Cup to Galatasaray of Turkey and Spain's Barcelona at their last two attempts. They have not lifted a European Trophy since 1991 when they beat Barcelona in the Cup Winners's Cup final, and their one and only European Cup triumph was way back in 1968, when they beat Benfica of Portugal 4-1 at Wembley. Juventus (ORGANISATION)

Two Iranian opposition leaders meet in Baghdad . An Iranian exile group based in Iraq vowed on Thursday to extend support to Iran's Kurdish rebels after they were attacked by Iranian troops deep inside Iraq last month. A Mujahideen Khalq statement said its leader Massoud Rajavi met in Baghdad the Secretary-General of the Kurdistan Democratic Party of Iran (KDPI) Hassan Rastegar on Wednesday and voiced his support to Iran's rebel Kurds. "Rajavi emphasised that the Iranian Resistance would continue to stand side by side with their Kurdish compatriots and the resistance movement in Iranian Kurdistan ," it said. A spokesman for the group said the meeting "signals a new level of cooperation between Mujahideen Khalq and the Iranian Kurdish oppositions". Iran heavily bombarded targets in northern Iraq in July in pursuit of KDPI guerrillas based in Iraqi Kurdish areas outside the control of the government in Baghdad (LOCATION)

Figure 4.4 NER examples showing learned attention to long-range contextual dependencies. The last word in each example is the target word w_t, followed by its NER label in parentheses (in all cases, shared with attended items). Colour-intensity indicates attention value.

hard in previous studies, and automatically learned without the use of heuristics. Notice that this soft consistency places focus on similar entities in the same class beyond those identified by simple lexical overlap-based heuristics. As shown in the rest of the examples in Figure 4.4, the attention mechanism guides the model to find similar entities in previous sentences regardless of the distance to the target word (the last word in each example): (1) *Twickenham* and *Murrayfield*, two rugby stadiums in the UK, (2) *Juventus*, *Fenerbahce* and *Benfica*, soccer clubs in Europe, (3) *Baghdad*, the capital of *Iraq*, as well as neighbouring countries, e.g., *Iran* and *Kurdistan*.

4.4.2 Thread Discourse Structure Prediction

Next, we describe how ME-CRFs can be applied to the task of thread discourse structure prediction, wherein we attempt to predict which post(s) a given post directly responds to, and in what way(s) (as captured by dialogue acts). ME-CRFs are a novel approach to this problem and capable of natively handling both tasks within the same network architecture.

4.4.2.1 Dataset and Task

We adopt the dataset of Kim et al. (2010),[4] which consists of 315 threads and 1,332 posts, collected from the Operating System, Software, Hardware and Web Development sub-forums of CNET (see Section 2.6.2.3 for a detailed description of this dataset and task).[5] Every post has been manually linked to preceding post(s) in the thread that it is a direct response to (in the form of "links"), and the nature of the response for each link (in the form of "dialogue acts", or "DAs"). In this dataset, it is not uncommon to see messages respond to posts which occur much earlier in the thread (based on the chronological ordering of posts). In fact, 18% of the posts link to posts other than their immediately preceding post.

[4] http://people.eng.unimelb.edu.au/tbaldwin/resources/conll2010-thread/
[5] http://forums.cnet.com/

The task is defined as follows: given a list of preceding posts $x_1, \ldots, x_{t-1}$ and the current post x_t, classify which posts it links to (l_t) and the dialogue act (y_t, including 12 types in total) of each such link. In the scope of this task, ME-CRFs are capable of modelling both tasks natively, and therefore a natural fit for this problem.

4.4.2.2 Experimental Setup

In this dataset, in addition to the body of text, each post is also associated with a title. We therefore use two encoders, $\Phi_{title}(\cdot)$ and $\Phi_{text}(\cdot)$, to process them separately and then concatenate $\boldsymbol{x}_t = [\Phi_{title}(x_t)^\top; \Phi_{text}(x_t)^\top]^\top$. Here, $\Phi_{title}(\cdot)$ and $\Phi_{text}(\cdot)$ take word embeddings as input, processing each post at the word level, as opposed to the post-level bi-directional GRU in Equations (4.1) and (4.2), and the representation of a post $\boldsymbol{x}_t$ (either title or text) is obtained by transforming the last and first hidden states of the forward and backward word-level GRU, similar to Equation (4.3). Note that $\Phi_{title}(\cdot)$ and $\Phi_{text}(\cdot)$ do not share parameters. As in Tang et al. (2016b), we further restrict $\boldsymbol{m}_i^{(k)} = \boldsymbol{c}_i^{(k)}$ to curb overfitting.

In keeping with Wang (2014), we complement the textual representations with hand-crafted structural features $\Phi_s(x_t) \in \mathbb{R}^2$:

- initiator: a binary feature indicating whether the author of the current post is the same as the initiator of the thread,
- position: the relative position of the current post, as a ratio over the total number of posts in the thread;

Also drawing on Wang (2014), we incorporate punctuation-based features $\Phi_p(x_t) \in \mathbb{R}^3$: the number of question marks, exclamation marks and URLs in the current post. The resultant feature vectors are projected into an embedding space by $\boldsymbol{W}_s$ and $\boldsymbol{W}_p$ and concatenated with $\boldsymbol{x}_i$, resulting in the new $\boldsymbol{x}_i'$. Subsequently, $\boldsymbol{x}_i'$ is fed into a post-level bi-directional GRUs to obtain $\boldsymbol{m}_i$.

For link prediction, we generate a supervision signal from the annotated links, guiding the attention to focus on the correct memory position:

$$\mathscr{L}_{\text{LNK}} = \sum_{t=1}^{T} \text{CrossEntropy}(\boldsymbol{l}_t, \boldsymbol{p}_t) \tag{4.11}$$

where $\boldsymbol{l}_t$ is a one-hot vector indicating where the link points to for the t-th post in a thread, and $\boldsymbol{p}_t = \{p_{t,1}, \ldots, p_{t,t}\}$ is the predicted distribution of attention over the t posts in the memory.[6] This is used in conjunction with the CRF log likelihood (defined in Equation (4.9)), which is detailed in Equation (4.12). To accommodate the first post in a thread, as it points to a virtual "head" post, we set a dummy post, $\boldsymbol{m}_0 = \mathbf{0}$, of the same size as $\boldsymbol{m}_i$. While the dataset contains multi-headed posts (posts with more than one outgoing link), following Wang (2014), we only include the most recent linked post during training, but evaluate over the full set of labelled links.

For this task, ME-CRF is jointly trained to predict both the link and dialogue act with the following loss function:

$$\mathscr{L}' = \alpha \mathscr{L}_{\text{DA}} + (1 - \alpha)\mathscr{L}_{\text{LNK}} \tag{4.12}$$

where $\mathscr{L}_{\text{DA}}$ is the CRF likelihood defined in Equation (4.9), and α is a hyper-parameter for balancing the emphasis between the two tasks.

Training uses Adam (Kingma and Ba, 2015) with 50 epochs and a batch size of 32. We use the following hyper-parameter settings: word embedding size = 20, $\boldsymbol{W}_p \in \mathbb{R}^{100\times 3}$, $\boldsymbol{W}_s \in \mathbb{R}^{50\times 2}$, $\alpha = 0.5$, the hidden sizes of Φ_{title} and Φ_{text} are 20, the hidden sizes of the post-level $\overrightarrow{\text{GRU}}$ and $\overleftarrow{\text{GRU}}$ are 50. Dropout is applied to all GRU recurrent units on the input and output connections with a rate of 0.3.

Lastly, we also explore the idea of curriculum learning (Bengio et al., 2009), by fixing the CRF transition matrix $\boldsymbol{A} = \mathbf{0}$ for the first $e = 20$ epochs, after which we train the parameters for the remainder of the run. This allows the ME-CRF to learn a good strategy for DA and

[6] While it is possible to generate similar supervision signals for other tasks, for example NER, doing so inevitably requires heuristics (e.g., string matching) or coreference resolution to identify expressions referring to the same entity, which may result in noisy supervision and damage model performance.

link prediction, as an independent "maxent" type classifier, before attempting to learn sequence dynamics. We refer to this variant as "ME-CRF$_{\text{CURRL}}$".

4.4.2.3 Evaluation

Following Wang (2014), we evaluate based on post-level micro-averaged F-score. All experiments were carried out with 10-fold cross validation, stratifying at the thread level.

We benchmark against the following previous work: the feature-rich CRF-based approach of Kim et al. (2010), where the authors trained independent models for each of link and DA classification ("CRF$_{\text{KIM}}$"); the feature-rich CRF-based approach of Wang (2014), where the author further extended the feature set of Kim et al. (2010) and jointly trained a CRF over the link and DA prediction tasks ("CRF$_{\text{WANG}}$"); and the dependency parser-based approach of Wang (2014), where the author treated the discourse structure prediction task as a constrained dependency parsing problem, with posts as nodes in the dependency graph, and the constraint that links must connect to preceding posts in the thread ("DEPPARSER").[7] In addition to the CRF/parser-based systems, we also build a MEMN2N-based baseline (named MEMN2N) where MEMN2N shares the architecture of the memory layer in ME-CRF but excludes the use of the CRF layer. Instead, MEMN2N, following the work of Sukhbaatar et al. (2015), predicts the final answer by:

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{W}_{\text{DA}}(\boldsymbol{u}^{(K)} + \boldsymbol{o}^{(K)})) \tag{4.13}$$

where $\hat{\boldsymbol{y}}$ is the predicted DA distribution, $\boldsymbol{W}_{\text{DA}} \in \mathbb{R}^{|\mathscr{Y}| \times d}$ is a parameter matrix for the model to learn, and $K = 1$ is the total number of hops. This is equivalent to classifying link and DA independently at each time step t without taking transitions between DA labels into account.

4.4.2.4 Results

The experimental results are presented in Table 4.2.

[7] Note that a mistake was found in the results in the original paper Wang et al. (2011), and we use the corrected results from Wang (2014).

Model	Link	DA	Joint
CRF_{KIM}	86.3	75.1	—
CRF_{WANG}	82.3	73.4	66.5
DEPPARSER	85.0	75.7	70.6
MEMN2N	85.8	76.0	69.5
ME-CRF	**86.4**	**77.5**	70.9
$\text{ME-CRF}_{\text{CURRL}}$	86.3	77.4	**71.2**

Table 4.2 Post-level Link and DA F-scores. Performance for ME-CRF and $\text{ME-CRF}_{\text{CURRL}}$ is macro-averaged over 5 runs. The first three rows are the three baseline systems.

State-of-the-art post-level results. ME-CRFs achieve state of the art results in terms of joint post-level F-score, substantially better than the baselines. While ME-CRF slightly outperforms the current state-of-the-art (DEPPARSER), $\text{ME-CRF}_{\text{CURRL}}$ improves the performance and achieves a further 0.3% absolute gain.

Curriculum learning improves joint prediction. Despite the slight performance drop on the DA and link prediction tasks, $\text{ME-CRF}_{\text{CURRL}}$, with the CRF transition matrix frozen for the first 20 epochs, achieves a ~0.3% absolute gain in joint F-score over ME-CRF. This suggests that sequence dynamics between posts, while difficult to capture, are beneficial to the overall task (resulting in more coherent DA and link predictions) if trained with proper initialisation.

MEMN2N vs. ME-CRFs. We see consistent gains across all three tasks, underlining the benefits of the CRF layer in modelling the transitions between DA labels.

CRF vs. ME-CRFs. Note that CRF_{KIM} is not trained jointly on the two component tasks, but individually on each task. Without additional data, jointly training on the two tasks generates results that are comparable or substantially better over the individual tasks. This highlights the effectiveness of ME-CRF, especially with the link prediction performance comparable to that of a single-task model CRF, and surpassing it in the case of DA.

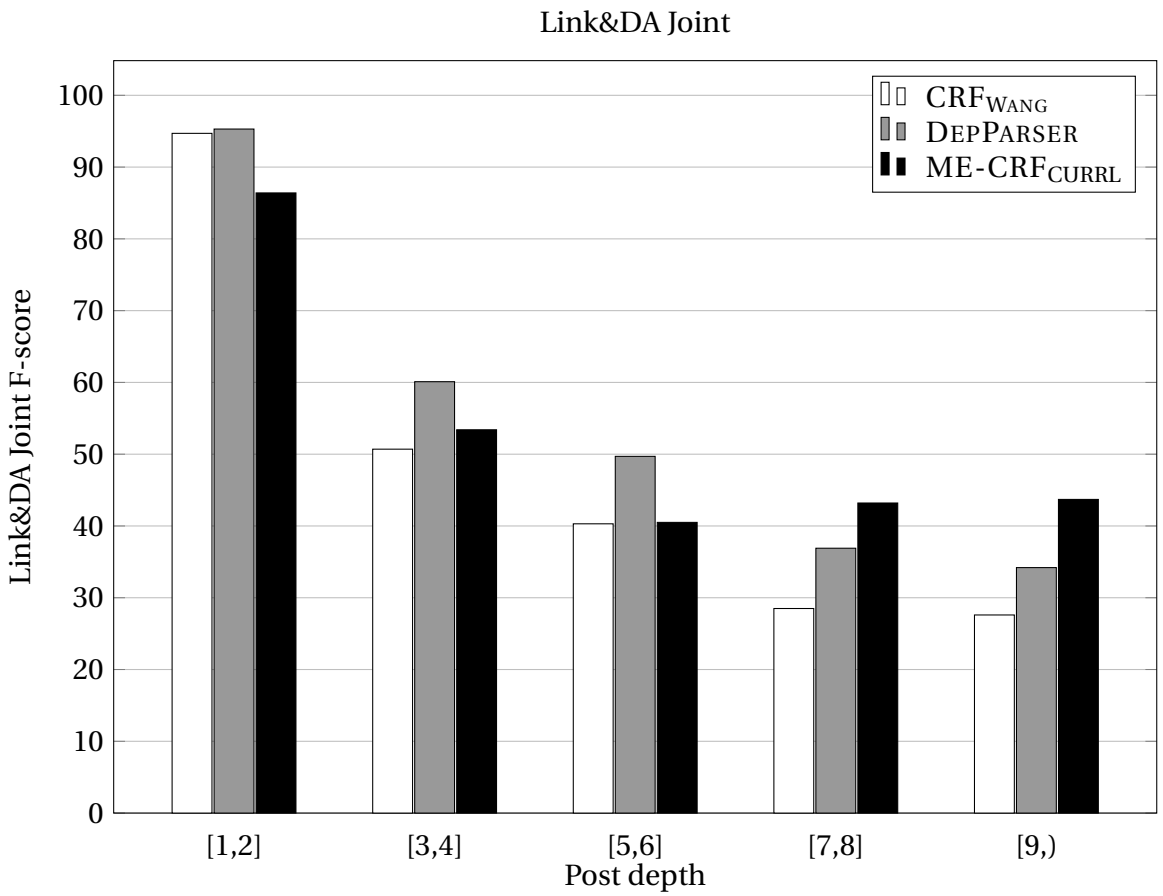

Figure 4.5 Breakdown of post-level Joint F-scores by post depth, where e.g. "[1,2]" is the joint F-score over posts of depth 1–2, i.e. the first or second post in the thread. Note that we take the reported performance of CRF$_{WANG}$ and DEPPARSER from Wang (2014).

4.4.2.5 Analysis

We break the performance down by the depth of each post in a given thread, and present the results in Figure 4.5. Although below the baselines for the interval [1,2], ME-CRF$_{CURRL}$ consistently outperforms CRF$_{WANG}$ from depth 3 onwards, and is superior to DEPPARSER for depths [7,). Breaking down the performance further to the individual tasks of Link and DA prediction, as displayed in Figure 4.6, we observe a similar trend. In line with the findings in the work of Wang (2014), this confirms that prediction becomes progressively more difficult as threads grow longer, which is largely due to the increased variability in discourse structure. Despite the escalated difficulty, ME-CRF$_{CURRL}$ is substantially superior to the baselines when classifying deeper posts.

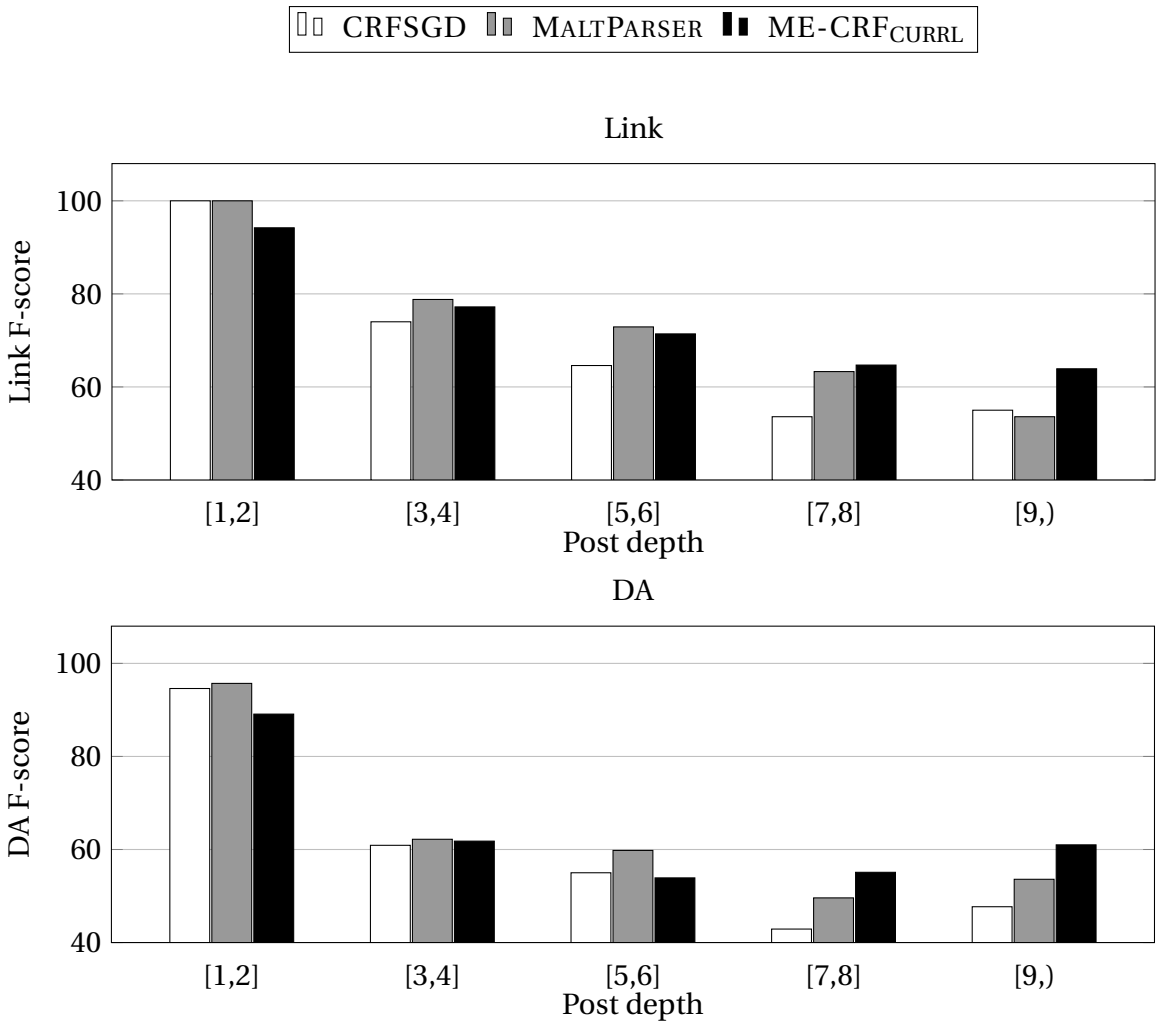

Figure 4.6 Breakdown of post-level Link and DA F-scores by post depth.

Between the CRF-based models, it is worth noting that despite the lower performance for [1,2], ME-CRF$_{\text{CURRL}}$ benefits from having global access to the entire sequence, and consistently outperforms CRF$_{\text{WANG}}$ for depths [3,), highlighting the effectiveness of the memory mechanism. Overall, these results validate our hypothesis that having unrestricted access to the whole sequence is beneficial, especially for long-range dependencies, offering further evidence of the power of ME-CRFs.

4.4.3 Dialog State Tracking (DSTC-2)

Lastly, we present experiments in a third setting: dialog state tracking, over the DSTC-2 dataset (Henderson et al., 2013). Our goal here is to evaluate the benefits of learning the dynamics and evolution of dialog states as opposed to modelling them in isolation as in Section 3.4.3.

4.4.3.1 Dataset and Task

We use the DSTC-2 dataset (see Section 2.6.2.2), consisting of 1,612/506/1,117 instances in the train, validation and test set respectively, to test the effectiveness of ME-CRFs. Note that while it is common practice to evaluate using the N-best Automatic Speech Recognition (ASR) hypotheses and their associated ASR scores, our goal here is to demonstrate the benefits of the proposed ME-CRF over the baselines presented in Section 3.4.3. Consistent with the previous chapter, we therefore train and evaluate using only the transcripts as input. Also note, we use the cleaned version of the dataset provided by Mrkšić et al. (2017) who manually rectified numerous spelling errors in the original release of DSTC-2.

In this section, we frame the task as one of sequence labelling where a pair of system response $\mathbf{x}_i^s$ and user request $\mathbf{x}_i^u$ is taken as input and the goal is to predict the values to three slots: area ($\boldsymbol{y}_i^{\text{area}}$), food ($\boldsymbol{y}_i^{\text{food}}$) and price range ($\boldsymbol{y}_i^{\text{price}}$), based on previous user-system utterances including the current step. System responses and user requests are both sequences of words, whose embeddings are denoted $\boldsymbol{x}_{i,j}^s$ and $\boldsymbol{x}_{i,j}^u$ respectively, with j indexing the position of a word in a sequence, such that $\mathbf{x}_i^s = \{\boldsymbol{x}_{i,1}^s, \ldots, \boldsymbol{x}_{i,j}^s, \ldots, \boldsymbol{x}_{i,|\mathbf{x}_i^s|}^s\}$ and $\mathbf{x}_i^u = \{\boldsymbol{x}_{i,1}^u, \ldots, \boldsymbol{x}_{i,j}^u, \ldots, \boldsymbol{x}_{i,|\mathbf{x}_i^u|}^u\}$.

4.4.3.2 Experimental Setup

Similar to the settings in the thread discourse structure prediction experiments in Section 4.4.2, we encode each sequence with a bi-directional GRU:

$$\overrightarrow{\boldsymbol{h}}^{s}_{i,j} = \overrightarrow{\text{GRU}}(\overrightarrow{\boldsymbol{h}}^{s}_{i,j-1}, \boldsymbol{x}^{s}_{i,j}), \qquad \overleftarrow{\boldsymbol{h}}^{s}_{i,j} = \overleftarrow{\text{GRU}}(\overleftarrow{\boldsymbol{h}}^{s}_{i,j+1}, \boldsymbol{x}^{s}_{i,j}) \tag{4.14}$$

$$\overrightarrow{\boldsymbol{h}}^{u}_{i,j} = \overrightarrow{\text{GRU}}(\overrightarrow{\boldsymbol{h}}^{u}_{i,j-1}, \boldsymbol{x}^{u}_{i,j}), \qquad \overleftarrow{\boldsymbol{h}}^{u}_{i,j} = \overleftarrow{\text{GRU}}(\overleftarrow{\boldsymbol{h}}^{u}_{i,j+1}, \boldsymbol{x}^{u}_{i,j}) \tag{4.15}$$

where $\overrightarrow{\boldsymbol{h}}^{s}_{i,j}$ and $\overrightarrow{\boldsymbol{h}}^{u}_{i,j}$ are the forward representations for the j-th word in the i-th dialog turn respectively, $\overleftarrow{\boldsymbol{h}}^{s}_{i,j}$ and $\overleftarrow{\boldsymbol{h}}^{u}_{i,j}$ are the backward counterparts. Note that $\overrightarrow{\text{GRU}}$ and $\overleftarrow{\text{GRU}}$ each has its own set of trainable parameters. We then take the element-wise sum of the representations of the last and first step as the encoding of the sequence:

$$\boldsymbol{h}^{s}_{i} = \overrightarrow{\boldsymbol{h}}^{s}_{i,|\mathbf{x}^{s}_{i}|} + \overleftarrow{\boldsymbol{h}}^{s}_{i,1} \tag{4.16}$$

$$\boldsymbol{h}^{u}_{i} = \overrightarrow{\boldsymbol{h}}^{u}_{i,|\mathbf{x}^{s}_{i}|} + \overleftarrow{\boldsymbol{h}}^{u}_{i,1}, \tag{4.17}$$

and store them in the input and output memory, such that:

$$\boldsymbol{m}_i = \boldsymbol{c}_i = \boldsymbol{h}^{s}_{i} + \boldsymbol{h}^{u}_{i}. \tag{4.18}$$

At each input step, the model tries to predict the values to three slots: area, food and price range. To ensure that the representations of the slots are in the same space as the memory, we apply the same bi-directional GRU to the embeddings of those three slots and also take the element-wise sum of the outputs at the last and first steps as the their encoding, forming the "queries" at each step: $\boldsymbol{u}^{\text{area}}_{i}$, $\boldsymbol{u}^{\text{food}}_{i}$ and $\boldsymbol{u}^{\text{price}}_{i}$. Note that despite sharing the same collection of memory representations $\boldsymbol{m}_i$ and $\boldsymbol{c}_i$, the attention focus may differ depending on the slot, resulting in different output memory representations $\boldsymbol{o}^{\text{area}}_{i}$, $\boldsymbol{o}^{\text{food}}_{i}$ and $\boldsymbol{o}^{\text{price}}_{i}$. Note that the bi-directional GRU is only applied at the word level, not the utterance level, to ensure that the model does not have access to future information.

Prior to the CRF layers, the output from the memory layer is transformed:

$$(\boldsymbol{P}^{\text{area}}_{i,:})^\top = \boldsymbol{W}^{\text{area}}(\boldsymbol{u}^{\text{area}}_i + \boldsymbol{o}^{\text{area}}_i) \tag{4.19}$$

$$(\boldsymbol{P}^{\text{food}}_{i,:})^\top = \boldsymbol{W}^{\text{food}}(\boldsymbol{u}^{\text{food}}_i + \boldsymbol{o}^{\text{food}}_i) \tag{4.20}$$

$$(\boldsymbol{P}^{\text{price}}_{i,:})^\top = \boldsymbol{W}^{\text{price}}(\boldsymbol{u}^{\text{price}}_i + \boldsymbol{o}^{\text{price}}_i) \tag{4.21}$$

where $\boldsymbol{W}^{\text{area}}$, $\boldsymbol{W}^{\text{food}}$ and $\boldsymbol{W}^{\text{price}}$ are trainable weight matrices, and $\boldsymbol{P}^{\text{area}}_{i,:}$ denotes a row in $\boldsymbol{P}^{\text{area}} \in \mathbb{R}^{T \times |\mathscr{Y}|}$ defined in Equation (4.7) ($\boldsymbol{P}^{\text{food}}_{i,:}$ and $\boldsymbol{P}^{\text{price}}_{i,:}$ serve analogous purposes for $\boldsymbol{P}^{\text{food}}$ and $\boldsymbol{P}^{\text{price}}$). Dropout is also applied to $\boldsymbol{P}^{\text{area}}_{i,:}$, $\boldsymbol{P}^{\text{food}}_{i,:}$ and $\boldsymbol{P}^{\text{price}}_{i,:}$ with a rate of 0.4 to reduce overfitting.

Given the number of slots considered in this task, we apply three CRFs, each with its own set of trainable parameters and responsible for a single slot, to capture the dynamics of these slots. This results in three CRF loss terms:

$$\mathscr{L} = \lambda^{\text{area}}\mathscr{L}^{\text{area}} + \lambda^{\text{food}}\mathscr{L}^{\text{food}} + \lambda^{\text{price}}\mathscr{L}^{\text{price}}. \tag{4.22}$$

As the number of possible food values (93) is much greater than that of either area (7) or price range (5), and the task is more challenging for the food CRF, we empirically set λ^{food} to 2.0 and both λ^{area} and λ^{price} to 0.5 to place more emphasis on the food task.

Training uses Adam (Kingma and Ba, 2015) with 50 epochs, a batch size of 32, and a learning rate of 8×10^{-4}. We set the embedding size to 300 and the hidden sizes of the bi-directional GRUs in Equations (4.14) and (4.15) to 300 as well. Dropout is applied to all GRU recurrent units on the input and output connections with a rate of 0.5. To account for OOV words in the validation and test sets, we randomly mask 10% of the words in the training partition.

Model	Area	Food	Price	Joint
MEMN2N + ADJ	93.3	87.3	89.1	73.5
MEMN2N + LW	93.7	91.4	90.6	79.8
UMEMN2N	95.8	**93.3**	89.9	82.1
MEMN2N + LW + BI-GRU	95.9	92.6	90.5	80.2
ME-CRF	**96.8**	93.1	**91.8**	**84.0**

Table 4.3 Average accuracy (%) of various models on the DSTC-2 dataset over 5 runs. Baseline models take the performance reported in Section 3.4.3.

4.4.3.3 Evaluation

For evaluation, we measure accuracy (marco-averaged over 5 runs with different random seeds) following Henderson et al. (2013).[8] We reuse the baselines presented in Section 3.4.3.2: (1) a MEMN2N with adjacent weight tying (MEMN2N + ADJ); (2) a MEMN2N with layer-wise weight tying (MEMN2N + LW); and (3) a MEMN2N with unified weight tying (UMEMN2N). Apart from the addition of the CRF layer, the configuration of the memory layer of ME-CRF in this experiment renders it equivalent to a MEMN2N with layer-wise weight tying with the exception of the difference in sentence-level representation. With MEMN2N + LW, sentences are encoded by first multiplying word embeddings by the positional encoding element-wise (see Section 2.3.2.4) to include word order information and then taking the sum of such word embeddings. In contrast, ME-CRFs make use of a bi-directional GRU to perform sentence-level encoding. To better study and isolate the source of the performance gains, we also develop another baseline: MEMN2N + LW + BI-GRU with the same configuration and training settings as the memory layer of ME-CRF, that is, a single memory hop with BI-GRU-based sentence-level encoding.

4.4.3.4 Results

The experimental results are shown in Table 4.3. We take the baseline results reported in Section 3.4.3. Performance is measured by accuracy macro-averaged over 5 runs with different random seeds for all baselines and ME-CRF.

[8] Under schedule 2 and label scheme A: only turns where the value to a slot has been filled are included and slot states are accumulated forwards through the dialog.

ME-CRF vs. UMemN2N. ME-CRF achieves the best overall performance against the selected baselines, improving the joint accuracy by 1.9% absolute over UMemN2N. Looking at the sub-categories, the performance boost in the joint category can largely be attributed to the gains in the area and price range slots. The slightly inferior performance in the food slot is likely due to the large search space of the possible food values (93 as opposed to 7 and 5 for area and price range respectively), causing data sparsity and therefore learning difficulties.

ME-CRF vs. MemN2N + LW + Bi-GRU. Despite the minor improvements in the overall joint category, the performance of MemN2N + LW + Bi-GRU remains largely comparable with that of MemN2N + LW, suggesting that the addition of Bi-GRU for sentence-level processing does not offer much gain. On the other hand, it is obvious that taking the dynamics and transitions between neighbouring steps into account with a CRF layer is beneficial to the overall performance of the task, as evidenced by the performance boost of ME-CRF over MemN2N + LW + Bi-GRU, highlighting the effectiveness of the proposed method.

4.4.3.5 Analysis and Discussion

To better understand the source of the performance gains of ME-CRF over MemN2N + LW + Bi-GRU, we conduct further evaluation at the discourse level, that is, a discourse output of the entire sequence of utterances is considered correct if and only if all predictions, at every dialog turn, are correct.[9] The discourse-level performance is presented in Table 4.4.

The new discourse-level evaluation is admittedly much more strict, causing accuracy in all categories to drop to various degrees compared to that in Table 4.3. While the area and price range categories are less affected, the decline in food is more obvious, likely due to its large number of possible slot values: 93 as opposed to < 10 in the other two categories, making the task even more challenging at the discourse level. Despite the performance drop, the gap between the two models is wider than that at the dialog turn

[9] Also subject to the same evaluation standard as in Section 4.4.3.3: under schedule 2 and label scheme A.

Model	Area	Food	Price	Joint
MEMN2N + LW + BI-GRU	93.4	79.3	88.1	57.0
ME-CRF	**96.1**	**87.3**	**89.0**	**67.7**

Table 4.4 Average discourse-level accuracy (%) on the DSTC-2 dataset over 5 runs.

level in almost all categories. The only exception is price range where the performance differences between the two models at the dialog turn and discourse level are comparable at ≈ 1.0. The more pronounced superiority of ME-CRF at the discourse level can largely be attributed to the addition of the CRF layer, allowing the model to make more consistent predictions than MEMN2N + LW + BI-GRU.

The improved prediction consistency is further illustrated in Table 4.5 where we provide an error analysis of MEMN2N + LW + BI-GRU vs. ME-CRF with four examples. In the first two examples, the baseline model makes the mistake of impossible transitions of slot values from specified to unspecified, which is unattainable according to the definition of the task. This is mainly caused by the fact that MEMN2N + LW + BI-GRU makes predictions independently at each step and does not take into account the likelihood of transitions between neighbouring dialog turns. In contrast, ME-CRF, with the help of the CRF, in particular the transition matrix $\boldsymbol{A}$ defined in Equation (4.7), models such transitions and is aware of the impossibility of an already specified slot followed by a *None* value in the very next dialog turn. In the bottom two examples, when asked to confirm the preferred type of food, MEMN2N + LW + BI-GRU is liable to change, producing inconsistent outputs and resulting in incorrect predictions. ME-CRF, on the other hand, generates more consistent results with the aid of the added CRF. The globally optimised nature of the CRF training objective helps in learning the likelihood of transitions between possible slot values so that, at decoding time, such improbable predictions (e.g., *north american* and *don't care* in the third and fourth examples) can be eliminated by the Viterbi algorithm due to low transition probability scores.

User	Agent	MemN2N + LW + Bi-GRU Food	MemN2N + LW + Bi-GRU Price	MemN2N + LW + Bi-GRU Area	ME-CRF Food	ME-CRF Price	ME-CRF Area
	…	…	…	…	…	…	…
…							
	kymmoy serves asian oriental food in the expensive price range	asian oriental	expensive	don't care	asian oriental	expensive	don't care
can i have the phone number							
	The phone number of kymmoy is 01223 311911	asian oriental	expensive	don't care	asian oriental	expensive	don't care
and what area is that in							
	kymmoy is a nice place in the centre of town	asian oriental	expensive	–	asian oriental	expensive	don't care
thank you good bye							
	Hello and welcome	vietnamese	moderate	–	vietnamese	moderate	–
a moderately priced restaurant that serves vietnamese food							
	You are looking for a vietnamese restaurant right ?	vietnamese	moderate	–	vietnamese	moderate	–
yes							
	thanh binh serves vietnamese food	vietnamese	–	–	vietnamese	moderate	–
is it moderately priced							
	…	…	…	…	…	…	…
…							
	Hello and welcome	russian	expensive	–	russian	expensive	–
can i find an expensive russian food restaurant							
	What kind of food would you like ?	russian	expensive	–	russian	expensive	–
russian							
	What kind of food would you like ?	north american	expensive	–	russian	expensive	–
russian							
	You are looking for a restaurant serving any kind of food right ?	british	expensive	–	british	expensive	–
can i have a british restaurant							
	…	…	…	…	…	…	…
…							
	Hello and welcome	hungarian	–	–	hungarian	–	–
hungarian							
	What kind of food would you like ?	hungarian	–	west	hungarian	–	west
west side							
	You are looking for a restaurant serving any kind of food right ?	don't care	–	west	hungarian	–	west
no							
	You are looking for a restaurant serving any kind of food right ?	indian	–	west	indian	–	west
indian							
	…	…	…	…	…	…	…
…							

Table 4.5 Four examples of ME-CRF + LW + Bi-GRU vs. ME-CRF. User and agent utterances are left and right aligned on the left half of the table. Predicted slot values are on the right half of the table. Double horizontal lines separate examples. – indicates not specified. Red indicates incorrect slot value.

4.5 Summary

In this chapter, we have addressed the second research question of enhancing modelling capacity by integrating external memory to better capture long-range contextual dependencies. To this end, we have presented ME-CRF, a model extending linear-chain CRFs by including external memory. This allows the model to look beyond neighbouring items and access long-range context. Experimental results demonstrated the effectiveness of the proposed method over three tasks: named entity recognition, forum thread discourse analysis, and dialog state tracking. The performance gains can mainly be attributed to the incorporation of static memory, enabling the model to leverage contexts of much larger size, looking beyond the boundaries of sentences. From the perspective of static memory networks, the addition of CRFs allows MEMN2Ns to learn the evolution of states between neighbouring steps, thereby generating predictions with improved consistency and further boosting the performance of memory-augmented models on the task of dialog state tracking. In both cases, the proposed model offers clear improvements over a collection of strong baselines while demonstrating its wide applicability to a diverse set of tasks.

With the advent of large-scale pre-trained TRANSFORMER-based language encoders, such as BERT, it is unclear whether the incorporation of CRFs is still necessary. While it is possible that the benefits of linear-chain CRFs may now be implicitly learned by such pre-trained language models, this remains an open question and requires further investigation.

In the next chapter, we consider a different type of memory, dynamic memory, where the contents of memory cells are dynamically updated depending on the relevance of the input. In particular, we shift our focus to recurrent entity networks (ENTNETS) (Henaff et al., 2017).

Chapter 5

Relaxing Limitations of Dynamic Memory Chains

Sentences in a discourse often involve multiple entities and describe the interactions between them, forming an important aspect of discourse. More specifically, modelling entities in a discourse requires one to identify and keep track of the dynamics of such entities throughout the narrative, focusing particularly on how they are referenced and interact with one another (Eisenstein, 2018). To this end, many have made impressive progress on entity-centric models (see Section 2.4.2.1), ranging from a "reference-aware" language model (Yang et al., 2017), to a generative entity-aware approach to coreference resolution (Ji et al., 2017). This chapter focuses on Recurrent Entity Networks (EntNets) (Henaff et al., 2017) (see Section 2.4.2) as a means to address the third research question of how we can relax limitations of (dynamic) memory networks to better capture entities and their interactions. EntNets are equipped with multiple memory chains and dynamically update them to keep track of the most up-to-date representations of entities in a discourse. Compared to static memory networks, where update is performed by pushing information through a memory controller, EntNets update each memory cell independently upon seeing relevant input. Preliminary experimental results on bAbI indicate strong generalisation capabilities of EntNets, superior to a range of baselines, including static memory-based models, such as MemN2N (Sukhbaatar et al., 2015).

Initially designed for the task of reading comprehension, ENTNETs are, however, somewhat limited in their applicability to a wide variety of discourse tasks. For tasks requiring fine-grained information extraction, the difference in the input granularity (sentence vs. word) causes ENTNETs to overlook critical word-level signals, rendering the model impractical for such tasks. In addition, while ENTNETs can be trained end-to-end, doing so requires the model to learn the implicit distribution of responsibility across a number of memory chains to track the diverse set of entities in a discourse. This is a challenging task on its own, one that requires further exploration of ideas such as using a sparse coefficient vector to make key embeddings context-dependent (Liang et al., 2019). If not done properly, it may lead to ambiguous and improper distribution of responsibility among memory chains as to which entity should be captured by which memory chain, causing performance degradation, as we show in Section 5.3.

The goal of this chapter is to rectify the shortcomings identified above, improving the performance of ENTNETs in discourse understanding. We explore two orthogonal directions, both build on the underlying idea of ENTNET, i.e., tracking the states of entities, either physical or virtual, individually and independently, enabled with the help of external memory augmentation. With ENTNETs serving as the cornerstone, the improvements allow one to not only better model entity interactions (Section 5.2), a crucial element in discourse, but also store diversified knowledge stably over long distances as irrelevant inputs are discarded, thanks to a gating mechanism (Section 5.3).

This chapter is based on and extends the following papers:

> Fei Liu, Trevor Cohn and Timothy Baldwin. 2018. Recurrent Entity Networks with Delayed Memory Update for Targeted Aspect-based Sentiment Analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies (NAACL-HLT 2018)*, pages 278–283, New Orleans, USA.
>
> Fei Liu, Trevor Cohn and Timothy Baldwin. 2018. Narrative Modeling with Memory Chains and Semantic Supervision. In *Proceedings of the 56th Annual Meeting of the*

Association for Computational Linguistics (ACL 2018), pages 278–284, Melbourne, Australia.

While this chapter is made up of two conceptually distinct approaches, they build on the same underlying model architecture (ENTNET) and are bound together by the common goal of better modelling entities in the context of discourse.

5.1 Brief Overview of ENTNET

Section 5.1 reviews the architecture of Recurrent Entity Networks (ENTNETs) (Henaff et al., 2017). For a more detailed description of the model, we refer the reader to Section 2.4.2. Our novel contributions, building on ENTNETs, are presented subsequently in Sections 5.2 and 5.3. For those already familiar with MEMN2N, feel free to skip this section.

The key motivation of ENTNETs is the idea of tracking the state of the world with external memory. Developed in the context of reading comprehension, an ENTNET maintains a number of "memory chains" — one for each entity — and dynamically updates the representations of them as it progresses through a story. In this chapter, we extend the concept of "entity", mainly referring to physical entities in the original work of Henaff et al. (2017), to include a wider range of subjects such as different aspects and perspectives of a discourse. An illustration of an ENTNET with n memory chains is provided in Figure 5.1, and the inner workings of an ENTNET recurrent unit is shown in Figure 5.2.

Memory update candidate. At every time step i, the internal memory update candidate $\vec{\tilde{\boldsymbol{m}}}_i^{(j)}$ for the j-th memory chain is computed as:

$$\vec{\tilde{\boldsymbol{m}}}_i^{(j)} = \phi(\vec{\boldsymbol{U}}\vec{\boldsymbol{m}}_{i-1}^{(j)} + \vec{\boldsymbol{V}}\boldsymbol{k}^{(j)} + \vec{\boldsymbol{W}}\boldsymbol{x}_i) \tag{5.1}$$

where $\boldsymbol{k}^{(j)}$ is the embedding for the j-th entity (key), $\vec{\boldsymbol{U}}$, $\vec{\boldsymbol{V}}$ and $\vec{\boldsymbol{W}}$ are trainable parameters, and ϕ is the parametric ReLU (He et al., 2015) (see Section 2.4.2.3).

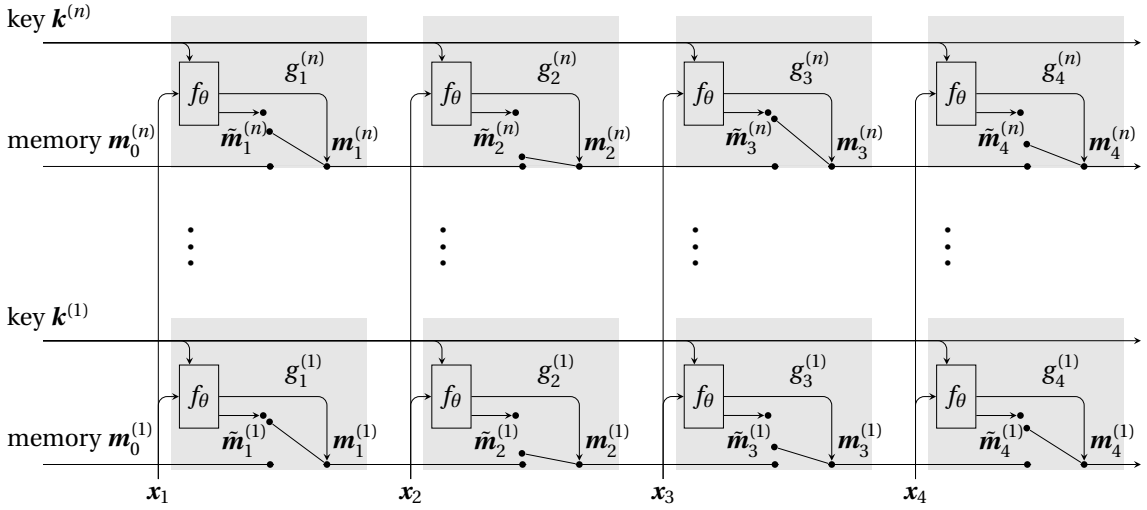

Figure 5.1 Illustration of an instance of EntNet with n memory chains, reproduced from Figure 2.3.

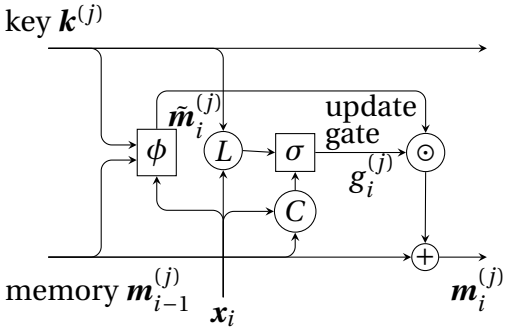

Figure 5.2 Illustration of an EntNet recurrent unit, depicting the inner workings of a shaded recurrent cell in Figure 5.1. Figure reproduced from Figure 2.4.

Memory update. The update of each memory chain is controlled by a gating mechanism:

$$\overrightarrow{g}_i^{(j)} = \sigma(\boldsymbol{x}_i \cdot \boldsymbol{k}^{(j)} + \boldsymbol{x}_i \cdot \overrightarrow{\boldsymbol{m}}_{i-1}^{(j)}) \tag{5.2}$$

$$\overrightarrow{\boldsymbol{m}}_i^{(j)} = \overrightarrow{\boldsymbol{m}}_{i-1}^{(j)} + \overrightarrow{g}_i^{(j)} \odot \overrightarrow{\tilde{\boldsymbol{m}}}_i^{(j)} \tag{5.3}$$

where $\odot$ denotes the Hadamard product (resulting in the unnormalised memory representation $\overrightarrow{\tilde{\boldsymbol{m}}}_i^{(j)}$), and σ is the sigmoid function. The gate $\overrightarrow{g}_i^{(j)}$ controls how much the memory chain should be updated, a decision factoring in two elements: (1) the "location" term, quantifying how related the current input $\boldsymbol{x}_i$ is to the key $\boldsymbol{k}^{(j)}$ (identity) of the entity being tracked; and (2) the "content" term, measuring the similarity between the input and the current state $\overrightarrow{\boldsymbol{m}}_{i-1}^{(j)}$ of the entity tracked by the j-th memory chain.

Normalisation. We normalise each memory representation:

$$\overrightarrow{\boldsymbol{m}}_i^{(j)} = \frac{\overrightarrow{\tilde{\boldsymbol{m}}}_i^{(j)}}{\|\overrightarrow{\tilde{\boldsymbol{m}}}_i^{(j)}\|} \tag{5.4}$$

where $\|\overrightarrow{\tilde{\boldsymbol{m}}}_i^{(j)}\|$ denotes the Euclidean norm of $\overrightarrow{\tilde{\boldsymbol{m}}}_i^{(j)}$. In doing so, we allow the model to forget in the sense that, as $\overrightarrow{\boldsymbol{m}}_i^{(j)}$ is constrained to be lying on the surface of the unit sphere, adding new information carried by $\overrightarrow{\tilde{\boldsymbol{m}}}_i^{(j)}$ and then performing normalisation inevitably causes forgetting in that the cosine distance between the original and updated memory decreases.

Bi-directionality. In contrast to the vanilla ENTNET model, we apply the above steps in both directions, scanning a story both forward and backward, with arrows denoting the processing direction. $\overleftarrow{\boldsymbol{m}}_i^{(j)}$ is computed analogously to $\overrightarrow{\boldsymbol{m}}_i^{(j)}$ but with a different set of parameters, and

$$\boldsymbol{m}_i^{(j)} = \overrightarrow{\boldsymbol{m}}_i^{(j)} + \overleftarrow{\boldsymbol{m}}_i^{(j)}. \tag{5.5}$$

Final classifier. The final classifier first locates relevant memory chains via attention and then makes predictions based on the support of the states of those relevant entities. More specifically, in a question answering scenario, the prediction output $\hat{\boldsymbol{y}}$ to a question $\boldsymbol{q}$ given its context story is performed by incorporating the last states of all memory chains in the form of a weighted sum $\boldsymbol{u}$:

$$p^{(j)} = \text{softmax}\left(\left(\boldsymbol{k}^{(j)}\right)^{\top} \boldsymbol{W}_{att}\boldsymbol{q}\right) \tag{5.6}$$

$$\boldsymbol{u} = \sum_{j} p^{(j)} \boldsymbol{m}_{T}^{(j)} \tag{5.7}$$

where T denotes the total number of inputs in the context and $\boldsymbol{W}_{att}$ is a trainable weight matrix. We then transform $\boldsymbol{u}$ to get the final prediction:

$$\hat{\boldsymbol{y}} = \text{softmax}\left(\boldsymbol{R}\phi\left(\boldsymbol{H}\boldsymbol{u} + \boldsymbol{q}\right)\right) \tag{5.8}$$

where $\boldsymbol{R}$ and $\boldsymbol{H}$ are trainable weight matrices.

5.1.1 Motivating Example

To better understand why EntNets are a good fit for discourse tasks, we return to the motivating example shown in Table 5.1. For a detailed walk through of this motivating example, see Section 2.4.2.5.

Suppose a sentence-level EntNet, that is, taking a sentence as input at each step. We further assume that there are two memory chains responsible for keeping track of the states of the entities *Jeff* and *milk* respectively. Upon seeing the third sentence in Table 5.1, the EntNet is expected to update both memory chains regarding the states of their respective entities: *Jeff* is now carrying the *milk* and conversely the *milk* is carried by *Jeff*. The next sentence, *Jeff travelled to the bedroom*, should make the model realise that not only that *Jeff* is now in the bedroom, but the *milk* is also there as it is carried by *Jeff*. It is straightforward to see that the update to the *Jeff* memory chain is carried out because of the mention of the entity, activating the location term $\boldsymbol{x}_i \cdot \boldsymbol{k}^{(j)}$ in Equation (5.2).

#	Story
1	Jeff went to the kitchen.
2	Mary travelled to the hallway.
3	Jeff picked up the milk.
4	Jeff travelled to the bedroom.
5	Jeff left the milk.
6	Jeff went to the bathroom.

Question	Answer
Where is the milk now?	bedroom
Where was Jeff before the bedroom?	kitchen

Table 5.1 An example triple of story, question and answer, extracted from the bAbI dataset (Weston et al., 2016). Table reproduced from Table 1.1. Key pieces of evidence for the first question are underlined, with distractors marked with dashed underlining.

The update to the *milk* entity, on the other hand, is achieved through the content term $\boldsymbol{x}_i \cdot \overrightarrow{\boldsymbol{m}}_{i-1}^{(j)}$ in Equation (5.2) where the state of the entity stored in $\overrightarrow{\boldsymbol{m}}_{i-1}^{(j)}$ should be aware of the fact that the *milk* is carried by *Jeff*, resulting in a high score for the dot product, thereby also activating its update gate $\overrightarrow{g}_i^{(j)}$. Through a series of updates, at the end of the story, the j-th memory $\boldsymbol{m}_T^{(j)}$ stores the state of the entity it keeps track of and can therefore be used to perform discourse comprehension tasks.

This walk through of a motivating example shows how EntNets process a discourse in the question answering setting. Observe that the multi-sentence nature of discourse also implies that there are multiple entities, interconnected and interacting with one another. This is an important aspect of discourse, one that is difficult to capture by previous models, making EntNets the ideal candidate for a variety of discourse tasks.

5.2 Delayed Memory Update for Targeted Aspect-based Sentiment Analysis

In this section, we describe the first novel contribution in improving dynamic memory chains for discourse understanding. Targeted aspect-based sentiment analysis (TABSA)

LOC1 is different though, it's a bit of a run down area so don't expect too much when you get there, however the students take over most of it so you'll be safe enough

LOC1 (pronounced Gren-Itch) is a lovely town, but parts of it can be very expensive

LOC2, although it's a bit of a train journey to work, the lifestyle is very good,also *LOC1* has a bit of a reputation

Figure 5.3 Targeted aspect-based sentiment analysis examples. All examples taken from SENTIHOOD (Saeidi et al., 2016) with minor revisions.

is the task of extracting fine-grained opinion polarity with respect to a pre-defined set of aspects and their associated targets. Successful TABSA approaches provide an important tool for analysing user generated text, commonly taking the form of product reviews, in a range of domains, with the most notable ones being online shopping and hotel booking. Three examples of user generated reviews are given in Figure 5.3. Reviews often involve multiple entities, switching contexts between different targets and their associated aspects while relying heavily on discourse markers to connect these short segments. Such discourse markers (e.g., *and*, *also*, *but* and *however*) are crucial to the identification of sentiment flow (Pateria, 2016), providing critical evidence to the classification of sentiment polarity and signifying the importance of taking discourse elements into consideration. In recognition of this, researchers have made notable progress by incorporating a discourse parser (Zirn et al., 2011) or exploiting discourse relations (Wang et al., 2012, Mukherjee and Bhattacharyya, 2012). The frequent use of discourse elements in these efforts on review texts emphasises how TABSA is closely related to discourse understanding.

More recently, while neural networks have been shown to achieve impressive results for sentence-level sentiment analysis, TABSA remains a difficult task (Saeidi et al., 2016) In particular, this task requires a model to be able to keep track of multiple entities and aspects while capturing long-range dependencies as aspect and sentiment expressions may not be stated until much later than the associated target, rendering it a crucial element

in discourse understanding. In the examples in Figure 5.3, the SAFETY/PRICE/LIFESTYLE aspects of example 1/2/3 come much later than the mention of their associated targets: *LOC1*/*LOC1*/*LOC2*.

Motivated by recent advances in memory-augmented models for machine reading, we propose a novel architecture, utilising external "memory chains" with a delayed memory update mechanism to track entities. Evaluated on the task of targeted aspect-based sentiment analysis, the proposed model demonstrates substantial improvements over existing approaches, even superior to those incorporating external commonsense knowledge sources, setting a new state of the art in both sentiment classification and aspect detection.[1]

In this section, we first present the motivation for the proposed approach in Section 5.2.1. Next, we outline the task definition of TABSA in Section 5.2.2, followed by a detailed description of the proposed method: the delayed memory update mechanism, the model architecture and its training objective in Sections 5.2.3 — 5.2.4. We further contrast the proposed method with the vanilla ENTNET model in Section 5.2.5. Lastly, we describe the experimental setup and results, along with additional analysis, in Section 5.2.6, before concluding the section in Section 5.2.7.

5.2.1 Motivation

Targeted aspect-based sentiment analysis (TABSA) is the task of identifying fine-grained opinion polarity towards a specific aspect associated with a given target. The task requires classification of opinions on different entities across a range of different attributes, with the expectation that there will be no overt opinion expressed on a given entity for many attributes. This can be seen in Table 5.2, e.g., where opinions on the aspects SAFETY and PRICE are expressed for entity *LOC1* but not entity *LOC2*.[2]

The earliest work on (T)ABSA relied heavily on feature engineering (Wagner et al., 2014, Kiritchenko et al., 2014), but more recent work based on deep learning has used

[1] At the time publication. Subsequent studies have reported higher performance on this task in Liang et al. (2019) and Sun et al. (2019a). Brief summaries and discussions of their models are provided in Section 5.2.6.7.

[2] Note that in our dataset, all entity mentions have been pre-normalised to *LOCn*, where *n* is an index.

LOC1 is your best bet for security although expensive and *LOC2* is too far.

Target	Aspect	Sentiment
LOC1	SAFETY	positive
	PRICE	negative
	LOCATION	none
	GENERAL	none

Target	Aspect	Sentiment
LOC2	SAFETY	none
	PRICE	none
	LOCATION	negative
	GENERAL	none

Table 5.2 An example of the TABSA task. Expressions regarding aspect/sentiment are colour-coded with "none" indicating the pair of target and aspect not occurring in the sentence. Best viewed in colour.

models such as LSTMs to automatically learn aspect-specific word and sentence representations (Tang et al., 2016a).

Despite these successes, keeping track of multiple entity–aspect pairs remains a difficult task, even for an LSTM. As reported in Saeidi et al. (2016), a target-dependent BI-LSTM is ineffective, both in terms of aspect detection and sentiment classification, compared to a simple logistic regression model with n-gram features. Intuitively, we would expect that a model which better captures linguistic structure via the original word sequencing should perform better, which provides the motivation for this research.

More recently, successful works in (T)ABSA have explored the idea of leveraging external memory (Tang et al., 2016b, Chen et al., 2017). Their models are largely based on memory networks (Weston et al., 2015), originally developed for reasoning-focused machine reading comprehension tasks. In contrast to memory networks, where each input sentence/word occupies a memory slot and is then accessed via attention independently, recent advances in machine reading suggest that processing inputs sequentially is beneficial to the overall performance (Seo et al., 2017, Henaff et al., 2017).

Despite the slightly superior performance, bag-of-words models, ignoring word ordering information, are not plausible and make little sense from a linguistic perspective, especially given the semantic nature of the task (Socher et al., 2013). Given the complexity of the task, TABSA demands a model to be capable of keeping track of opinions on entity/aspect pairs while making use of word sequential information.

However, successful machine reading models may not be directly applicable to TABSA due to the key difference in the granularity of inputs between the two tasks: on the Children's Book Test corpus (CBT), for example, competitive models take as input a window of text, centred around candidate entities, with crucial information contained within that window (Hill et al., 2015, Henaff et al., 2017). In TABSA, given the fine-grained nature of the task, it is common practice for models to operate at the word- rather than the chunk/sentence-level. It is not uncommon to see examples like Table 5.2, where the sentence starts with *LOC1*, but the negative PRICE sentiment towards the entity is not expressed until much later. Moreover, phrases such as *best bet* and *although* play the role of triggers, indicating that succeeding tokens bear aspect/sentiment signal. This key difference necessitates the ability to model the delayed activation of memory updates.

In this section, we propose a novel model architecture for TABSA, augmented with multiple "memory chains", and equipped with a delayed memory update mechanism, to keep track of numerous entities independently and model the interaction between them.

5.2.2 Task Description

In TABSA, a sentence s typically consists of a sequence of words: $\{w_1, \ldots, w_i, \ldots, w_m\}$ where w_i denotes words interleaved with one or more targets (t), which we assume to be pre-identified as with *LOC1* and *LOC2* in Table 5.2. Following Saeidi et al. (2016), we frame the task as a 3-class classification problem: given a sentence s, a pre-identified set of target entities T and fixed set of aspects A, predict the sentiment polarity $y \in \{positive, negative, none\}$ over the full set of target–aspect pairs $\{(t,a) : t \in T, a \in A\}$. For example, the pair (*LOC1*,SAFETY) has a gold-standard polarity of *positive*, while (*LOC1*,TRANSIT-LOCATION) has the *none* polarity, suggesting it is not mentioned in the sentence.

5.2.3 Delayed Memory Update

We design a neural network architecture, capable of tracking and updating the states of entities at the right time with external memory, making it a natural fit for the task.

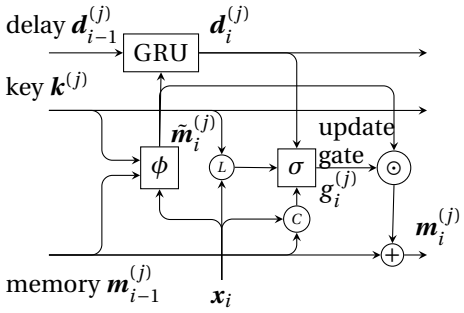

Figure 5.4 Illustration of our model with a single memory chain at time i. σ, ϕ and GRU represent Equations (5.9), (5.10) and (5.11), while circled nodes L, C, $\odot$ and + depict the location, content terms, Hadamard product, and addition, respectively.

Specifically, our model maintains a number of "memory chains" $\boldsymbol{m}^{(j)}$, one for each entity with the key $\boldsymbol{k}^{(j)}$ and dynamically updates the states ($\boldsymbol{m}^{(j)}$) of them as it progresses through the sentence with the help of the delay recurrence $\boldsymbol{d}^{(j)}$, taking previous activations into account. The input consists of a sequence of words w_i whose embeddings are denoted $\boldsymbol{x}_i$. An illustration of our model is provided in Figure 5.4.

The update of each memory chain is controlled by a gating mechanism, consisting of three components: the "content" term $\boldsymbol{x}_i \cdot \boldsymbol{m}_{i-1}^{(j)}$, the "location" term $\boldsymbol{x}_i \cdot \boldsymbol{k}^{(j)}$ and the "delay" term $\boldsymbol{v} \cdot \boldsymbol{d}_i^{(j)}$ where $\boldsymbol{d}_i^{(j)}$ carries knowledge regarding previous activations of the gate and $\boldsymbol{v}$ is a trainable parameter vector. All three terms may lead to the activation of $g_i^{(j)}$, but differ in how they turn the gate on. While the "location" term causes the gate to open for memory chains whose keys ($\boldsymbol{k}^{(j)}$) match the input, the "content" term triggers the activation when the content of the entities ($\boldsymbol{m}_{i-1}^{(j)}$) matches the input. The delay term models how and when the gate was turned on in the past with a GRU (Chung et al., 2014) and how past activations should influence the current one.

More formally, with arrows denoting processing direction, the new update gate (with the delayed memory update mechanism, replacing Equation (5.2)) is defined as:

$$\overrightarrow{g}_i^{(j)} = \sigma(\boldsymbol{x}_i \cdot \overrightarrow{\boldsymbol{m}}_{i-1}^{(j)} + \boldsymbol{x}_i \cdot \boldsymbol{k}^{(j)} + \overrightarrow{\boldsymbol{v}} \cdot \overrightarrow{\boldsymbol{d}}_i^{(j)}) \tag{5.9}$$

where $\overrightarrow{g}_i^{(j)}$ is the update gate value for the j-th memory at time i,[3] $\boldsymbol{k}^{(j)}$ is the embedding for the j-th entity (key), $\overrightarrow{\boldsymbol{m}}_{i-1}^{(j)}$ is the hidden memory representation responsible for keeping track of the state of the j-th entity (content), and σ is the sigmoid activation function. The delay recurrence $\overrightarrow{\boldsymbol{d}}_i^{(j)}$ is defined as:

$$\overrightarrow{\tilde{\boldsymbol{m}}}_i^{(j)} = \phi(\overrightarrow{\boldsymbol{U}}\overrightarrow{\boldsymbol{m}}_{i-1}^{(j)} + \overrightarrow{\boldsymbol{V}}\boldsymbol{k}^{(j)} + \overrightarrow{\boldsymbol{W}}\boldsymbol{x}_i) \tag{5.10}$$

$$\overrightarrow{\boldsymbol{d}}_i^{(j)} = \overrightarrow{\text{GRU}}(\overrightarrow{\tilde{\boldsymbol{m}}}_i^{(j)}, \overrightarrow{\boldsymbol{d}}_{i-1}^{(j)}) \tag{5.11}$$

where $\overrightarrow{\tilde{\boldsymbol{m}}}_i^{(j)}$ is the new candidate memory vector to be incorporated into the existing memory $\overrightarrow{\boldsymbol{m}}_{i-1}^{(j)}$ to form the new memory $\overrightarrow{\boldsymbol{m}}_i^{(j)}$, ϕ is the parametric ReLU activation function (He et al., 2015), and $\overrightarrow{\boldsymbol{U}}$, $\overrightarrow{\boldsymbol{V}}$ and $\overrightarrow{\boldsymbol{W}}$ are trainable weight matrices.

Once the update gate value has been computed, the j-th memory is then updated according to the intensity of $\overrightarrow{g}_i^{(j)}$:

$$\overrightarrow{\breve{\boldsymbol{m}}}_i^{(j)} = \overrightarrow{\boldsymbol{m}}_{i-1}^{(j)} + \overrightarrow{g}_i^{(j)} \odot \overrightarrow{\tilde{\boldsymbol{m}}}_i^{(j)} \tag{5.12}$$

where $\odot$ is the Hadamard product, and $\overrightarrow{\breve{\boldsymbol{m}}}_i^{(j)}$ is the unnormalised memory representation for the j-th entity. $\overrightarrow{\breve{\boldsymbol{m}}}_i^{(j)}$ is then normalised using Equation (5.4).

Essentially, gate $\overrightarrow{g}_i^{(j)}$ determines how much the j-th memory should be updated, factoring in three elements: (1) how similar the current input $\boldsymbol{x}_i$ is to the entity being tracked ($\boldsymbol{k}^{(j)}$); (2) how related the current input $\boldsymbol{x}_i$ is to the state of the j-th entity ($\overrightarrow{\boldsymbol{m}}_{i-1}^{(j)}$); and (3) how past activation should influence the current one. Update of the memory of an entity is only triggered when the gate is activated.

[3] While $\overrightarrow{g}_i^{(j)}$ could instead be a vector for finer-grained control, following Henaff et al. (2017), we use a scalar for simplicity.

5.2.4 Final Classifier

Our model predicts the sentiment polarity $\hat{\boldsymbol{y}}$ to the given target $\boldsymbol{t}$ and aspect $\boldsymbol{a}$ embeddings by incorporating the states of all tracked entities in the form of a weighted sum $\boldsymbol{u}$:

$$p^{(j)} = \text{softmax}\left((\boldsymbol{k}^{(j)})^{\top} \boldsymbol{W}^{att} \begin{bmatrix} \boldsymbol{t} \\ \boldsymbol{a} \end{bmatrix}\right) \tag{5.13}$$

$$\boldsymbol{u} = \sum_{j} p^{(j)} \boldsymbol{m}_T^{(j)} \tag{5.14}$$

where [] denotes concatenation, T is sentence length, and $\boldsymbol{W}_{att}$ is a trainable weight matrix. Here, the values of both $\boldsymbol{t}$ and $\boldsymbol{a}$ take the embedding values of their corresponding words (i.e. $\boldsymbol{t}$ and $\boldsymbol{a}$ are drawn from the same embedding matrix as are the input words $\boldsymbol{x}_i$). In the case of multi-word aspect expressions (e.g. TRANSIT-LOCATION), we take the mean of the embeddings of the constituent words. We then transform $\boldsymbol{u}$ to get:

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{R}\phi(\boldsymbol{H}\boldsymbol{u} + \boldsymbol{a})) \tag{5.15}$$

where $\hat{\boldsymbol{y}}$ is the predicted distribution over the possible polarity candidates {*positive*, *negative*, *none*}. Training is carried out based on the categorical cross entropy loss:

$$\mathscr{L} = \text{CrossEntropy}(\boldsymbol{y}, \hat{\boldsymbol{y}}). \tag{5.16}$$

5.2.5 Comparison with ENTNET

While our model is largely inspired by Recurrent Entity Networks (ENTNETs: Henaff et al. (2017)), it differs in three main respects. First, we explicitly model the delay of activation of the update gates $g^{(j)}$ with the GRU in Equations (5.9) and (5.11) as opposed to making $\boldsymbol{m}_i^{(j)}$ implicitly assume the same responsibility in ENTNETs. Admittedly, for ENTNETs on bAbI and CBT, given the coarse-grained nature and the difference in the granularity of inputs (sentences vs. words), the demand for modelling the delay in memory update is less obvious. With this delayed gate activation mechanism, we essentially decouple the

duty of capturing transitions of activations between steps from the task of entity state tracking. That is, $\boldsymbol{m}_i^{(j)}$ is now dedicated to keeping track of the state of the j-th entity only and released from the burden of monitoring the activation of the update gate. Second, tailored to the task of TABSA, we incorporate not only the target $\boldsymbol{t}$ but also the aspect $\boldsymbol{a}$ when trying to determine the attention in the softmax function. Third, the proposed model is bi-directional.

5.2.6 Experiments on Sentihood

5.2.6.1 Dataset

To test the effectiveness of our model, we use SENTIHOOD, a dataset constructed by Saeidi et al. (2016) for the purpose of detecting aspects and identifying the sentiment for each target–aspect pair, consisting of 5,215 sentences, 3,862 of which contain a single target, and the remainder multiple targets. Each sentence is annotated with a list of tuples $\{(t, a, y)\}$ with each identifying the sentiment polarity y towards a specific aspect a of a given target t in s. Ultimately, given a sentence s, we are interested in both detecting the mention of an aspect a for target t (a label other than *none*), and also identifying the specific sentiment y w.r.t. the target–aspect pair. A detailed description of the task is presented in Section 5.2.2.

5.2.6.2 Model Configuration

We initialise our model with GLOVE[4] (300-D, trained on 42B tokens, 1.9M vocab, not updated during training: Pennington et al. (2014)) and pre-process the corpus with tokenisation using NLTK (Bird et al., 2009) and case folding. Training is carried out over 800 epochs with the FTRL optimiser (McMahan et al., 2013) and a batch size of 128 and learning rate of 0.05. We use the following hyper-parameters for weight matrices in both directions: $\boldsymbol{R} \in \mathbb{R}^{300\times 3}$, $\boldsymbol{H}$, $\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{W}$ are all matrices of size $\mathbb{R}^{300\times 300}$, $\boldsymbol{v} \in \mathbb{R}^{300}$, and hidden size of the GRU in Equation (5.11) is 300. Dropout is applied to the output of ϕ in the

[4]http://nlp.stanford.edu/data/glove.42B.300d.zip

final classifier (Equation (5.15)) with a rate of 0.2. Moreover, we employ the technique introduced by Gal and Ghahramani (2016) where the same dropout mask is applied to the input $\boldsymbol{w}_i$ at every step with a rate of 0.2. Lastly, to curb overfitting, we regularise the last layer (Equation (5.15)) with an L_2 penalty on its weights: $\lambda\|\boldsymbol{R}\|$ where $\lambda = 0.001$.

We empirically set the number of memory chains to 6, with the keys of two of them set to the same embeddings as the target words *LOC1* and *LOC2*, respectively, and the other 4 chains with free key embeddings which are updated during training, and therefore free to capture any entities.[5] A further sensitivity study is conducted and presented in Section 5.2.6.6.

Consistent with Saeidi et al. (2016), we tackle the class imbalance problem (*none* $\gg$ *positive* + *negative*) by sampling the same number of training instances within a batch randomly from each class.

5.2.6.3 Evaluation

We benchmark against baseline systems presented in the works of Saeidi et al. (2016) and Ma et al. (2018): (1) LR: a logistic regression classifier with n-gram and POS tag features; (2) LSTM-FINAL: a BI-LSTM taking the final states as representations; (3) LSTM-LOC: a BI-LSTM taking the states at the location where target t is mentioned as representations; (4) LSTM+TA+SA: a BI-LSTM equipped with complex target and sentence-level attention mechanisms; and (5) SENTICLSTM: an improved version of (4) incorporating the SENTICNET external knowledge base (Cambria et al., 2016). We additionally implement a bi-directional ENTNET without the proposed delay mechanism and train it with the same hyper-parameter settings and GLOVE embeddings as our model (Henaff et al., 2017).

In terms of evaluation, we adopt the standard 70/10/20 train/validation/test split, and report the test performance corresponding to the model with the best validation score.

[5] In line with the findings of Henaff et al. (2017) that tying key vectors damages model performance, we observed similar performance deterioration when using tied keys only. While we also experimented with various configurations (all tied vs. all free), this hybrid setup results in the best performance on the validation set.

Model	Aspect			Sentiment	
	Acc.	F_1	AUC	Acc.	AUC
LR (Saeidi et al., 2016)	—	39.3	92.4	87.5	90.5
LSTM-FINAL (Saeidi et al., 2016)	—	68.9	89.8	82.0	85.4
LSTM-LOC (Saeidi et al., 2016)	—	69.3	89.7	81.9	83.9
LSTM+TA+SA (Ma et al., 2018)	66.4	76.7	—	86.8	—
SENTICLSTM (Ma et al., 2018)	67.4	78.2	—	89.3	—
ENTNET†	66.3	69.8	89.5	87.6	89.7
Our model†	**73.5**	**78.5**	**94.4**	**91.0**	**94.8**

Table 5.3 Performance on SENTIHOOD. We take the results reported in Saeidi et al. (2016) and Ma et al. (2018), respectively; **Bold** = best performance; "—" = not reported; † = average performance over 5 runs.

Following Saeidi et al. (2016), we consider the top 4 aspects only (GENERAL, PRICE, TRANSIT-LOCATION, and SAFETY) and employ the following evaluation metrics: macro-average F_1 and AUC for aspect detection ignoring the *none* class, and accuracy and macro-average AUC for sentiment classification. Following Ma et al. (2018), we also report strict accuracy for aspect detection, as the fraction of sentences where all aspects are detected correctly.

5.2.6.4 Results

The experimental results are presented in Table 5.6.

State-of-the-art results.[6] Our model achieves state-of-the-art results for both aspect detection and sentiment classification. While ENTNET only outperforms existing baselines slightly, our model further improves the performance with the proposed delayed activation mechanism in both accuracy and AUC, setting a new state-of-the-art. It is impressive that the proposed model, equipped only with the domain-independent general-purpose GLOVE embeddings, outperforms SENTICLSTM, an approach heavily reliant on external knowledge bases and domain-specific embeddings.

[6]At the time of publication. Subsequent studies have reported higher performance on this task in Liang et al. (2019) and Sun et al. (2019a). Brief summaries and discussions of their models are provided in Section 5.2.6.7.

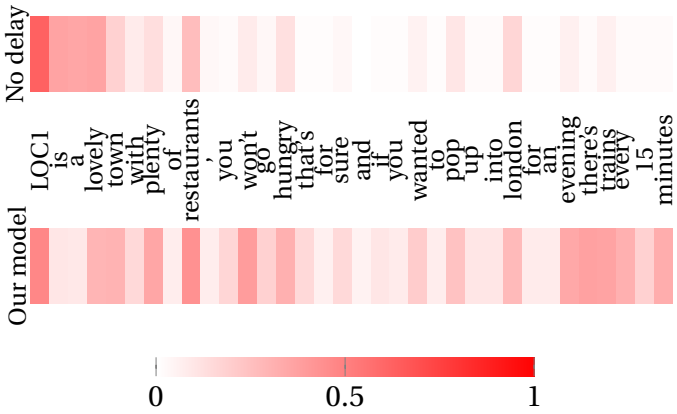

Figure 5.5 Example of the gate value g_i averaged across memory chains, forward and backward, in ENTNET without delay vs. our model (with delay).

ENTNET vs. our model. We see consistent performance gains for our model in both aspect detection and sentiment classification, compared to ENTNET, especially for aspect detection, underlining the benefits of delayed update gate activation.

5.2.6.5 Discussion

To better understand what the model has learned, we visualise the average gate value g_i in Figure 5.5, where colour intensity indicates how much memory is updated. Observe that, while updated less by the mention of *LOC1*, our model carries out memory updates upon seeing *lovely town* and *plenty of restaurants*, key phrases associated with aspects such as GENERAL and DINNING. Perhaps even more importantly, despite the distance between *LOC1* and the final portion of the sentence, our model recognises the relevance to TRANSIT-LOCATION and keeps the update gates open to track this particular aspect, as opposed to ENTNET where the last phase is overlooked. The ultimate prediction for the TRANSIT-LOCATION aspect of *LOC1* is correct with our model (*positive*), but not detected by ENTNET (*none*), resulting in a false negative. More interestingly, with ENTNET, once distant from a target, it can be frequently observed that the activation rate of g_i tends to drop, a tendency not so apparent with our model.

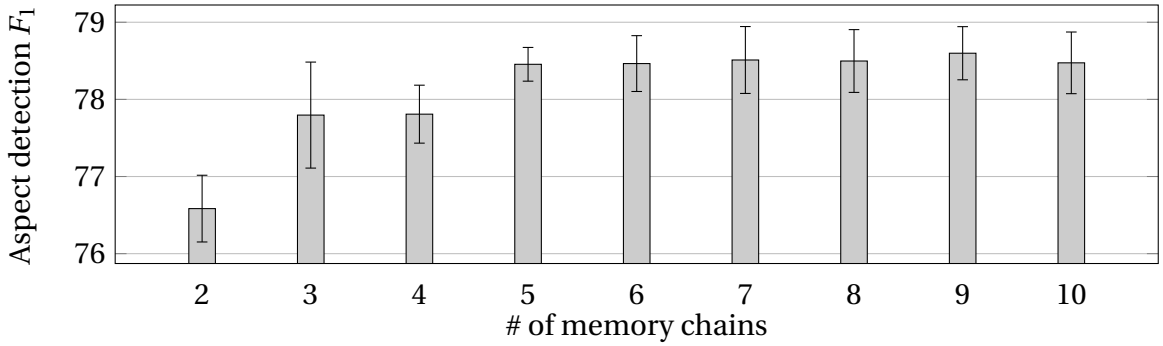

Figure 5.6 Sensitivity study of model performance to # of memory chains n. Note that we report average performance over 5 runs with standard deviation.

5.2.6.6 Sensitivity Study

In Figure 5.6, we further study the sensitivity of model performance to the number of memory chains n (2 of which are constrained to track *LOC1* and *LOC2*, the rest are unconstrained chains). Observe that, when $n < 5$, the model suffers from insufficient capacity (not enough memory chains) to capture the various aspects required by the task, with aspect detection F_1 remaining below 78. In particular, when $n = 2$ (no unconstrained chains), model performance drops substantially to a F_1 of 76.6 ± 0.4. Once $n \geq 5$, aspect detection F_1 increases to around 78, and is quite stable even with as many as $n = 10$ chains.

5.2.6.7 Subsequent Studies

At the time of publication, the results in Table 5.3 represented the state of the art on the TABSA task. Subsequent studies have reported more competitive results, as presented in Table 5.4.

Observing that different targets and aspects may be represented by the same or similar embeddings regardless of their contexts, Liang et al. (2019) propose to use a sparse coefficient vector to refine such embeddings, making them context-dependent and therefore more distinguishable. Their work builds on our model and achieves further improvements on (T)ABSA, demonstrating the potentials of the proposed delayed memory update mechanism.

Model	Aspect			Sentiment	
	Acc.	F_1	AUC	Acc.	AUC
EntNet†	66.3	69.8	89.5	87.6	89.7
Our model†	73.5	78.5	94.4	91.0	94.8
RE+our model (Liang et al., 2019)‡	76.4	81.0	96.8	92.8	96.2
BERT-pair-QA-M (Sun et al., 2019a)	79.4	86.4	97.0	**93.6**	96.4
BERT-pair-NLI-M (Sun et al., 2019a)	78.3	87.0	**97.5**	92.1	96.5
BERT-pair-QA-B (Sun et al., 2019a)	79.2	**87.9**	97.1	93.3	**97.0**
BERT-pair-NLI-B (Sun et al., 2019a)	**79.8**	87.5	96.6	92.8	96.9

Table 5.4 Performance of subsequent studies on SentiHood. We take the results reported in Liang et al. (2019) and Sun et al. (2019a), respectively; **Bold** = best performance; †/‡ = average performance over 5/10 runs.

Based on the highly popular pre-trained BERT language encoder (Devlin et al., 2019) and further fine-tuning, Sun et al. (2019a) frame the task of TABSA as one of either question answering (QA) or natural language inference (NLI) by generating a pair of sentences: the original input context and the auxiliary sentence describing the target and aspect. This framework can be further divided into two different modes: (1) multi-class classification (M), and (2) binary classification (B), depending on whether the sentiment polarity for the target/aspect pair being considered is included the auxiliary sentence (included: B; otherwise: M). Despite the minor performance differences between various modes, the impressive performance gains over previous state of the art approaches may largely be attributed to: (1) BERT, which is pre-trained on large-scale corpora, allowing the encoder to capture a broad range language regularities; and (2) the appropriate use of a pair of sentences as input, as opposed to a single-sentence input (the original input sentence only) which results in inferior predictive accuracy (Sun et al., 2019a).

5.2.7 Summary

In this section, we have demonstrated the power of the proposed delayed memory update mechanism, improving the vanilla EntNet model and enabling its memory component to better model the interactions between entity–aspect pairs and their sentiment.

In the next section, we will see a training technique for memory chains, incorporating semantic supervision generated by existing linguistic tools automatically, thereby allowing memory chains to better store knowledge presented in the context stably over long timescales.

5.3 Narrative Modelling with Memory Chains and Semantic Supervision

We now turn to another discourse task, story comprehension, and present the second novel contribution in improving dynamic memory chains for discourse understanding. Story comprehension requires a deep semantic understanding of the narrative, making it a challenging task. Inspired by previous studies on ROC Story Cloze Test, in this section, we adopt ENTNET (Henaff et al., 2017) with a novel training method, tracking various semantic aspects with external neural memory chains while encouraging each to focus on a particular semantic aspect. Not only is the proposed method free of feature engineering, making it easy to deploy to unseen data without preprocessing, it also imposes more modest data requirements during training. Evaluated on the task of story ending prediction, our model demonstrates superior performance to a collection of competitive baselines, setting a new state of the art while under more modest data requirements.[7].

In this section, we first present the motivation for the proposed approach in Section 5.3.1. Next, we outline the task definition of Story Cloze Test in Section 5.3.2, followed by a detailed description of the proposed method: semantically motivated memory chains, its architecture and the training objective in Sections 5.3.3 – 5.3.7. In Section 5.3.8, we further contrast the proposed method with the delayed memory update mechanism introduced in the previous section (Section 5.2). Lastly, we present the experimental setup and results, along with additional analysis, in Section 5.3.9.

[7]At the time of publication. Subsequent studies have reported higher performance on this task in Radford et al. (2019) and Sun et al. (2019b). A brief summary and discussion of each model is provided in Section 5.3.9.7.

Context: Sam loved his old belt. He matched it with everything. Unfortunately he gained too much weight. It became too small.
Coherent Ending: Sam went on a diet.
Incoherent Ending: Sam was happy.

Context: Rick fell while hiking in the woods. He was terrified! He thought he had fallen into a patch of poison ivy. Then he used his nature guide to identify the plant.
Coherent Ending: He was relieved to find out he was wrong.
Incoherent Ending: Rick was soaking wet from falling in the pond.

Context: Tim and Deb have been dating for almost a year. They met at a party they attended last Christmas. Tim picked Deb up for a date tonight and took her to a fancy dinner. Just after dessert he proposed marriage.
Coherent Ending: Deb said yes to Tim's marriage proposal.
Incoherent Ending: Deb told Tim she was only interested in woman.

Context: Dave walked into the grocery store. He was going there to buy his favourite energy drink. He only had enough money to buy one can. He reached the aisle and what he saw made him smile.
Coherent Ending: They were on sale.
Incoherent Ending: Dave bought an entire case.

Figure 5.7 ROC Story Cloze Test examples.

5.3.1 Motivation

Story narrative comprehension has been a long-standing challenge in artificial intelligence (Winograd, 1972, Turner, 1994, Schubert and Hwang, 2000). The difficulties of this task arise from the necessity for understanding not only narratives, but also commonsense and normative social behaviour (Charniak, 1972). Of particular interest to this section is the work by Mostafazadeh et al. (2016) on understanding commonsense stories in the form of a Story Cloze Test: given a short story, we must predict the most coherent sentential ending from two options (e.g. see Figure 5.7; more in Section 2.6.1.3).

Early work has focused on various aspects of story understanding and many attempts have been made to solve this problem, based on either linear classifiers with handcrafted

features (Schwartz et al., 2017, Chaturvedi et al., 2017), or representation learning via deep learning models (Mihaylov and Frank, 2017, Bugert et al., 2017, Mostafazadeh et al., 2017). While designing such features for the former group is time consuming and requires domain-specific knowledge, BI-LSTMs used in the latter group struggle to capture long-range contextual dependencies, a key element in narrative comprehension given the length of such stories (Lai et al., 2015, Linzen et al., 2016). This is further evidenced by the inferior performance of BI-LSTMs to simple, feature-based linear classifiers in the LSDSem 2017 shared task (Mostafazadeh et al., 2017). Another widely used component of competitive systems is language model-based features, which require training on large, domain-specific corpora.

The state-of-the-art approach of Chaturvedi et al. (2017) at the time this research was carried out was based on understanding the context from three perspectives: (1) event sequence, (2) sentiment trajectory, and (3) topic consistency. Chaturvedi et al. (2017) adopt external tools to recognise relevant aspect-triggering words, and manually design features to incorporate them into the classifier. While identifying triggers has been made easy by the use of various linguistic resources, crafting such features is time consuming and requires domain-specific knowledge along with repeated experimentation.

Inspired by the argument for tracking the dynamics of events, sentiment and topic, we propose to address the issues identified above with multiple external memory chains, each responsible for a single aspect. Building on ENTNETS, we introduce a novel multi-task learning objective, encouraging each chain to focus on a particular aspect. Specifically, in addition to the supervision at the output step, we also supervise the memory update gate activations with linguistically motivated signals, guiding each chain to concentrate on a certain aspect only, thereby diversifying the focus of each. While still making use of external linguistic resources, we do not extract or design features from them but rather utilise such tools to generate labels. The generated labels are then used to guide training so that each chain focuses on tracking a particular aspect. At test time, our model is free of feature engineering such that, once trained, it can be easily deployed to unseen data without preprocessing. Moreover, our approach also differs in the lack of a ROC Stories

language model component, eliminating the need for large, domain-specific training corpora.

5.3.2 Task Description

In the story cloze test, given a story of length T, consisting of a sequence of context words $\mathbf{w} = \{w_1, w_2, \ldots, w_T\}$, we are interested in predicting the coherent ending to the story out of two ending options o_1 and o_2. Following previous studies (Schwartz et al., 2017), we frame this problem as a binary classification task. Assuming o_1 and o_2 are the logical and inconsistent endings respectively with their associated labels being y_1 and y_2, we assign $y_1 = 1$ and $y_2 = 0$. At test time, given a pair of possible endings, the system returns the one with the higher score as the prediction.

5.3.3 Semantically Motivated Memory Chains

Building on EntNets (see Section 5.1 for a brief overview and Section 2.4.2 for more details), we supervise the activation of each update gate (Equation (5.2)) with binary labels in order to encourage each chain to focus on a particular aspect. Intuitively, for the sentiment chain, a sentiment-bearing word (e.g. *amazing*) receives a label of 1, allowing the model to open the gate and therefore change the internal state relevant to this aspect, while a neutral one (e.g. *school*) should not trigger the activation with an assigned label of 0. To achieve this, we use three memory chains to independently track each of: (1) event sequence, (2) sentiment trajectory, and (3) topical consistency. To supervise the memory update gates of these chains, we design three sequences of binary labels: $\mathbf{l}^{(j)} = \{l_1^{(j)}, l_2^{(j)}, \ldots, l_T^{(j)}\}$ for $j \in [1,3]$ representing event, sentiment, and topic, respectively, and $l_i^{(j)} \in \{0,1\}$. The label at time i for the j-th aspect is only assigned a value of 1 if the word is a trigger for that particular aspect: $l_i^{(j)} = 1$; otherwise: $l_i^{(j)} = 0$. During training, we utilise such sequences of binary labels to supervise the memory update gate activations of each chain. Specifically, each chain is encouraged to open its gate only when it sees a trigger bearing information semantically sensitive to that particular aspect. Note that

	Ricky	fell	while	hiking	in	the	woods
$\mathbf{l}^{Event}$	0	1	0	1	0	0	1
$\mathbf{l}^{Sentiment}$	0	1	0	0	0	0	0
$\mathbf{l}^{Topic}$	1	1	0	1	0	0	1

Table 5.5 An example of the linguistically motivated memory chain supervision binary labels.

at test time, we do not apply such supervision. This effectively becomes a binary tagging task for each memory chain independently and results in individual memory chains being sensitive to only a specific set of triggers bearing one of the three types of signals.

While largely inspired by Chaturvedi et al. (2017), our approach differs in how we extract and use such features. Also note that, while still making use of external linguistic resources to detect trigger words, our approach lets the memory chains decide how such words should influence the final prediction, as opposed to the handcrafted conditional probability features of Chaturvedi et al. (2017).

To identify the trigger words, we use external linguistic tools, and mark trigger words for each aspect with a label of 1. An example is presented in Table 5.5, noting that the same word can act as triggers for multiple aspects. An illustration of the architecture of an ENTNET with semantic supervision based on the same example as above is given in Figure 5.8 where recurrent units whose gates are encouraged to open are highlighted with respective colours reflecting their semantic class.

Event sequence. We parse each sentence into its FrameNet representation with SEMAFOR (Das et al., 2010), and identify each frame target (word or phrase tokens evoking a frame).

Sentiment trajectory. Following Chaturvedi et al. (2017), we utilise a pre-compiled list of sentiment words (Liu et al., 2005). To take negation into account, we parse each sentence with the STANFORD CORE NLP dependency parser (Manning et al., 2014, Chen and Manning, 2014) and include negation words as trigger words.

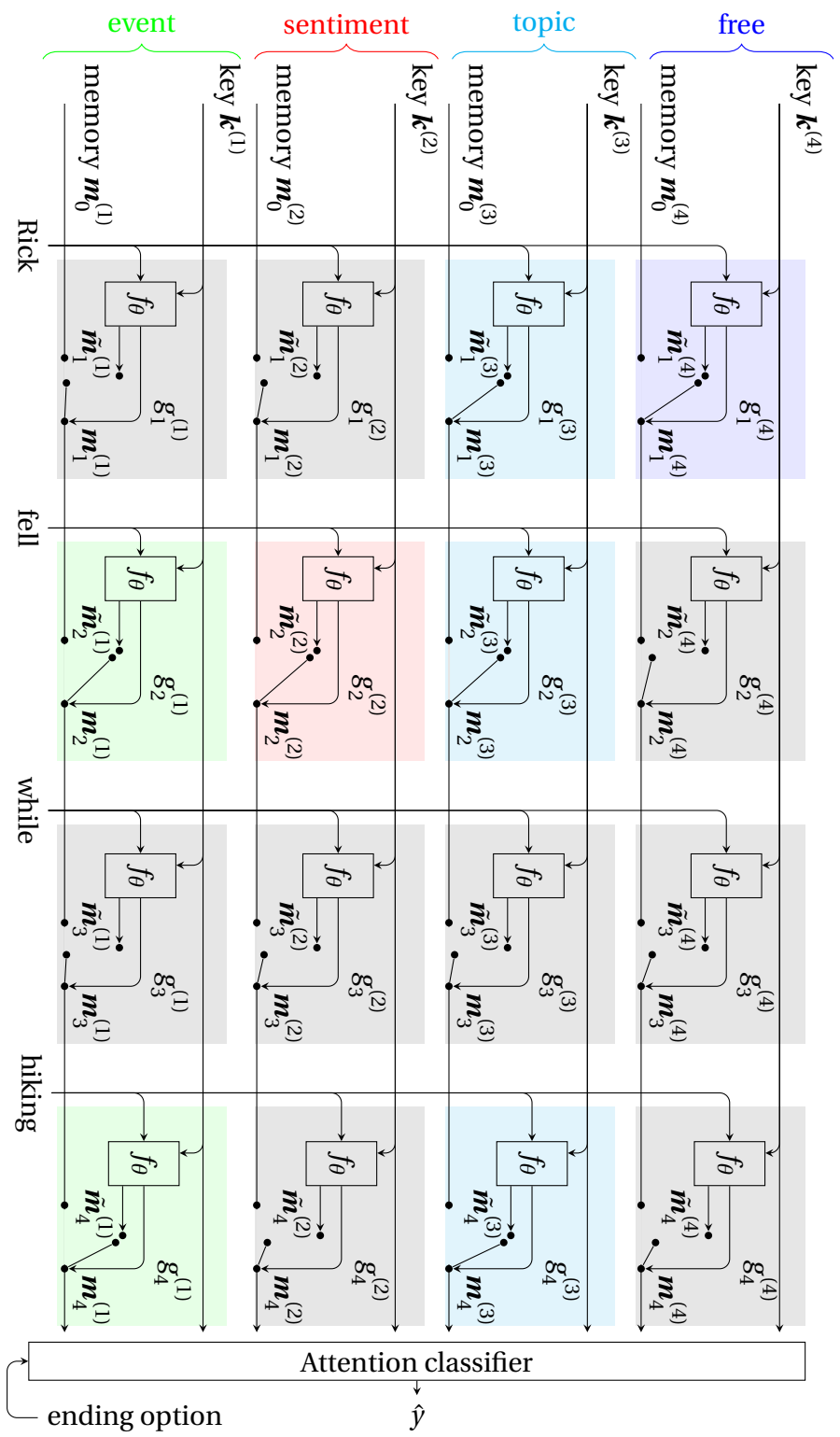

Figure 5.8 An example of EntNet with semantic supervision. EntNet cells activated due to highly relevant input are highlighted in colours matching those used for the label of each memory chain. Best viewed in colour.

Topical consistency. We process each sentence with the STANFORD CORE NLP POS tagger and identify nouns and verbs, following Chaturvedi et al. (2017).

5.3.4 Sentence-level Representation

To take neighbouring contexts into account, we process context sentences and ending options at the word level with a bi-directional gated recurrent unit ("BI-GRU": Chung et al. (2014)): $\overrightarrow{\boldsymbol{h}}_i = \overrightarrow{\text{GRU}}\left(\boldsymbol{x}_i, \overrightarrow{\boldsymbol{h}}_{i-1}\right)$ where $\boldsymbol{x}_i$ is the embedding of the i-th word in the story or ending option. The backward hidden representation $\overleftarrow{\boldsymbol{h}}_i$ is computed analogously but with a different set of $\overleftarrow{\text{GRU}}$ parameters. We simply take the sum $\boldsymbol{h}_i = \overrightarrow{\boldsymbol{h}}_i + \overleftarrow{\boldsymbol{h}}_i$ as the representation at time i. An ending option, denoted $\boldsymbol{o}$, is represented by taking the sum of the final states of the same BI-GRU over the ending option word sequence.

5.3.5 Bi-directionality

In addition to making the memory representations bi-directional as in Equation (5.5), we further define $g_i^{(j)}$ to be the average of the gate values of both directions at time i for the j-th chain:

$$g_i^{(j)} = \frac{\overrightarrow{g}_i^{(j)} + \overleftarrow{g}_i^{(j)}}{2} \tag{5.17}$$

which will be used for semantic supervision.

5.3.6 Final Classifier

The final prediction $\hat{y}$ to an ending option given its context story is performed by incorporating the last states of all memory chains in the form of a weighted sum $\boldsymbol{u}$:

$$p^{(j)} = \text{softmax}\left(\left(\boldsymbol{k}^{(j)}\right)^\top \boldsymbol{W}_{att}\boldsymbol{o}\right) \tag{5.18}$$

$$\boldsymbol{u} = \sum_j p^{(j)} \boldsymbol{m}_T^{(j)} \tag{5.19}$$

where T denotes the total number of words in a story and $\boldsymbol{W}_{att}$ is a trainable weight matrix. We then transform $\boldsymbol{u}$ to get the final prediction:

$$\hat{y} = \sigma\left(\boldsymbol{R}\phi\left(\boldsymbol{H}\boldsymbol{u} + \boldsymbol{o}\right)\right) \tag{5.20}$$

where $\boldsymbol{R}$ and $\boldsymbol{H}$ are trainable weight matrices. Note that $\hat{y}$ is a scalar representing the probability of the option $\boldsymbol{o}$ being the coherent ending to the story.

5.3.7 Training Loss

In addition to the categorical cross entropy loss of the final prediction of right/wrong endings, we also take into account the memory update gate supervision of each chain by adding the second term. More formally, the model is trained to minimise the loss:

$$\mathscr{L} = \text{CrossEntropy}\left(y, \hat{y}\right) + \alpha \sum_{i,j} \text{CrossEntropy}\left(l_i^{(j)}, g_i^{(j)}\right) \tag{5.21}$$

where $\hat{y}$ and $g_i^{(j)}$ are defined in Equations (5.20) and (5.17) respectively, y is the gold label for the current ending option o, and $l_i^{(j)}$ is the semantic supervision binary label at time i for the j-th memory chain. In our experiments, we empirically set α to 0.5 based on development data.

5.3.8 Why Not Delayed Memory Update?

A natural question to ask is why not try incorporating the delayed memory update mechanism into the method introduced in this section? While the focus of Section 5.2 was on better tracking the dynamics of the sentiment of multiple targets and aspects by modelling the delay in memory updates, the goal of this section is to model story coherence and find a sensible ending option to a given story. The fine-grained nature of TABSA requires careful modelling of the delay in memory updates at the word level, particularly when a target or aspect is mentioned. In contrast, we now focus on a single goal: story coherence, as opposed to multiple entities, and model it by looking at the same story from a variety of

perspectives. We are more interested in how occurrences of words influence the narrative of a story rather than to which target or aspect the current sentiment is attached. This renders the delayed memory update mechanism ineffective to the task of Story Cloze Test. This is further evidenced by preliminary experiments with the two methods combined, showing slightly inferior results and suggesting the delayed memory update mechanism is detrimental to the task.

5.3.9 Experiments on ROC Story Cloze Test

5.3.9.1 Dataset

To test the effectiveness of our model, we employ the Story Cloze Test dataset of Mostafazadeh et al. (2016) (see Section 2.6.1.3). The development and test set each consist of 1,871 4-sentence stories, each with a pair of ending options. Consistent with previous studies, we split the development set into a training and validation set (for early stopping), resulting in 1,683 and 188 in each set, respectively. Note that while most current approaches make use of the much larger training set, comprised of 100K 5-sentence ROC stories (with coherent endings only, also released as part of the dataset) to train a language model, we make no use of this data.

5.3.9.2 Model Configuration

We initialise our model with WORD2VEC embeddings (300-D, pre-trained on 100B Google News articles, not updated during training: Mikolov et al. (2013a,b)). In addition to the three supervised chains, we also add a "free" chain, unconstrained to any semantic aspect.

Training is carried out over 200 epochs with the FTRL optimiser (McMahan et al., 2013) and a batch size of 128 and learning rate of 0.1. We use the following hyper-parameters for weight matrices in both directions: $\boldsymbol{R} \in \mathbb{R}^{300\times 1}$, $\boldsymbol{H}$, $\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{W}$ are all matrices of size $\mathbb{R}^{300\times 300}$, and hidden size of the BI-GRU is 300. Dropout is applied to the output of ϕ in the final classifier (Equation (5.20)) with a rate of 0.2. Moreover, we employ the technique introduced by Gal and Ghahramani (2016) where the same dropout mask is applied at

every step to the input $\boldsymbol{x}_i$ to the BI-GRU and the input $\boldsymbol{h}_i$ to the memory chains with rates of 0.5 and 0.2 respectively. Lastly, to curb overfitting, we regularise the last layer (Equation (5.20)) with an L_2 penalty on its weights: $\lambda\|\boldsymbol{R}\|$ where $\lambda = 0.001$.

5.3.9.3 Evaluation

Following previous studies, we evaluate the performance in terms of coherent ending prediction accuracy: $\frac{\#\text{correct}}{\#\text{total}}$. Here, we report the average accuracy over 5 runs with different random seeds. Models are selected based on their performance on the validation set.

We benchmark against a collection of strong baselines, including the top-3 systems of the 2017 LSDSem workshop shared task (Mostafazadeh et al., 2017): MSAP (Schwartz et al., 2017), HCM (Chaturvedi et al., 2017)[8], and TBMIHAYLOV (Mihaylov and Frank, 2017). The first two primarily rely on a simple logistic regression classifier, both taking linguistic and probability features from a ROC Stories domain-specific neural language model. TBMIHAYLOV is LSTM-based; we also include DSSM (Mostafazadeh et al., 2016). We additionally implement a bi-directional ENTNET (Henaff et al., 2017) with the same hyper-parameter settings as our model and no semantic supervision.[9]

5.3.9.4 Results and Discussion

The experimental results are shown in Table 5.6.

State-of-the-art results.[10] Our model outperforms a collection of strong baselines, setting a new state of the art. Note that this is achieved without any external linguistic resources at test time or domain-specific language model-based probability features, highlighting the effectiveness of the proposed model.

[8] We take the performance reported in a subsequent paper, 3.2% better than the original submission to the shared task.

[9] ENTNET is also subject to the same model selection criterion described above.

[10] At the time of publication. Subsequent studies have reported higher performance on this task in Radford et al. (2019) and Sun et al. (2019b). A brief summary and discussion of each model is provided in Section 5.3.9.7.

Model	Acc.	SD
DSSM	58.5	—
TBMIHAYLOV	72.8	—
MSAP	75.2	—
HCM	77.6	—
ENTNET †	77.6	±0.5
Our model†	**78.5**	±0.5

Table 5.6 Performance on the Story Cloze test set. **Bold** = best performance, † = ave. accuracy over 5 runs, SD = standard deviation, "—" = not reported.

ENTNET vs. our model. We see clear improvements of the proposed method over ENTNET, an absolute gain of 0.9% in accuracy. This validates the hypothesis that encouraging each memory chain to focus on a unique, well-defined aspect is beneficial.

5.3.9.5 Discussion

To better understand the benefits of the proposed method, we visualise the learned word representations (output representation of the BI-GRU, $\boldsymbol{h}_i$) and keys between ENTNET and our model in Figure 5.9. With ENTNET, while sentiment words form a loose cluster, words bearing event and topic signal are placed in close proximity and largely indistinguishable. With our model, on the other hand, the degree of separation is much clearer. The intersection of a small portion of the event and topic groups is largely due to the fact that both aspects include verbs. Another interesting perspective is the location of the automatically learned keys (displayed as diamonds): while all the keys with ENTNET end up overlapping, indicating little difference among them, the keys learned by our method demonstrate semantic diversity, with each placed within its respective cluster. Note that the free key is learned to complement the event key, a difficult challenge given the two disjoint clusters.

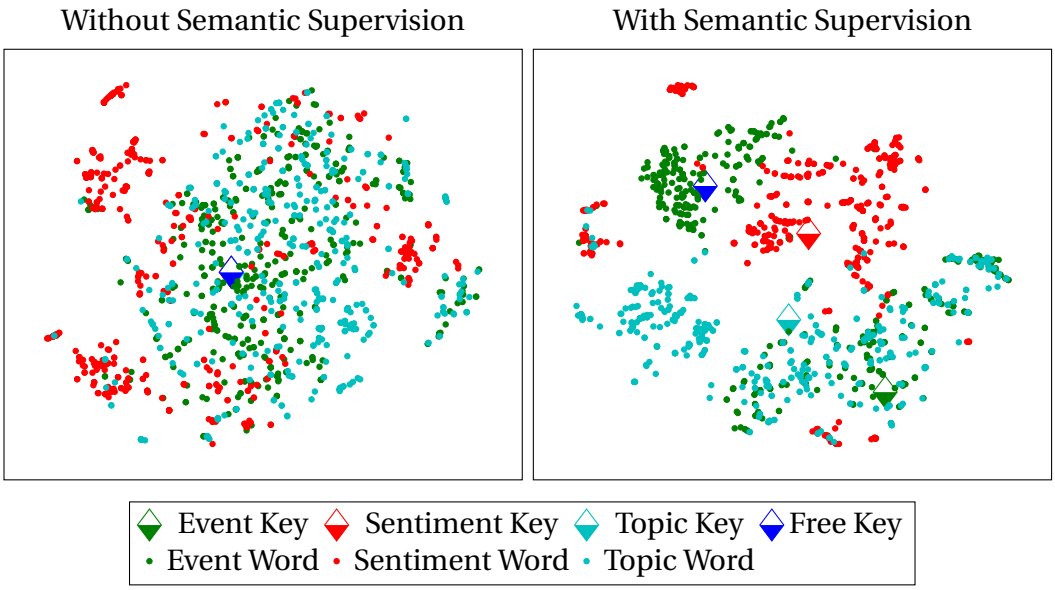

Figure 5.9 t-SNE scatter plot of aspect-triggering words (output representations $\boldsymbol{h}_i$ by BI-GRU, 500 randomly sampled from each aspect) and the learned keys. ENTNET (left) vs. our model (right). Best viewed in colour.

5.3.9.6 Ablation Study

We further perform an ablation study to analyse the utility of each component in Table 5.7.[11] The uni-directional variant, with its performance down to 77.8, is inferior to its bi-directional cousin. Removing all semantic supervision (resulting in 4 free memory chains) is also damaging to the accuracy (dropping to 77.6). Among the different types of semantic supervision, we see various degrees of performance degradation, with removing sentiment trajectory being the most detrimental, reflecting its value to the task. Interestingly, we observe improvement when removing event sequence supervision from consideration. We suspect that this is mainly due to the noise introduced by the rather inaccurate FrameNet representation output of SEMAFOR ($F_1 = 61.4\%$ in frame identification as reported in Das et al. (2010)). For example in Table 5.8, the word *woods*, a noun describing where the event is taking place, has been mistakenly identified as a frame-evoking target, which, in most cases, takes the form of a verb. It is possible that

[11] While there is some overlap (taking standard deviation into consideration) between different configurations, the key findings and conclusions presented in this section remain well-founded based on mean model performance.

Model	Acc.	SD
Our model	78.5	±0.5
—bi-directionality	77.8	±0.7
—all semantic supervision	77.6	±0.5
—event sequence	**78.7**	±0.2
—sentiment trajectory	75.9	±0.4
—topical consistency	77.3	±0.4
—free chain	77.0	±0.4

Table 5.7 Ablation study. Average accuracy over 5 runs on the Story Cloze test set (all subject to the same model selection criterion as our model). **Bold**: best performance. SD: standard deviation.

	Ricky	fell	while	hiking	in	the	woods
$\mathbf{l}^{Event}$	0	1	0	1	0	0	1
$\mathbf{l}^{Sentiment}$	0	1	0	0	0	0	0
$\mathbf{l}^{Topic}$	1	1	0	1	0	0	1

Table 5.8 An example of the linguistically motivated memory chain supervision binary labels, reproduced from Figure 5.5.

replacing SEMAFOR with the SemLM approach (with the extended frame definition to include explicit discourse markers, e.g. *but*) in Peng and Roth (2016) and Chaturvedi et al. (2017) may potentially reduce the amount of noise.

5.3.9.7 Subsequent Studies

While the systems presented in Table 5.6 achieved state of the art at the time of publication, subsequent studies have improved on these results, as shown in Table 5.9.

Radford et al. (2019) propose to leverage large-scale unlabelled text corpora for generative language modelling pre-training, followed by discriminative fine-tuning on a specific task. Similar to BERT, their model is built on stacked TRANSFORMER blocks with a simple, task-agnostic classifier on top, achieving substantial improvements over a wide range of diverse language understanding tasks, including ROC Story Cloze Test.

Building on the work of Radford et al. (2019) and inspired by studies in cognitive science (Sheorey and Mokhtari, 2001, Mokhtari and Sheorey, 2002, Mokhtari and Reichard,

Model	Acc.
Our model	78.5
—event sequence	78.7
Radford et al. (2019)	86.5
Sun et al. (2019b)	**88.1**

Table 5.9 Performance comparison against subsequent studies by Radford et al. (2019) and Sun et al. (2019b). We take the reported accuracy from their respective papers. **Bold**: best performance.

2002), Sun et al. (2019b) introduce three domain-independent reading strategies: (1) back and forth reading, (2) highlighting, and (3) self-assessment, to further fine-tune a pre-trained language model, improving the performance on this task by another 1.6% to an accuracy of 88.1%.

While both works outperform the method we present in this section, they come at the cost of time-consuming language modelling pre-training on large-scale corpora, a compute-intensive luxury many cannot afford. In contrast, our approach was under more modest data requirements, not requiring pre-training on domain-specific corpora (although it may potentially benefit from such additional resources).

5.4 Summary

In this chapter, we have addressed the third research question of relaxing limitations of (dynamic) memory networks to better capture entities and their interactions. In this regard, we have demonstrated two methods amied at better discourse understanding. While orthogonal to one another, they both improve upon vanilla ENTNETs in their ability to either model entity interactions or store information more stably over long distances, thereby achieving more competitive performance on discourse tasks. Additionally, with the help of such methods, ENTNETs outperform previous state of the art models under more modest data requirements, eliminating the need for either external commonsense knowledge sources or large-scale, domain-specific corpora for language model training. Further in-depth analysis and ablation/sensitivity studies provide additional insights into

why such methods enable better learning dynamics on the selected tasks, confirming and validating our motivation and hypotheses.

In the next chapter, we conclude this thesis and point out a number of promising directions in this line of memory augmentation research.

Chapter 6

Conclusions

While much effort has been focused on word- or sentence-level phenomena, language is more complex than individual and isolated sentences, often comprised of groups of structured sentences expressing a coherent narrative, commonly known as discourse. The multi-sentence nature of discourse requires consideration of larger contexts, capturing long-range dependencies between distant elements, and modelling the relationships and interactions between entities, three key elements to discourse understanding. This poses a challenge to conventional models with limited memory capacity as they are unable to store information stably over long timescales, rendering them ineffective for discourse. Memory-augmented models, on the other hand, are better capable of storing and accessing knowledge with the help of external memory, making them a viable alternative and more suited for discourse tasks.

The goal of this thesis is to improve memory-augmented neural networks for better discourse understanding, focusing on the three aspects identified above. To this end, we first reviewed the literature on memory-augmented models in Chapter 2, with an emphasis on neural network-based models and their application to the text domain, in particular narrative and discourse comprehension tasks. We broadly classified such models into two major categories: static and dynamic memory networks, depending on how the contents of their memory components are updated. In the context of this thesis, we limited the scope to two particular models: (end-to-end) memory networks (MemN2Ns) (Weston

et al., 2015, Sukhbaatar et al., 2015) and recurrent entity networks (ENTNETS) (Henaff et al., 2017). Additionally, we also outlined a number of discourse datasets, focusing on those most relevant to this thesis and tasks we use to evaluate the efficacy of the proposed methods in Chapters 3, 4, and 5. We then investigated a number of approaches in the subsequent chapters and demonstrated their applicability to a wide array of discourse tasks, ranging from reasoning-focused question answering on synthetically generated text, to dialog state tracking, to commonsense story understanding. In this chapter, we first revisit our research questions by summarising the contents, findings and contributions of each chapter and then discuss future work.

6.1 Research Question Revisited

Question 1: *How can we increase memory capacity in existing (static) memory architectures while not causing significant overfitting, thereby making them more generalisable to a wide range of discourse tasks?*

In Chapter 3, we introduced a unified weight tying mechanism (UMEMN2N), bridging the gap between the existing methods of adjacent (ADJ) and layer-wise (LW) weight tying, both of which are designed for MEMN2Ns to ameliorate overfitting by making trainable weight matrices shared across memory hops. Not only do these two weight tying techniques come at the cost of limiting memory capacity, they also come with the necessity to choose between them. This decision is made even more difficult by their uneven performance on the bAbI dataset, that is, neither of them performs consistently well over all tasks. In contrast, the proposed unified approach is capable of dynamically choosing the appropriate type of weight tying based on the input, achieving uniformly high performance, thereby outperforming a collection of baselines on a range of discourse tasks, including reasoning-focused reading comprehension, goal-oriented dialog completion, and dialog state tracking. The experimental results can be further explained by the analysis and visualisation provided in Section 3.4.4, suggesting that the proposed model indeed favours the better weight tying method out of ADJ and LW, validating our hypothesis.

Question 2: *How can we enhance modelling capacity by memory augmentation to better capture long-range contextual dependencies in discourse?*

In Chapter 4, we turned our attention to long-range dependencies, which are prevalent in language, especially in discourse, and proposed a novel method for modelling such relationships between distant elements. In particular, we focused on a widely popular model, linear-chain conditional random fields (CRFs), which, despite successful applications to a broad list of NLP tasks, have been limited to consider local features only due to practical concerns regarding inference tractability. Attempts to fix this issue, such as coupling RNN-based encoders (e.g., LSTMs) with CRFs, still exhibit a significant locality bias in practice despite their theoretical superiority (Lai et al., 2015, Linzen et al., 2016) Taking inspiration from memory networks (MEMN2Ns), we proposed an extension to CRFs by integrating external memory, allowing the mode to have access to the entire input sequence, thereby looking beyond localised features. More specifically, this is achieved with attention over the entire memory. From the perspective of MEMN2Ns, the addition of CRF allows memory networks to model the dynamics and transitions between neighbouring steps, thereby improving prediction consistency. Evaluated on a diverse array of discourse tasks, the proposed model, named memory-enhanced conditional random fields (ME-CRFs), demonstrated superior performance to a collection of strong baselines both in terms of capturing long-range contextual dependencies and making more consistent predictive output, validating the utility of memory integration.

Question 3: *How can we relax limitations of (dynamic) memory networks to better capture entities and their interactions?*

In Chapter 5, we considered the other major type of memory networks, dynamic memory chains, to model another important aspect of discourse: entities, and, more importantly, how they are referenced and interact with one another. Among many entity-centric models, we focus on recurrent entity networks (ENTNETS) (Henaff et al., 2017) because of their promising potential, indicated by preliminary experiments on a select collection of discourse tasks. ENTNETS are designed based on the idea of capturing

the states of entities with multiple memory chains, each responsible for keeping track of a single entity. Despite the positive results, ENTNETs are somewhat limited in their applicability to a wide variety of discourse tasks due to the difference in input granularity (sentence vs. word), especially for fine-grained information extraction tasks. Additionally, training ENTNETs end-to-end often results in ambiguous distribution of responsibility among memory chains, causing confusion as to which entity should be tracked by which memory chain. In Chapter 5, we presented two techniques to improve the performance of ENTNETs with one focusing on the delayed activation of memory update and the other incorporating semantic supervision into the training objective. They both substantially boost model performance on discourse tasks, advancing state of the art at the time of publication. Further analysis and visualisation provided additional clarifications as to why the proposed methods outperform the baselines, allowing us to better understand the source of the performance gains.

6.2 Future Directions

While there are many possible ways in which the research in this thesis could be extended, in this section, we identify three promising avenues for future research.

6.2.1 Dynamic Entity Representation

There has been increasing interest in learning to represent entities on the fly with entity-centric models. To this end, Yang et al. (2017) introduce a "reference-aware" language model, taking entities stored in different information sources into consideration. Kobayashi et al. (2017) propose a new task, a variant of language modelling where entity mentioned are normalised to anonymised identifiers, serving as a testbed for their entity-aware model. Ji et al. (2017) introduce a generative language model, named ENTITYNLM, capable of increasing its memory capacity, allowing it to accommodate previously unseen entities upon seeing one, and dynamically construct representations for them on the fly. While all making impressive progress on a number of tasks, they are limited by their shortcomings,

as identified in Section 2.4.2.1, such as demanding annotation requirements or a costly inference step relying on highly computationally-expensive importance sampling.

More recently, Liu et al. (2019) introduce a referential reader (REFREADER) with a fixed-size memory.[1] Different from recurrent entity networks (ENTNETS) (Henaff et al., 2017), a REFREADER is allowed to reuse its memory cells, re-writing them to keep track of different entities while processing text. While ENTITYNLM allows its memory to grow, it comes at the cost of memory efficiency. The REFREADER model, in contrast, can recycle its memory cells by overwriting the one with the lowest memory salience value, a decision modelled using the Gumbel softmax function (Jang et al., 2017).

This line of research can be further extended by taking inspiration from previous studies. For example, centering theory, which presents a unified framework for characterising the relationship between discourse structure and entity reference (Grosz et al., 1995), can be incorporated into such models, which may provide more grounded discourse-level information and be beneficial to downstream tasks. More specifically, models can exploit the entity grid representation of Barzilay and Lapata (2008) to better quantify the coherence between utterances, thereby enabling better discourse understanding. While the memory salience mechanism in REFREADERS resembles centering theory, which may be implicitly learned as indicated in the case study provided by Liu et al. (2019) where entities of less utility are eliminated from further consideration, explicit modelling of this theory requires careful design and further evaluation.

Another potential direction is to train such models in an unsupervised fashion, which addresses the issue of data annotation. In this setting, an entity-aware model can be trained to recognise entities and utilise external memory cells to keep track of their evolution. To reduce training difficulties, a simple set of rules may be used to guide learning, such as singular/plural and gender agreement. To this end, many attempts have been made (Liu et al., 2019, Joshi et al., 2019, Sun et al., 2019c), showing promising results and potential.

[1] Although the work of the author of this thesis, the paper by Liu et al. (2019) is based on the work carried out as part of an internship at Facebook AI Research and therefore excluded from this thesis.

6.2.2 Incorporating External Knowledge

The development of large-scale knowledge bases (KBs), such as Freebase (Bollacker et al., 2008) and ConceptNet (Speer et al., 2017), allows the incorporation of structured information, typically in the form of a triple, into machine learning models, facilitating research in leveraging such external knowledge to help make better predictions. their large size, exploiting such knowledge sources often requires careful design and consideration, restricting the utility of KBs. To this end, Miller et al. (2016) propose to store knowledge facts in a key-value memory network. More precisely, information is accessed in a three-step process: (1) key hashing — a small subset of candidates are selected based on their lexical overlap with a question; (2) key addressing — attention over the keys of every candidate in the subset; and (3) value reading — a weighted sum, according to the attention weights, of the values of each candidate. This effectively allows the model to benefit from the incorporation of large-scale KBs with little access overhead. This line of research is particularly helpful in the question answering domain. Recently, many have made impressive progress with the application of key-value memory networks to question answering while incorporating KBs (Chen et al., 2019b, Xu et al., 2019).

Another potential task is commonsense reasoning. While some types of commonsense relationships are explicitly stated in unstructured text, the vast majority of them are implicit, rendering them difficult to capture (Gordon and Van Durme, 2013, Bosselut et al., 2019). The incorporation of KBs, particularly with the help of memory augmentation, is therefore necessary and can help reduce learning difficulties. To this end, Das et al. (2017) propose a universal schema, storing unified representations of knowledge from either unstructured text or KB triples in a memory network, thereby allowing the model to leverage heterogeneous knowledge sources. Mihaylov and Frank (2018) demonstrate the efficacy of a key-value memory network when used to store external commonsense knowledge on a reading comprehension task.

Despite the popularity of large-scale language model pre-training, such as BERT (Devlin et al., 2019) or GPT(-2) (Radford et al., 2018, 2019), it remains to be seen whether knowledge stored with such memory-augmentation techniques can be incorporated to

help the training process and making the models more knowledge-aware. The work by Zhang et al. (2019) provides a good starting point by aligning words in a sentence with their entity representations in a knowledge graph. Bosselut et al. (2019) build on the work of GPT and frame the task of learning knowledge-enhanced language representation as a generative process where, given the subject and relation in a triple, the goal is to generatively predict the object. While both making explicit use of external KBs, they focus on encoding knowledge within model parameters and do not consider memory augmentation, which has been demonstrated to be of utility in the incorporation of commonsense knowledge (Das et al., 2017, Mihaylov and Frank, 2018).

There has been a recent surge of popularity and interest in this area with many new datasets constructed (Talmor et al., 2019, Ostermann et al., 2019, Huang et al., 2019, Sap et al., 2019), leaving much room for further investigation and improvements.

6.2.3 Identifying Connections with Cognitive Science

The study of memory augmentation and its application to the text domain stem from the assumption that parallels can be drawn between the memory component in a neural network and the human cognitive processes related to memory. Wehbe et al. (2014) present a model capable of distinguishing two story segments with considerably high accuracy by predicting fMRI activities of subjects reading a story involving interactions among multiple characters.

Input in other formats can be another possible avenue to explore. For instance, participants in a listening test can be seen as likely candidates as such input forces test-takers to memorise key pieces of information. It is therefore easier to observe and identify strong connections with the human cognitive processes. To this end, Yoon et al. (2017) study item generation in the context of a listening test, serving as a stepping stone for future research in this area.

6.3 Summary

In summary, this thesis motivates the need for memory-augmented neural networks for better discourse understanding. The multi-sentence nature of discourse requires consideration of larger context sizes as opposed to a single sentence, necessitating the ability to capture dependencies between distant elements and model the relationships and interactions of entities. While RNN-based models, in particular LSTMs, can store information for an arbitrary period of time, in practice, the memory component in such architectures can easily be influenced by neighbouring elements, rendering them ineffective for discourse. Memory-augmented models are better capable of storing and accessing knowledge and are therefore more suited for discourse tasks. In this thesis, we have demonstrated a number of methods for improving memory-augmented models to better understand discourse and shown that they can be applied to a diverse set of tasks. This validates the utility of memory augmentation in the context of discourse understanding, establishing a firm base for future studies to build upon.

References

C. Atkeson and S. Schaal. 1995. Memory-based neural networks for robot learning. *Neurocomputing*, 9(3):243–269.

R. Atkinson and R. Shiffrin. 1968. Human memory: A proposed system and its control process. In K. Spence and J. Spence, editors, *The Psychology of Learning and Motivation: Advances in Research and Theory*, volume 2, chapter 10, pages 89–195.

J. Ba, J. Kiros, and G. Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

S. Back, S. Yu, S. R. Indurthi, J. Kim, and J. Choo. 2018. MemoReader: Large-scale reading comprehension through neural memory controller. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2131–2140, Brussels, Belgium.

A. Baddeley. 1986. *Working memory*. Oxford University Press.

A. Baddeley, M. Eysenck, and M. Anderson. 2015. *Memory*. Psychology Press.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, USA.

P. Bailey. 1999. Searching for storiness: Story-generation from a reader's perspective. In *AAAI Fall Symposium on Narrative Intelligence*, pages 157–164, North Falmouth, USA.

A. Banino, A. P. Badia, R. Köster, M. J. Chadwick, V. Zambaldi, D. Hassabis, C. Barry, M. Botvinick, D. Kumaran, and C. Blundell. 2020. Memo: A deep network for flexible combination of episodic memories. *arXiv preprint arXiv:2001.10913*.

R. Barzilay and M. Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

L. Baum. 1972. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In *Inequalities III: Proceedings of the Third Symposium on Inequalities*, pages 1–8, Los Angeles, USA.

Y. Bengio, P. Frasconi, and P. Simard. 1993. The problem of learning long-term dependencies in recurrent networks. In *Proceedings of International Conference on Neural Networks*, pages 1183–1188, San Francisco, USA.

Y. Bengio, J. Louradour, R. Collobert, and J. Weston. 2009. Curriculum learning. In *Proceedings of the 26th International Conference on Machine Learning*, pages 41–48, Montréal, Canada.

Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, USA.

S. Bird, E. Klein, and E. Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Y. Bisk, S. Reddy, J. Blitzer, J. Hockenmaier, and M. Steedman. 2016. Evaluating induced CCG parsers on grounded semantic parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2022–2027, Austin, USA.

K. Bock and C. Miller. 1991. Broken agreement. *Cognitive Psychology*, 23(1):45–93.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, Vancouver, Canada.

A. Bordes and J. Weston. 2017. Learning end-to-end goal-oriented dialog. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France.

A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Celikyilmaz, and Y. Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy.

M. Bugert, Y. Puzikov, A. Rücklé, J. Eckle-Kohler, T. Martin, E. Martínez-Cámara, D. Sorokin, M. Peyrard, and I. Gurevych. 2017. LSDSem 2017: Exploring data generation methods for the story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (LSDSem 2017)*, pages 56–61, Valencia, Spain.

K. Burton, A. Java, I. Soboroff, et al. 2009. The ICWSM 2009 Spinn3r dataset. In *Proceedings of the Third Annual Conference on Weblogs and Social Media*, San Jose, USA.

E. Cambria, S. Poria, R. Bajpai, and B. Schuller. 2016. Senticnet 4: A semantic resource for sentiment analysis based on conceptual primitives. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2666–2677, Osaka, Japan.

N. Chambers and D. Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797, Columbus, USA.

E. Charniak. 1972. Toward a model of children's story comprehension. Technical report, Massachusetts Institute of Technology.

S. Chaturvedi, H. Peng, and D. Roth. 2017. Story comprehension for predicting what happens next. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1603–1614, Copenhagen, Denmark.

S. Chaudhari, G. Polatkan, R. Ramanath, and V. Mithal. 2019. An attentive survey of attention models. *arXiv preprint arXiv:1904.02874*.

D. Chen, J. Bolton, and C. Manning. 2016. A thorough examination of the CNN/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2358–2367, Berlin, Germany.

D. Chen and C. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar.

P. Chen, Z. Sun, L. Bing, and W. Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 452–461, Copenhagen, Denmark.

X. Chen, J. Xu, and B. Xu. 2019a. A working memory model for task-oriented dialog response generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2687–2693, Florence, Italy.

Y. Chen, L. Wu, and M. Zaki. 2019b. Bidirectional attentive memory networks for question answering over knowledge bases. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2913–2923, Minneapolis, USA.

J. Cheng, L. Dong, and M. Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, USA.

K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar.

K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

T. Ciodaro, D. Deva, J. De Seixas, and D. Damazio. 2012. Online particle detection with neural networks based on topological calorimetry information. In *Journal of Physics: Conference Series*, volume 368, pages 12–30.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

T. Cover and P. Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

H.-J. Dai, P.-T. Lai, Y.-C. Chang, and R. Tsai. 2015. Enhancing of chemical compound and drug name recognition using representative tag scheme and fine-grained tokenization. *Journal of Cheminformatics*, 7(1).

D. Das, N. Schneider, D. Chen, and N. A. Smith. 2010. Probabilistic frame-semantic parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 948–956, Los Angeles, USA.

R. Das, M. Zaheer, S. Reddy, and A. McCallum. 2017. Question answering on knowledge bases and text using universal schema and memory networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 358–365, Vancouver, Canada.

S. Das, C. Giles, and G.-Z. Sun. 1992. Learning context-free grammars: Capabilities and limitations of a recurrent neural network with an external stack memory. In *Proceedings of The Fourteenth Annual Conference of Cognitive Science Society*, pages 791–795, San Mateo, USA.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, USA.

B. Dhingra, K. Mazaitis, and W. W. Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

Z.-Y. Dou. 2017. Capturing user and product information for document level sentiment analysis with deep memory network. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 521–526, Copenhagen, Denmark.

C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China.

J. Eisenstein. 2018. *Natural Language Processing*. The MIT Press.

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

Y. Feng, S. Zhang, A. Zhang, D. Wang, and A. Abel. 2017. Memory-augmented neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1390–1399, Copenhagen, Denmark.

J. Finkel, T. Grenager, and C. Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, Ann Arbor, USA.

E. Forster. 1927. *Aspects of the Novel*. Edward Arnold.

Y. Fu and Y. Feng. 2018. Natural answer generation with heterogeneous memory. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 185–195, New Orleans, USA.

Y. Gal and Z. Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of Advances in Neural Information Processing Systems 29*, pages 1027–1035, Barcelona, Spain.

S. Geman and D. Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

X. Glorot, A. Bordes, and Y. Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the 14th International Conference on Artifical Intelligence and Statistics*, pages 315–323, Ft. Lauderdale, USA.

P. Goldman-Rakic. 1987. Circuitry of primate prefrontal cortex and regulation of behavior by representational memory. *Handbook of physiology*, 5(1):373–417.

P. Goldman-Rakic. 1995. Cellular basis of working memory. *Neuron*, 14(3):477–485.

I. Goodfellow, Y. Bengio, and A. Courville. 2016. *Deep Learning*. The MIT Press.

A. Gordon and R. Swanson. 2009. Identifying personal stories in millions of weblog entries. In *Proceedings of the Third International Conference on Weblogs and Social Media, Data Challenge Workshop*, San Jose, USA.

J. Gordon and B. Van Durme. 2013. Reporting bias and knowledge acquisition. In *Proceedings of the 2013 workshop on Automated Knowledge Base Construction*, pages 25–30, San Francisco, USA.

A. Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

A. Graves, A.-r. Mohamed, and G. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, Vancouver, Canada.

A. Graves, G. Wayne, and I. Danihelka. 2014. Neural Turing machines. *arXiv preprint arXiv:1410.5401*.

A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. Badia, K. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

E. Grefenstette, K. Hermann, M. Suleyman, and P. Blunsom. 2015. Learning to transduce with unbounded memory. In *Proceedings of Advances in Neural Information Processing Systems 28*, pages 1828–1836. Montréal, Canada.

B. Grosz, S. Weinstein, and A. Joshi. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

J. Gu, Z. Lu, H. Li, and V. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany.

M. Han, M. Kang, H. Jung, and S. Hwang. 2019. Episodic memory reader: Learning what to remember for question answering from streaming data. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4407–4417, Florence, Italy.

K. He, X. Zhang, S. Ren, and J. Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision*, pages 1026–1034, Santiago, Chile.

F. C. Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles. 2015. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, Boston, USA.

M. Henaff, J. Weston, A. Szlam, A. Bordes, and Y. LeCun. 2017. Tracking the world state with recurrent entity networks. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France.

M. Henderson, B. Thomson, and J. Williams. 2013. Dialog state tracking challenge 2 & 3. *http://camdial.org/~mh521/dstc/downloads/handbook.pdf*.

M. Henderson, B. Thomson, and J. Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272, Philadelphia, USA.

K. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of Advances in Neural Information Processing Systems 28*, pages 1693–1701. Montréal, Canada.

F. Hill, A. Bordes, S. Chopra, and J. Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of the 4th International Conference on Learning Representations*, San Juan, Puerto Rico.

S. Hochreiter. 1991. Untersuchungen zu dynamischen neuronalen netzen. Master's thesis, Technische Universität München.

S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. Kremer and J. Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*, pages 237–243.

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

J. Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558.

D. Hu. 2018. An introductory survey on attention mechanisms in NLP problems. *arXiv preprint arXiv:1811.05544*.

G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the 2017 IEEE International Conference on Computer Vision*, pages 2261–2269, Venice, Italy.

L. Huang, R. Le Bras, C. Bhagavatula, and Y. Choi. 2019. Cosmos QA: Machine reading comprehension with contextual commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2391–2401, Hong Kong, China.

Z. Huang, W. Xu, and K. Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

E. Jang, S. Gu, and B. Poole. 2017. Categorical reparameterization with Gumbel-softmax. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France.

Y. Ji, C. Tan, S. Martschat, Y. Choi, and N. Smith. 2017. Dynamic entity representations in neural language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1830–1839, Copenhagen, Denmark.

Y. Jiang and M. Bansal. 2018. Closed-book training to improve summarization encoder memory. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4067–4077, Brussels, Belgium.

M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy. 2019. Spanbert: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529*.

M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada.

A. Joulin and T. Mikolov. 2015. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Proceedings of Advances in Neural Information Processing Systems 28*, pages 190–198. Montréal, Canada.

D. Jurafsky and J. Martin. 2009. *Speech and Language Processing (2nd Edition)*. Prentice Hall.

Kaggle. 2014. Higgs boson machine learning challenge. https://www.kaggle.com/c/higgs-boson. Accessed: Aug 28th, 2018.

B. Kim, H. Kim, and G. Kim. 2019. Abstractive summarization of Reddit posts with multi-level memory networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2519–2531, Minneapolis, USA.

S. Kim, L. Wang, and T. Baldwin. 2010. Tagging and linking web forum posts. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 192–202, Uppsala, Sweden.

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

D. Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, USA.

S. Kiritchenko, X. Zhu, C. Cherry, and S. Mohammad. 2014. NRC-Canada-2014: Detecting Aspects and Sentiment in Customer Reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 437–442, Dublin, Ireland.

S. Kobayashi, N. Okazaki, and K. Inui. 2017. A neural language model for dynamically representing the meanings of unknown words and entities in a discourse. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 473–483, Taipei, Taiwan.

J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber. 2014. A clockwork RNN. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1863–1871, Bejing, China.

R. Krishna, K. Hata, F. Ren, L. Fei-Fei, and J. Carlos Niebles. 2017. Dense-captioning events in videos. In *Proceedings of the 2017 IEEE International Conference on Computer Vision*, pages 706–715, Venice, Italy.

V. Krishnan and C. Manning. 2006. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1121–1128, Sydney, Australia.

A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, and R. Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the 33rd International Conference on Machine Learning*, New York, USA.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, San Francisco, USA.

S. Lai, L. Xu, K. Liu, and J. Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2267–2273, Austin, USA.

T. Lai, Q. H. Tran, T. Bui, and D. Kihara. 2019. A gated self-attention memory network for answer selection. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5955–5961, Hong Kong, China.

G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, USA.

Y. LeCun. 1989. Generalization and network design strategies. Technical report, University of Toronto.

Y. LeCun, Y. Bengio, and G. Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.

H. Levesque, E. Davis, and L. Morgenstern. 2012. The Winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 552–561, Rome, Italy.

O. Levy, K. Lee, N. FitzGerald, and L. Zettlemoyer. 2018. Long short-term memory as a dynamically computed element-wise weighted sum. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 732–739, Melbourne, Australia.

B. Liang, J. Du, R. Xu, B. Li, and H. Huang. 2019. Context-aware embedding for targeted aspect-based sentiment analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4678–4683, Florence, Italy.

S. Liao and R. Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797, Uppsala, Sweden.

Z. Lin, X. Huang, F. Ji, H. Chen, and Y. Zhang. 2019. Task-oriented conversation generation using heterogeneous memory networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4550–4559, Hong Kong, China.

W. Ling, Y. Tsvetkov, S. Amir, R. Fermandez, C. Dyer, A. W. Black, I. Trancoso, and C.-C. Lin. 2015. Not all contexts are created equal: Better word representations with variable attention. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1367–1372, Lisbon, Portugal.

T. Linzen, E. Dupoux, and Y. Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

B. Liu, M. Hu, and J. Cheng. 2005. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th International Conference on World Wide Web*, pages 342–351, Chiba, Japan.

F. Liu and J. Perez. 2017. Gated end-to-end memory networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1–10, Valencia, Spain.

F. Liu, L. Zettlemoyer, and J. Eisenstein. 2019. The referential reader: A recurrent entity network for anaphora resolution. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5918–5925, Florence, Italy.

P. Liu, X. Qiu, and X. Huang. 2016. Deep multi-task learning with shared memory for text classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 118–127, Austin, USA.

R. Lowe, N. Pow, I. Serban, and J. Pineau. 2015. The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294, Prague, Czech Republic.

T. Luong, H. Pham, and C. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal.

J. Ma, R. Sheridan, A. Liaw, G. Dahl, and V. Svetnik. 2015. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of Chemical Information and Modeling*, 55(2):263–274.

Y. Ma, H. Peng, and E. Cambria. 2018. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 5876–5883, New Orleans, USA.

N. Majumder, S. Poria, A. Gelbukh, M. Akhtar, E. Cambria, and A. Ekbal. 2018. IARM: Inter-aspect relation modeling with memory networks in aspect-based sentiment analysis. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3402–3411, Brussels, Belgium.

C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

M. Manshadi, R. Swanson, and A. S. Gordon. 2008. Learning a probabilistic model of event sequences from internet weblog stories. In *Proceedings of 21st Conference of the Florida AI Society, Applied Natural Language Processing Track*, pages 159–164, Coconut Grove, USA.

S. Maruf and G. Haffari. 2018. Document context neural machine translation with memory networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1275–1284, Melbourne, Australia.

H. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. Hrafnkelsson, T. Boulos, and J. Kubica. 2013. Ad click prediction: A view from the trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1222–1230, Chicago, USA.

T. Mihaylov and A. Frank. 2017. Story cloze ending selection baselines and data examination. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (LSDSem 2017)*, pages 87–92, Valencia, Spain.

T. Mihaylov and A. Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832, Melbourne, Australia.

T. Mikolov. 2012. *Statistical Language Models based on Neural Networks.* Ph.D. thesis, Brno University of Technology.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations*, Scottsdale, USA.

T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. Ranzato. 2015. Learning longer memory in recurrent neural networks. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, USA.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in Neural Information Processing Systems 26*, pages 3111–3119, Stateline, USA.

A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, USA.

G. Miller. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63(2):81.

M. Minsky and S. Papert. 1969. *Perceptron: an introduction to computational geometry*. The MIT Press.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1928–1937, New York, USA.

K. Mokhtari and C. Reichard. 2002. Assessing students' metacognitive awareness of reading strategies. *Journal of Educational Psychology*, 94(2):249–259.

K. Mokhtari and R. Sheorey. 2002. Measuring ESL students' awareness of reading strategies. *Journal of Developmental Education*, 25(3):2–11.

N. Mostafazadeh, N. Chambers, X. He, D. Parikh, D. Batra, L. Vanderwende, P. Kohli, and J. Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, USA.

N. Mostafazadeh, M. Roth, A. Louis, N. Chambers, and J. Allen. 2017. LSDSem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (LSDSem 2017)*, pages 46–51, Valencia, Spain.

N. Mrkšić, D. Ó Séaghdha, T.-H. Wen, B. Thomson, and S. Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. pages 1777–1788, Vancouver, Canada.

S. Mukherjee and P. Bhattacharyya. 2012. Sentiment analysis in Twitter with lightweight discourse analysis. In *Proceedings of COLING 2012*, pages 1847–1864, Mumbai, India.

J. Nairne. 1990. A feature model of immediate memory. *Memory & Cognition*, 18(3):251–269.

J. Nairne. 2002. Remembering over the short-term: The case against the standard model. *Annual Review of Psychology*, 53(1):53–81.

I. Neath and A. Surprenant. 2003. *Human memory: An introduction to research, data, and theory (2nd edn.)*. Wadsworth Publishing.

J. von Neumann. 1993. First draft of a report on the edvac. *IEEE Annals of the History of Computing*, 15(4):27–75.

D. Norman. 1970. *Models of human memory*. Academic Press.

F. J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

S. Ostermann, M. Roth, and M. Pinkal. 2019. MCScript2.0: A machine comprehension corpus focused on script events and participants. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 103–117, Minneapolis, USA.

R. Pascanu, T. Mikolov, and Y. Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1310–1318, Atlanta, USA.

S. Pateria. 2016. Aspect based sentiment analysis using sentiment flow with local and non-local neighbor information. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2635–2646, Osaka, Japan.

H. Peng and D. Roth. 2016. Two discourse driven language models for semantics. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 290–300, Berlin, Germany.

J. Pennington, R. Socher, and C. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

J. Perez and F. Liu. 2017. Dialog state tracking, a machine reading approach using memory network. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 305–314, Valencia, Spain.

M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar. 2014. SemEval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland.

A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. 2019. Language models are unsupervised multitask learners. Technical report, OpenAI.

J. Rae, J. Hunt, I. Danihelka, T. Harley, A. Senior, G. Wayne, A. Graves, and T. Lillicrap. 2016. Scaling memory-augmented neural networks with sparse reads and writes. In *Proceedings of Advances in Neural Information Processing Systems 29*, pages 3621–3629, Barcelona, Spain.

P. Rajpurkar, R. Jia, and P. Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, USA.

L. Ratinov and D. Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, USA.

M. Richardson, C. Burges, and E. Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, USA.

A. Rohrbach, A. Torabi, M. Rohrbach, N. Tandon, C. Pal, H. Larochelle, A. Courville, and B. Schiele. 2017. Movie description. *International Journal of Computer Vision*, 123(1):94–120.

D. Rumelhart, G. Hinton, and R. Williams. 1986a. Learning internal representations by error propagation. In D. Rumelhart and J. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pages 318–362.

D. Rumelhart, G. Hinton, and R. Williams. 1986b. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

M. Saeidi, G. Bouchard, M. Liakata, and S. Riedel. 2016. Sentihood: Targeted aspect based sentiment analysis dataset for urban neighbourhoods. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1546–1556, Osaka, Japan.

M. Sap, R. Le Bras, E. Allaway, C. Bhagavatula, N. Lourie, H. Rashkin, B. Roof, N. A. Smith, and Y. Choi. 2019. ATOMIC: an atlas of machine commonsense for if-then reasoning. In *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, pages 3027–3035, Honolulu, USA.

R. Schank and R. Abelson. 1977. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press.

J. Schmidhuber. 1992. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139.

J. Schmidhuber. 1993. A 'self-referential' weight matrix. In *International Conference on Artificial Neural Networks*, pages 446–450, Amsterdam, The Netherlands.

L. Schubert and C. Hwang. 2000. Episodic logic meets little red riding hood: A comprehensive, natural representation for language understanding. In L. Iwanska and S. Shapiro, editors, *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*, pages 111–174.

R. Schwartz, M. Sap, I. Konstas, L. Zilles, Y. Choi, and N. Smith. 2017. The effect of different writing tasks on linguistic style: A case study of the ROC story cloze task. pages 15–25, Vancouver, Canada.

M. Seo, S. Min, A. Farhadi, and H. Hajishirzi. 2017. Query-reduction networks for question answering. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France.

H. Seung. 1998. Continuous attractors and oculomotor control. *Neural Networks*, 11(7-8):1253–1258.

R. Sheorey and K. Mokhtari. 2001. Differences in the metacognitive awareness of reading strategies among native and non-native readers. *System*, 29(4):431–449.

H. Siegelmann and E. Sontag. 1995. On the computational power of neural nets. *Journal of Computer and System Sciences*, 50(1):132–150.

S. Singh, D. Litman, M. Kearns, and M. Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*, 16(1):105–133.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng, and C. Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, USA.

R. Speer, J. Chin, and C. Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 4444–4451, San Francisco, USA.

S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. 2015. End-to-end memory networks. In *Proceedings of Advances in Neural Information Processing Systems 28*, pages 2440–2448, Montréal, Canada.

C. Sun, L. Huang, and X. Qiu. 2019a. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 380–385, Minneapolis, USA.

G. Sun, C. Giles, H. Chen, and Y. Lee. 1993. The neural network pushdown automation: Model, stack and learning simulations. Technical report, College Park, USA.

K. Sun, D. Yu, D. Yu, and C. Cardie. 2019b. Improving machine reading comprehension with general reading strategies. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2633–2643, Minneapolis, USA.

Y. Sun, S. Wang, Y. Li, S. Feng, X. Chen, H. Zhang, X. Tian, D. Zhu, H. Tian, and H. Wu. 2019c. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

I. Sutskever, O. Vinyals, and Q. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of Advances in Neural Information Processing Systems 27*, pages 3104–3112, Montréal, Canada.

C. Sutton and A. McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. Technical report, University of Massachusetts.

A. Talmor, J. Herzig, N. Lourie, and J. Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, USA.

D. Tang, B. Qin, X. Feng, and T. Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3298–3307, Osaka, Japan.

D. Tang, B. Qin, and T. Liu. 2016b. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 214–224, Austin, USA.

Z. Tian, W. Bi, X. Li, and N. L. Zhang. 2019. Learning to abstract for memory-augmented conversational response generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3816–3825, Florence, Italy.

E. F. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, Edmonton, Canada.

K. Tran, A. Bisazza, and C. Monz. 2016. Recurrent memory networks for language modeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 321–331, San Diego, California.

S. Turner. 1994. *The Creative Process: A Computer Model of Storytelling and Creativity*. Psychology Press.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, Ł. Kaiser, and I. Polosukhin. 2017. Attention is all you need. In *Proceedings of Advances in Neural Information Processing Systems 30*, pages 5998–6008, Long Beach, USA.

O. Vinyals, M. Fortunato, and N. Jaitly. 2015. Pointer networks. In *Proceedings of Advances in Neural Information Processing Systems 28*, pages 2692–2700, Montréal, Canada.

A. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

J. Wagner, P. Arora, S. Cortes, U. Barman, D. Bogdanova, J. Foster, and L. Tounsi. 2014. DCU: Aspect-based polarity classification for SemEval task 4. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 223–229, Dublin, Ireland.

F. Wang, Y. Wu, and L. Qiu. 2012. Exploiting discourse relations for sentiment analysis. In *Proceedings of COLING 2012: Posters*, pages 1311–1320, Mumbai, India.

L. Wang. 2014. *Knowledge discovery and extraction of domain-specific web data*. Ph.D. thesis, The University of Melbourne.

L. Wang, M. Lui, S. Kim, J. Nivre, and T. Baldwin. 2011. Predicting thread discourse structure over technical web forums. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 13–25, Edinburgh, Scotland, UK.

M. Wang, N. Smith, and T. Mitamura. 2007. 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)*, pages 22–32, Prague, Czech Republic.

M. Wang, Z. Lu, H. Li, and Q. Liu. 2016. Memory-enhanced decoder for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 278–286, Austin, USA.

S. Wang, S. Mazumder, B. Liu, M. Zhou, and Y. Chang. 2018. Target-sensitive memory networks for aspect sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 957–967, Melbourne, Australia.

W. Wang, N. Yang, F. Wei, B. Chang, and M. Zhou. 2017. 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)*, pages 189–198, Vancouver, Canada.

L. Wehbe, B. Murphy, P. Talukdar, A. Fyshe, A. Ramdas, and T. Mitchell. 2014. Simultaneously uncovering the patterns of brain regions involved in different story reading subprocesses. *PLOS ONE*, 9(11):1–19.

J. Welbl, P. Stenetorp, and S. Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

J. Weston, A. Bordes, S. Chopra, A. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov. 2016. Towards AI-complete question answering: A set of prerequisite toy tasks. In *Proceedings of the 4th International Conference on Learning Representations*, San Juan, Puerto Rico.

J. Weston, S. Chopra, and A. Bordes. 2015. Memory networks. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, USA.

J. Williams, A. Raux, D. Ramachandran, and A. Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, Metz, France.

R. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

T. Winograd. 1972. Understanding natural language. *Cognitive Psychology*, 3(1):1–191.

C. Xiong, S. Merity, and R. Socher. 2016. Dynamic memory networks for visual and textual question answering. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2397–2406, New York, USA.

K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2048–2057.

K. Xu, Y. Lai, Y. Feng, and Z. Wang. 2019. Enhancing key-value memory neural networks for knowledge based question answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2937–2947, Minneapolis, USA.

Y. Yang, W. Yih, and C. Meek. 2015. 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, Lisbon, Portugal.

Z. Yang, P. Blunsom, C. Dyer, and W. Ling. 2017. Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1850–1859, Copenhagen, Denmark.

Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, USA.

D. Yogatama, Y. Miao, G. Melis, W. Ling, A. Kuncoro, C. Dyer, and P. Blunsom. 2018. Memory architectures in recurrent neural network language models. In *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, Canada.

S. Yoon, C. Lee, P. Houghton, M. Lopez, J. Sakano, A. Loukina, B. Krovetz, C. Lu, and N. Madnani. 2017. Analyzing item generation with natural language processing tools for the TOEIC® listening test. *ETS Research Report Series*, 2017(1):1–9.

A. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. Le. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. In *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, Canada.

W. Zaremba and I. Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615*.

R. Zellers, Y. Bisk, R. Schwartz, and Y. Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium.

J. Zeng, J. Li, Y. Song, C. Gao, M. R. Lyu, and I. King. 2018. Topic memory networks for short text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3120–3131, Brussels, Belgium.

A. Zhang, B. Culbertson, and P. Paritosh. 2017a. Characterizing online discussion using coarse discourse sequences. In *Proceedings of the 11th AAAI International Conference on Web and Social Media*, pages 357–366, Palo Alto, USA.

J. Zhang, Y. Feng, D. Wang, Y. Wang, A. Abel, S. Zhang, and A. Zhang. 2017b. Flexible and creative Chinese poetry generation using neural memory. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1364–1373, Vancouver, Canada.

Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy.

Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the 2015 IEEE International Conference on Computer Vision*, pages 19–27, Washington, DC, USA.

C. Zirn, M. Niepert, H. Stuckenschmidt, and M. Strube. 2011. Fine-grained sentiment analysis with structural features. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 336–344, Chiang Mai, Thailand.