NEURAL-BASED KNOWLEDGE TRANSFER IN NATURAL LANGUAGE PROCESSING

CHAO WANG

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO
OCTOBER 2021

Abstract

In Natural Language Processing (NLP), neural-based knowledge transfer, which is to transfer out-of-domain (OOD) knowledge to task-specific neural networks, has been applied to many NLP tasks. To further explore neural-based knowledge transfer in NLP, in this dissertation, we consider both structured OOD knowledge and unstructured OOD knowledge, and deal with several representative NLP tasks. For structured OOD knowledge, we study the neural-based knowledge transfer in Machine Reading Comprehension (MRC). In single-passage MRC tasks, to bridge the gap between MRC models and human beings, which is mainly reflected in the hunger for data and the robustness to noise, we integrate the neural networks of MRC models with the general knowledge of human beings embodied in knowledge bases. On the one hand, we propose a data enrichment method, which uses WordNet to extract inter-word semantic connections as general knowledge from each given passage-question pair. On the other hand, we propose a novel MRC model named Knowledge Aided Reader (KAR), which explicitly uses the above extracted general knowledge to assist its attention mechanisms. According to the experimental

results, KAR is comparable in performance with the state-of-the-art MRC models, and significantly more robust to noise than them. On top of that, when only a subset (20%–80%) of the training examples are available, KAR outperforms the state-of-the-art MRC models by a large margin, and is still reasonably robust to noise. In multi-hop MRC tasks, to probe the strength of Graph Neural Networks (GNNs), we propose a novel multi-hop MRC model named Graph Aided Reader (GAR), which uses GNN methods to perform multi-hop reasoning, but is free of any pre-trained language model and completely end-to-end. For graph construction, GAR utilizes the topic-referencing relations between passages and the entity-sharing relations between sentences, which is aimed at obtaining the most sensible reasoning clues. For message passing, GAR simulates a top-down reasoning and a bottom-up reasoning, which is aimed at making the best use of the above obtained reasoning clues. According to the experimental results, GAR even outperforms several competitors relying on pre-trained language models and filter-reader pipelines, which implies that GAR benefits a lot from its GNN methods. On this basis, GAR can further benefit from applying pre-trained language models, but pre-trained language models can mainly facilitate the within-passage reasoning rather than cross-passage reasoning of GAR. Moreover, compared with the competitors constructed as filter-reader pipelines, GAR is not only easier to train, but also more applicable to the low-resource cases. For unstructured OOD knowledge, we study the neural-based knowledge transfer in Natural Language Understanding (NLU), and focus on the neural-based knowledge

transfer between languages, which is also known as Cross-Lingual Transfer Learning (CLTL). To facilitate the CLTL of NLU models, especially the CLTL between distant languages, we propose a novel CLTL model named Translation Aided Language Learner (TALL), where CLTL is integrated with Machine Translation (MT). Specifically, we adopt a pre-trained multilingual language model as our baseline model, and construct TALL by appending a decoder to it. On this basis, we directly fine-tune the baseline model as an NLU model to conduct CLTL, but put TALL through an MT-oriented pre-training before its NLU-oriented fine-tuning. To make use of unannotated data, we implement the recently proposed Unsupervised Machine Translation (UMT) technique in the MT-oriented pre-training of TALL. According to the experimental results, the application of UMT enables TALL to consistently achieve better CLTL performance than the baseline model without using more annotated data, and the performance gain is relatively prominent in the case of distant languages.

Acknowledgements

I would like to thank my supervisor Prof. Hui Jiang for his consistent support and guidance during my PhD study. Furthermore, I would like to thank the other members in our research team for their collaborative efforts. I am also thankful to the EECS department of York University and all its staff for all the considerate help.

Publications

Conference Papers

- Chao Wang and Hui Jiang. "The Lower The Simpler: Simplifying Hierarchical Recurrent Models." 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). 2019.

- Chao Wang and Hui Jiang. "Explicit Utilization of General Knowledge in Machine Reading Comprehension." Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019.

- Chao Wang, Judith Gaspers, Quynh Do, and Hui Jiang. "Exploring Cross-Lingual Transfer Learning with Unsupervised Machine Translation." Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. 2021.

- Chao Wang and Hui Jiang. "Probing the Strength of Graph Neural Networks in Multi-hop Machine Reading Comprehension." To be submitted.

Qualifying Examination Report

- Chao Wang. “A Study of the Tasks and Models in Machine Reading Comprehension.” arXiv preprint arXiv:2001.08635. 2020.

Table of Contents

Abstract ii

Acknowledgements v

Publications vi

Table of Contents viii

List of Tables xiii

List of Figures xvi

1 Introduction 1

2 Background Introduction 7

2.1 Neural-based Approaches in NLP 7

2.1.1 Distributed Representations 7

2.1.2 Feed-forward Neural Networks 8

2.1.3 Recurrent Neural Networks 10
2.1.4 Attention Mechanisms 13
2.2 Neural-based Knowledge Transfer in NLP 15
2.2.1 Knowledge Base Encoding 15
2.2.2 Pre-trained Language Models 17
2.3 Related NLP Tasks 20
2.3.1 Machine Reading Comprehension (MRC) 20
2.3.2 Natural Language Understanding (NLU) 22

3 Explicit Utilization of General Knowledge in Machine Reading Comprehension 24

3.1 Introduction 24
3.2 Related Works 28
3.2.1 Attention Mechanisms for MRC Models 28
3.2.2 Performance-boosting Approaches for MRC Models 29
3.3 Data Enrichment Method 30
3.3.1 Semantic Relation Chain 30
3.3.2 Inter-word Semantic Connection 31
3.3.3 General Knowledge Extraction 32
3.4 Knowledge Aided Reader 33

3.4.1 Task Definition 33
3.4.2 Overall Architecture 34
3.4.3 Knowledge Aided Mutual Attention 37
3.4.4 Knowledge Aided Self Attention 39
3.5 Experiments 40
3.5.1 Experimental Settings 40
3.5.2 Model Comparison in both Performance and the Robustness to Noise 41
3.5.3 Model Comparison in the Hunger for Data 48
3.6 Analysis 49
3.7 Conclusion 50

4 Probing the Strength of Graph Neural Networks in Multi-hop Machine Reading Comprehension 51

4.1 Introduction 51
4.2 Related Works 55
4.2.1 Multi-passage Single-hop MRC Models 55
4.2.2 Multi-hop MRC Models 57
4.3 Graph Aided Reader 58
4.3.1 Task Definition 60

4.3.2 Graph Construction 60
4.3.3 Message Passing 63
4.3.4 Overall Architecture 67
4.4 Experiments 71
4.4.1 Experimental Settings 71
4.4.2 Model Comparison 72
4.4.3 Ablation Study 76
4.5 Conclusion 77

5 Exploring Cross-Lingual Transfer Learning with Unsupervised Machine Translation 78

5.1 Introduction 78
5.2 Related Works 81
5.3 Translation Aided Language Learner 84
5.3.1 Task Definition 84
5.3.2 Baseline Model 85
5.3.3 Proposed Model 88
5.4 Verification Experiments 94
5.4.1 Experimental Settings 94
5.4.2 Implementation details. 96

5.4.3 Experimental Results 97
5.4.4 Ablation Study 98
5.5 Further Discussion 99
5.6 Conclusion 100

6 Conclusion **101**

Bibliography **103**

List of Tables

2.1 NLU: slot filling and intent classification 22

3.1 Two examples about the importance of inter-word semantic connections to the reading comprehension ability of human beings: in the first one, we can find the answer because we know "facilitate" is a synonym of "help"; in the second one, we can find the answer because we know "Brooklyn" is a hyponym of "borough". 25

3.2 Model comparison based on SQuAD 1.1 and two of its adversarial sets: AddSent and AddOneSent. All the numbers are up to date as of October 18, 2018. Note that SQuAD 2.0 [1] is not involved in this chapter, because it requires MRC models to deal with the problem of answer triggering, but this chapter is aimed at improving the hunger for data and robustness to noise of MRC models. 42

3.3 With κ set to different values in the data enrichment method, we calculate the average number of inter-word semantic connections per word as an estimation of the amount of general knowledge, and evaluate the performance of KAR on the development set. 43

4.1 A multi-hop MRC example provided by [2]. 52

4.2 Model comparison on the test set of HotpotQA for the distractor setting. PLM means relying on a pre-trained language model. FRP means relying on a filter-reader pipeline. GSFs means being trained with ground-truth SFs. 73

4.3 Model comparison on the test set of HotpotQA for the fullwiki setting. The competitors are the published fully-functional single basic models that focus on multi-hop MRC. 74

4.4 Ablation results on the development set of HotpotQA for the distractor setting. 76

5.1 The translation performance of TALL on Wikipedia and the CLTL performance of both the baseline model and TALL on MultiATIS++. EN means English. JA means Japanese. DE means German. BSL means the baseline model. Gain means the CLTL performance gain of TALL over the baseline model. The gain numbers are in percentage and calculated as $(\mathrm{TALL} - \mathrm{BSL}) \div \mathrm{BSL}$. 93

5.2 The CLTL performance gain of TALL over the baseline model on the multi-domain multilingual NLU dataset. 94

List of Figures

3.1 An end-to-end MRC model: Knowledge Aided Reader (KAR) 34

3.2 With KAR, SAN, and QANet (without data augmentation) trained on the training subsets, we evaluate their performance on the development set. 45

3.3 With KAR, SAN, and QANet (without data augmentation) trained on the training subsets, we evaluate their performance on AddSent. 46

3.4 With KAR, SAN, and QANet (without data augmentation) trained on the training subsets, we evaluate their performance on AddOneSent. 47

4.1 A part of the graph constructed for the example in Table 4.1. The distracting passages are not shown, but they are involved in the constructed graph. 61

4.2 Our multi-hop MRC model: Graph Aided Reader (GAR). 67

5.1 The NLU-oriented fine-tuning of our baseline model. MLM means multilingual language model. 84

5.2 The MT-oriented pre-training and NLU-oriented fine-tuning of our proposed Translation Aided Language Learner (TALL). 87

1 Introduction

Natural Language Processing (NLP) is aimed at building models and systems that can automatically represent and analyze natural languages [3]. NLP covers a broad variety of tasks, which range from the basic tasks, such as Part-Of-Speech (POS) tagging, Named-Entity Recognition (NER), and text classification, to the high-level tasks, such as Information Retrieval (IR), Question Answering (QA), and Machine Translation (MT). It is the ability to address these tasks that lays the foundation of many real-world applications, such as search engines and virtual assistants. Due to the rapid advance and tremendous success of deep learning, neural networks have been the dominant techniques in NLP. So far, countless neural-based approaches, such as distributed representations [4, 5], Feed-forward Neural Networks (FNNs) [6, 7], Recurrent Neural Networks (RNNs) [8, 9], and attention mechanisms [10, 11, 12], have been proposed or borrowed to address NLP tasks. To successfully apply these approaches, it is necessary to train the corresponding neural networks in a proper way. From our point of view, training a neural network for an NLP task is much like imparting knowledge to the neural network, where

the knowledge comes from the training data provided with the NLP task, and is imparted to the neural network by optimizing a task-specific objective through gradient descent. However, besides utilizing the knowledge embodied in the training data, can we also utilize out-of-domain (OOD) knowledge to address NLP tasks?

Yes. In NLP, many methods have been proposed for transferring OOD knowledge to task-specific neural networks. In this dissertation, we describe these methods as performing neural-based knowledge transfer, and divide them into two categories, which separately correspond to structured OOD knowledge and unstructured OOD knowledge. Structured OOD knowledge refers to the knowledge embodied in knowledge bases, such as WordNet [13], ConceptNet [14], and Freebase [15]. Such knowledge is explicitly organized as "subject-predicate-object" triples, and thus forms a graph. Existing neural-based knowledge transfer methods for such knowledge mainly include knowledge base encoding, which is to encode the knowledge concepts (i.e. subjects, objects, and predicates) in a knowledge base into distributed representations, and thereby use the resulting knowledge representations to enhance the language representations in task-specific neural networks. Unstructured OOD knowledge refers to the knowledge embodied in large-scale corpora, such as Wikipedia dumps and arXiv dumps. Such knowledge is implicitly expressed in free text, and is usually very sparse. Existing neural-based knowledge transfer methods for such knowledge mainly include the pre-training and application of language models, where a language model, such as CoVe [16], ELMo [17], BERT [18], and GPT [19, 20], is

first pre-trained on a large-scale corpus, and then used to generate contextualized language representations in task-specific neural networks. It is worth mentioning that the above pre-training stage can take place on a multilingual corpus so that the pre-trained language model can be used to address cross-lingual NLP tasks.

To further explore neural-based knowledge transfer in NLP, in this dissertation, we consider both structured OOD knowledge and unstructured OOD knowledge, and deal with several representative NLP tasks. For structured OOD knowledge, we study the neural-based knowledge transfer in Machine Reading Comprehension (MRC), which requires a machine to read a context and answer a relevant question based on its comprehension to the context. We first focus on single-passage MRC tasks, where each context includes only a single passage. To bridge the gap between MRC models and human beings, which is mainly reflected in the hunger for data and the robustness to noise, we integrate the neural networks of MRC models with the general knowledge of human beings embodied in knowledge bases. On the one hand, we propose a data enrichment method, which uses WordNet to extract inter-word semantic connections as general knowledge from each given passage-question pair. On the other hand, we propose a novel MRC model named Knowledge Aided Reader (KAR), which explicitly uses the above extracted general knowledge to assist its attention mechanisms. According to the experimental results, KAR is comparable in performance with the state-of-the-art MRC models, and significantly more robust to noise than them. On top of that, when only a subset (20%–80%) of the

training examples are available, KAR outperforms the state-of-the-art MRC models by a large margin, and is still reasonably robust to noise.

Besides, for structured OOD knowledge, we also study the neural-based knowledge transfer in multi-hop MRC tasks, where each context includes multiple passages, and the question needs to be answered by performing multi-hop reasoning on at least two of them. In many multi-hop MRC models, multi-hop reasoning is realized by applying Graph Neural Networks (GNNs). This is a two-stage process, where the first stage is to construct a graph for each given context-question pair, and the second stage is to pass messages on the constructed graph. The constructed graph is expected to embody the clues for multi-hop reasoning, thus applying GNNs in this way can be seen as utilizing the knowledge embodied in a dynamically created knowledge base. Most of the existing GNN-based multi-hop MRC models heavily rely on pre-trained language models and filter-reader pipelines. Although these models are indeed good in performance, it is still not clear how much GNNs contribute to their performance. To probe the strength of GNNs in multi-hop MRC, we propose a novel multi-hop MRC model named Graph Aided Reader (GAR), which uses GNN methods to perform multi-hop reasoning, but is free of any pre-trained language model and completely end-to-end. For graph construction, GAR utilizes the topic-referencing relations between passages and the entity-sharing relations between sentences, which is aimed at obtaining the most sensible reasoning clues. For message passing, GAR simulates a top-down reasoning and a bottom-up reasoning, which

is aimed at making the best use of the above obtained reasoning clues. According to the experimental results, GAR even outperforms several competitors relying on pre-trained language models and filter-reader pipelines, which implies that GAR benefits a lot from its GNN methods. On this basis, GAR can further benefit from applying pre-trained language models, but pre-trained language models can mainly facilitate the within-passage reasoning rather than cross-passage reasoning of GAR. Moreover, compared with the competitors constructed as filter-reader pipelines, GAR is not only easier to train, but also more applicable to the low-resource cases.

For unstructured OOD knowledge, we study the neural-based knowledge transfer in Natural Language Understanding (NLU), which includes slot filling and intent classification, and focus on the neural-based knowledge transfer between languages, which is also known as Cross-Lingual Transfer Learning (CLTL). To facilitate the CLTL of NLU models, especially the CLTL between distant languages, we propose a novel CLTL model named Translation Aided Language Learner (TALL), where CLTL is integrated with MT. Specifically, we adopt a pre-trained multilingual language model, which is the state of the art in CLTL, as our baseline model, and construct TALL by appending a decoder to it. On this basis, we directly fine-tune the baseline model as an NLU model to conduct CLTL, but put TALL through an MT-oriented pre-training before its NLU-oriented fine-tuning. We believe that the MT-oriented pre-training can help TALL to enhance the correlation between the given source-target language pair in its representation space,

and thus can make CLTL easier to conduct in its NLU-oriented fine-tuning, especially when the source language and the target language are distant from each other. To make use of unannotated data, which is not only large in amount but also available for every language, we implement the recently proposed Unsupervised Machine Translation (UMT) technique in the MT-oriented pre-training of TALL. According to the experimental results, the application of UMT enables TALL to consistently achieve better CLTL performance than the baseline model without using more annotated data, and the performance gain is relatively prominent in the case of distant languages.

This dissertation is organized as follows. This chapter is a general introduction to this dissertation. Chapter 2 further introduces the background knowledge of this dissertation. Chapter 3, chapter 4, and chapter 5 separately correspond to the three works introduced above. Chapter 6 is a general conclusion to this dissertation.

2 Background Introduction

2.1 Neural-based Approaches in NLP

2.1.1 Distributed Representations

Distributed representations are the dense vectors used to represent language concepts, where each language concept is distributed over all the dimensions of its dense vector representation. The most common distributed representations are word embeddings, which refer to the distributed representations of words used as the inputs of neural networks. There are mainly two methods used for learning word embeddings, namely Continuous Bag of Words (CBoW) and Skip-Gram [4, 5]. In CBoW, the distributed representations of surrounding words are concatenated to predict the target word in the middle, while in Skip-Gram, the distributed representation of a target word is used to predict the surrounding words. By taking word embeddings as inputs, a neural network can generate not only higher-level distributed representations of words, but also distributed representations of other language concepts, such as sentences and passages, and thereby use the generated

distributed representations to address the NLP task at hand.

2.1.2 Feed-forward Neural Networks

Feed-forward neural networks (FNNs) are the most common type of neural networks. An FNN usually consists of multiple layers, where the first layer is called the input layer, the last layer is called the output layer, and the layers in between are called the hidden layers. The layers in an FNN are connected in a way that there is no cycle. That is to say, the information in an FNN moves in a single-directional manner, from the input layer through the hidden layers to the output layer. A typical category of FNNs is perceptrons. In a perceptron used for NLP, the input layer is usually word embeddings, while both the hidden layers and the output layer are fully-connected layers:

$$y = f(Wx + b)$$

where x represents the layer input, y represents the layer output, $f(\cdot)$ represents an activation function, W is a trainable weight, and b is a trainable bias. For hidden layers, ReLU [21] is the most widely-used activation function:

$$\mathrm{ReLU}(z) = \max(0, z)$$

Since most NLP tasks are classification tasks, Softmax is usually used as the activation function of an output layer, which predicts the probability of each class:

$$\mathrm{Softmax}_i(z) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

where i refers to the i-th class, j refers to the j-th class, and N represents the number of classes. Actually, for binary classification tasks (i.e. $N = 2$), Softmax is usually replaced with Sigmoid, which predicts the probability of the positive class:

$$\text{Sigmoid}(z) = \frac{1}{1 + e^{-z}}$$

Another typical category of FNNs is Convolutional Neural Networks (CNNs). Although originally aimed at Computer Vision (CV), CNNs have also been widely used in NLP [6, 7]. The difference between CNNs and perceptrons is that the hidden layers of a CNN include not only fully-connected layers but also convolutional layers and pooling layers. The first hidden layer of a CNN is a convolutional layer, it maintains a set of trainable kernels, and uses each of them to perform convolution on the layer input, where ReLU is usually used as the activation function. The output of the convolutional layer is then passed to a pooling layer, which uses certain pooling operation to reduce the dimentionality of the convolution result. Max pooling and average pooling are two widely-used pooling operations, which separately return the maximum value and average value of each portion in the layer input. The combination of convolutional layer and pooling layer is usually repeated several times to capture high-level features. Finally, there are usually several fully-connected layers before the output layer.

2.1.3 Recurrent Neural Networks

A Recurrent Neural Network (RNN) is a neural network where the connections between the layers form a cycle. That is to say, the information in an RNN moves in a loop. The most important component of an RNN is its hidden state, which is updated over time. At each time step, this hidden state is updated based on the current input. The vanilla version of this updating process can be formulated as bellow:

$$h_t = f(W[h_{t-1}; x_t] + b)$$

where x_t represents the input at the current time step t, h_{t-1} and h_t separately represent the hidden state at the previous time step $t-1$ and that at the current time step t, $[;]$ represents a vector concatenation, $f(\cdot)$ represents an activation function, W is a trainable weight, and b is a trainable bias. A popular choice for $f(\cdot)$ is Tanh:

$$\mathrm{Tanh}(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

There is also an output layer in an RNN, which is usually a Softmax or Sigmoid layer. At each time step, the hidden state is passed to this output layer for a task-specific prediction. That is to say, at each time step, the output depends not only on the current input but also on the previous ones. Therefore, RNNs are able to learn the dependencies in sequential data, especially text, which makes them applicable to many NLP tasks. However, as the length of text grows, a vanilla RNN usually becomes not effective enough [22]. To

deal with long-term dependencies in text, several advanced categories of RNNs have been proposed, where the most well-known two are Long Short Term Memory networks (LSTMs) [8] and Gated Recurrent Units (GRUs) [9].

LSTMs are designed to simulate the memory mechanism of the human brain so that they can remember information for long periods of time. Unlike a vanilla RNN that only maintains a hidden state, an LSTM also maintains a cell state. The cell state and hidden state of an LSTM can be separately interpreted as the long-term memory and short-term memory of the human brain. A gating mechanism is used to update these two states at each time step, which can be formulated as bellow:

$$\text{Forget Gate: } f_t = \text{Sigmoid}(W_f[h_{t-1}; x_t] + b_f)$$

$$\text{Input Gate: } i_t = \text{Sigmoid}(W_i[h_{t-1}; x_t] + b_i)$$

$$\text{Output Gate: } o_t = \text{Sigmoid}(W_o[h_{t-1}; x_t] + b_o)$$

$$\text{New Memory: } \hat{c}_t = \text{Tanh}(W_c[h_{t-1}; x_t] + b_c)$$

$$\text{New Cell State: } c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$$

$$\text{New Hidden State: } h_t = o_t \odot \text{Tanh}(c_t)$$

where $\odot$ represents an element-wise multiplication, W_f, W_i, W_o, and W_c are trainable weights, and b_f, b_i, b_o, and b_c are trainable biases.

GRUs are simplified variants of LSTMs. This simplification is two-fold. On the one hand, the cell state and hidden state of an LSTM are merged together to form the hidden state

of a GRU. On the other hand, the above three gates in an LSTM are modified into two gates in a GRU, namely an update gate and a reset gate. Accordingly, at each time step, the updating process of a GRU is simpler than that of an LSTM:

$$\text{Update Gate: } z_t = \text{Sigmoid}(W_z[h_{t-1}; x_t] + b_z)$$

$$\text{Reset Gate: } r_t = \text{Sigmoid}(W_r[h_{t-1}; x_t] + b_r)$$

$$\text{New Memory: } \hat{h}_t = \text{Tanh}(W_h[r_t \odot h_{t-1}; x_t] + b_h)$$

$$\text{New Hidden State: } h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t$$

where W_z, W_r, and W_h are trainable weights, and b_z, b_r, and b_h are trainable biases. To the best of our knowledge, although GRUs are simpler than LSTMs, there is almost no performance gap between these two advanced categories of RNNs.

Compared with many FNNs, such as perceptrons and CNNs, RNNs are better at learning long-term dependencies in text. However, due to the recurrent architecture, the training of RNNs is based on Back-Propagation Through Time (BPTT), which means that the gradients at each time step still have gradients at the previous time steps. As a result, the training of RNNs usually consumes a lot of time and memory, and may also cause some tricky problems, such as vanishing gradient and exploding gradient.

2.1.4 Attention Mechanisms

Attention mechanisms are the neural-based approaches to guiding the interactions between the elements in a neural network, whose purpose is to promote the interactions between relevant elements and suppress those between irrelevant elements. Although there are many attention mechanisms, they can be generalized as mapping a set of key-value pairs to a set of queries, which is formulated as bellow:

$$\mathrm{Attention}(Q, K, V) = V\mathrm{Softmax}^{\top}\big(\mathrm{Similarity}(Q, K)\big)$$

where $Q \in \mathbb{R}^{d\times|Q|}$ represents the queries, $K \in \mathbb{R}^{d\times|K|}$ represents the keys, $V \in \mathbb{R}^{d\times|V|}$ represents the values, and $\mathrm{Similarity}(Q, K)$ represents calculating the similarity between each query and each key to obtain a similarity matrix of size $|Q| \times |K|$. There are two similarity calculation methods, namely the additive method [10] and the multiplicative method [11]. The additive method is formulated as bellow:

$$\mathrm{Similarity}_{i,j}(Q, K) = v_a^{\top}\mathrm{Tanh}(W_a[q_i; k_j])$$

where q_i represents the i-th query, k_j represents the j-th key, v_a is a trainable vector, and W_a is a trainable matrix. The multiplicative method is simpler:

$$\mathrm{Similarity}_{i,j}(Q, K) = q_i^{\top}W_m k_j$$

where W_m is a trainable matrix. The above two methods usually achieve similar performance when the dimensionality is small. However, when the dimensionality is large, the

additive method usually outperforms the multiplicative method.

Attention mechanisms were originally applied in sequence-to-sequence models consisting of an RNN encoder and an RNN decoder. In such models, to learn the dependencies from the decoder inputs to the encoder inputs, attention is performed in the decoder at each decoding time step by using the previous decoder hidden state as the query and all the encoder hidden states as the keys and the values, and the attention result is used together with the current decoder input to update the decoder hidden state. So far, attention mechanisms have also been applied in many other models, where the most significant breakthrough is their application in Transformers [12]. A Transformer is a sequence-to-sequence model consisting of a multi-layer FNN encoder and a multi-layer FNN decoder, and it is mainly based on the following three attention mechanisms:

- **Cross attention.** To learn the dependencies from the decoder inputs to the encoder inputs, cross attention is performed in each decoder layer by using the layer hidden states as the queries and the final encoder hidden states as the keys and the values, and the attention results are added to the layer hidden states.

- **Self attention.** To learn the dependencies from the encoder inputs to themselves, self attention is performed in each encoder layer by using the layer hidden states as the queries, the keys, and the values, and the attention results are added to the layer hidden states. Self attention is also performed in each decoder layer, but the

similarity matrix obtained in each decoder layer is partially masked to guarantee that each time step is only affected by its previous time steps.

- **Multi-head attention.** Multi-head attention is to perform cross attention and self attention for multiple times in parallel, where each time is called an attention head and expected to focus on learning a different type of dependencies in text. In each attention head, the queries, the keys, and the values are separately converted from their original version through a linear transformation. The results obtained from all the attention heads are concatenated together, and the concatenation result is converted to the attention result through a linear transformation.

Due to the application of the above three attention mechanisms, Transformers possess strong capability on learning long-term dependencies in text. By the way, since Transformers are completely feed-forward in architecture, the training of Transformers consumes much less time and memory than that of RNNs.

2.2 Neural-based Knowledge Transfer in NLP

2.2.1 Knowledge Base Encoding

Knowledge bases, which originate from expert systems, are aimed at storing knowledge in structured formats. The knowledge stored in a knowledge base is explicitly organized as "subject-predicate-object" triples, where each triple implies that the subject is related to the

object through the predicate. Therefore, by representing subjects and objects as vertices and predicates as edges, the knowledge stored in a knowledge base can be expressed as a graph. There have been many knowledge bases, which store various knowledge. For example, WordNet [13] stores semantic knowledge, ConceptNet [14] stores commonsense knowledge, Freebase [15] stores factoid knowledge, and so on. By applying graph-based information retrieval techniques, the knowledge stored in a knowledge base can be directly utilized to answer questions, such as "Where is the capital of the United States?", which is known as Knowledge Base Question Answering (KBQA). Actually, the knowledge stored in a knowledge base can not only be utilized to address KBQA, but also to facilitate many other NLP tasks that do not require knowledge bases, such as Machine Reading Comprehension (MRC), where a machine is required to answer a question according to a relevant context. However, in the later case, the knowledge stored in a knowledge base cannot be directly utilized as in KBQA, but needs to be transferred as background knowledge into task-specific neural networks. This goal is usually achieved by applying knowledge base encoding approaches, which is to encode the knowledge concepts (i.e. subjects, objects, and predicates) in a knowledge base into distributed representations, and thereby use the resulting knowledge representations to enhance the language representations in task-specific neural networks. For example, to facilitate MRC, a word embedding refinement approach [23] first retrieves from ConceptNet the triples relevant to each given context-question pair and transforms them into text-form assertions,

then passes the assertions through a BiLSTM-based encoding layer, and finally uses the encoding outputs to update the corresponding word embeddings. A knowledge memory approach [24] is similar to the above approach, but it encodes each triple-based assertion into a key-value memory. On this basis, it performs attention on the memories related to each word, where the word representation is used as the query, and thereby merges the attention result into the word representation. An entity description encoding approach [25] first extracts from Freebase the text-form entity descriptions of the entities existing in each given context-question pair, then passes the entity descriptions through an LSTM-based encoding layer, and finally uses the last encoding output for each entity description as the representation of the corresponding entity.

2.2.2 Pre-trained Language Models

Large-scale corpora, such as the Wikipedia dumps, can be seen as a special kind of knowledge bases, where the stored knowledge is implicitly embodied in free text. Due to the linguistic diversity of free text, it will be beneficial if the knowledge stored in a large-scale corpus is utilized to facilitate NLP tasks. To this end, researchers usually pre-train a language model on a large-scale corpus, and thereby use the pre-trained language model to generate contextualized word representations in task-specific neural networks. There have been many pre-trained language models, such as:

- **Context Vectors (CoVe).** CoVe is a Machine Translation (MT) model, which

can be seen as a cross-lingual language model, and it consists of a two-layer BiLSTM encoder and a two-layer attention-based LSTM decoder. The pre-training task of CoVe is to translate English sentences into German sentences, where the training corpus between the two languages is collected from the corresponding WMT datasets. When applied to downstream NLP tasks, CoVe drops the decoder and outputs the final hidden states of the encoder as the contextualized word representations in the task-specific neural networks.

- **Embeddings from Language Models (ELMo).** ELMo is a three-layer bidirectional language model, where the bottom layer is a character-level CNN and the upper two layers are bidirectional LSTMs (BiLSTMs). The pre-training task of ELMo is to separately predict the next word and the previous word at each time step based on the output of the top-layer forward LSTM and that of the top-layer backward LSTM, where the training corpus is collected from 1 Billion Word Language Model Benchmark. When applied to downstream NLP tasks, ELMo first calculates a position-wise weighted sum on the hidden states of all its layers, then scales the sum by a scalar value, and finally outputs the scaling result as the contextualized word representations in the task-specific neural networks.

- **Generative Pre-training for Transformers (GPT).** GPT is a unidirectional language model implemented as a Transformer decoder. The input to GPT is required

to be a word sequence that starts with $\langle s \rangle$ and ends with $\langle e \rangle$. The input word sequence is allowed to consist of multiple sub-sequences, and if so, each sub-sequence is required to be separated from the next one by $\$$. The pre-training task of GPT is to predict the next word based on the final hidden state for each word, where the training corpus is collected from BooksCorpus and 1 Billion Word Language Model Benchmark. When applied to downstream NLP tasks, GPT converts each example to its required format, and outputs the final hidden states as the contextualized word representations in the task-specific neural networks.

- **Bidirectional Encoder Representations from Transformers (BERT).** BERT is a bidirectional language model implemented as a Transformer encoder. The input to BERT is required to be a word sequence that starts with $[CLS]$ and consists of two sub-sequences, with each sub-sequence ending with $[SEP]$. The pre-training task of BERT is a combo of two sub-tasks. The first sub-task is "masked words prediction", which is to predict some randomly masked words in the input word sequence based on the final hidden states for the unmasked words. The second sub-task is "next sentence prediction", which is to determine whether the two sub-sequences are consecutive or not based on the final hidden state for $[CLS]$. The training corpus used for the pre-training of BERT is collected from BooksCorpus and Wikipedia. When applied to downstream NLP tasks, BERT converts each example to its required format, and outputs the final hidden states as the contextualized word

representations in the task-specific neural networks.

In downstream NLP tasks, the contextualized word representations generated by pre-trained language models can be used in two manners. On the one hand, for all pre-trained language models, their generated contextualized word representations can be used in a feature-based manner. Specifically, the contextualized word representations are only used as additional features to the word embeddings in the input layer of a task-specific neural network, which does not affect the design for the upper layers. On the other hand, for the Transformer-based pre-trained language models, such as GPT and BERT, their generated contextualized word representations can also be used in a fine-tuning manner. Specifically, the contextualized word representations are directly passed to the output layer in a task-specific neural network for final predictions, thus training this neural network is actually fine-turning the pre-trained language model.

2.3 Related NLP Tasks

2.3.1 Machine Reading Comprehension (MRC)

MRC requires a machine to read a context and answer a relevant question based on its comprehension to the context. According to the style of the answer, the existing MRC tasks can be divided into four categories:

- **Multi-choice MRC.** Each context-question pair has multiple candidate answers,

and only one of them is the correct answer. Examples include MCTest [26], WikiQA [27], and WikiHop [28].

- **Cloze-test MRC.** Each context contains some blanks, where each blank can be seen as a question, and its answer is a word or entity from a given vocabulary. Examples include CNN/DailyMail [29] and Children's Book Test [30].

- **Span-extraction MRC.** For each given context-question pair, the answer is a span (i.e. a start position and an end position) in the context. Examples include SQuAD [31], NewsQA [32], SearchQA [33], TriviaQA [34], and HotpotQA [2].

- **Text-generation MRC.** For each given context-question pair, the answer is a generated snippet of text. Examples include MS MARCO [35], DuReader [36], and NarrativeQA [37].

Besides, according to the number of passages included in each context and the number of passages involved in the question answering process, the existing MRC tasks can also be divided into three categories, each of which is at a different level of complexity. The simplest category is single-passage MRC tasks, such as MCTest, WikiQA, CNN/DailyMail, Children's Book Test, SQuAD, and NewsQA, where each context includes only a single passage. The more complex category is multi-passage single-hop MRC tasks, such as MS MARCO, DuReader, NarrativeQA, SearchQA, and TriviaQA, where each context includes multiple passages, while the question can be answered by performing single-hop

Sentence	show	flights	from	Toronto	to	Las	Vegas	today
Slots	O	O	O	B-dept	O	B-arr	I-arr	B-date
Intent	find_flight							

Table 2.1: NLU: slot filling and intent classification

reasoning on only one of them. The most complex category is multi-hop MRC tasks, such as HotpotQA and WikiHop, where each context includes multiple passages, and the question needs to be answered by performing multi-hop reasoning on at least two of them.

2.3.2 Natural Language Understanding (NLU)

NLU is aimed at understanding the structure and meaning of natural languages so that human beings can interact with computers. Although NLU is a very broad topic in NLP, in this dissertation, we focus on the most fundamental two components of it, namely slot filling and intent classification, which are also the most important two functions of goal-oriented dialogue systems or personal assistants. As shown in Table 2.1, given a sentence, intent classification is to detect the intent of the sentence, and slot filling is to extract the arguments for the intent. There have been many NLU tasks for slot filling and intent classification, where the most well-known task is ATIS [38]. The sentences in ATIS are in English, and they are converted from audio recordings about flight reservation. Besides, there have also been several multilingual extensions to ATIS. For example, Multilingual

ATIS [39] extends ATIS to Turkish and Hindi, and MultiATIS++ [40] further extends Multilingual ATIS to Spanish, German, French, Portuguese, Chinese, and Japanese.

3 Explicit Utilization of General Knowledge in Machine Reading Comprehension

3.1 Introduction

Machine Reading Comprehension (MRC), as its name suggests, requires a machine to read a context and answer a relevant question based on its comprehension to the context. In this chapter, we focus on single-passage MRC tasks, where each context includes only a single passage. Since the answer to each question is supposed to stem from the corresponding passage, a common MRC solution is to develop a neural-based MRC model that predicts an answer span (i.e. the answer start position and the answer end position) from the passage of each given passage-question pair. To facilitate the explorations and innovations in this area, many MRC datasets have been established, such as SQuAD [31], MS MARCO [35], and TriviaQA [34]. Consequently, many pioneering MRC models have been proposed, such as BiDAF [41], R-NET [42], and QANet [43]. According to the leader board of SQuAD, the state-of-the-art MRC models have achieved the same

Passage	**Question**	**Answer**
Teachers may use a lesson plan to **facilitate** student learning, providing a course of study which is called the curriculum.	What can a teacher use to **help** students learn?	lesson plan
Manufacturing accounts for a significant but declining share of employment, although the city's garment industry is showing a resurgence in **Brooklyn**.	In what **borough** is the garment business prominent?	Brooklyn

Table 3.1: Two examples about the importance of inter-word semantic connections to the reading comprehension ability of human beings: in the first one, we can find the answer because we know "facilitate" is a synonym of "help"; in the second one, we can find the answer because we know "Brooklyn" is a hyponym of "borough".

performance as human beings. However, does this imply that they have possessed the same reading comprehension ability as human beings?

OF COURSE NOT. There is a huge gap between MRC models and human beings, which is mainly reflected in the hunger for data and the robustness to noise. On the one hand, developing MRC models requires a large amount of training examples (i.e. the passage-question pairs labeled with answer spans), while human beings can achieve good performance on evaluation examples (i.e. the passage-question pairs to address) without training examples. On the other hand, [44] revealed that intentionally injected noise (e.g. misleading sentences) in evaluation examples causes the performance of MRC models to drop significantly, while human beings are far less likely to suffer from this. The reason for these phenomena, we believe, is that MRC models can only utilize the knowledge contained in each given passage-question pair, but in addition to this, human beings can also utilize general knowledge. A typical category of general knowledge is inter-word semantic connections. As shown in Table 3.1, such general knowledge is essential to the reading comprehension ability of human beings.

A promising strategy to bridge the gap mentioned above is to integrate the neural networks of MRC models with the general knowledge of human beings. To this end, it is necessary to solve two problems: extracting general knowledge from passage-question pairs and utilizing the extracted general knowledge in the prediction of answer spans. The first problem can be solved with knowledge bases, which store general knowledge in structured

forms. A broad variety of knowledge bases are available, such as WordNet [13] storing semantic knowledge, ConceptNet [14] storing commonsense knowledge, and Freebase [15] storing factoid knowledge. In this chapter, we limit the scope of general knowledge to inter-word semantic connections, and thus use WordNet as our knowledge base. The existing way to solve the second problem is to encode general knowledge in vector space so that the encoding results can be used to enhance the lexical or contextual representations of words [23, 24]. However, this is an implicit way to utilize general knowledge, since in this way we can neither understand nor control the functioning of general knowledge. In this chapter, we discard the existing implicit way and instead explore an explicit (i.e. understandable and controllable) way to utilize general knowledge.

The contribution of this chapter is two-fold. On the one hand, we propose a data enrichment method, which uses WordNet to extract inter-word semantic connections as general knowledge from each given passage-question pair. On the other hand, we propose a novel MRC model named Knowledge Aided Reader (KAR), which explicitly uses the above extracted general knowledge to assist its attention mechanisms. According to the experimental results, KAR is comparable in performance with the state-of-the-art MRC models, and significantly more robust to noise than them. On top of that, when only a subset (20%–80%) of the training examples are available, KAR outperforms the state-of-the-art MRC models by a large margin, and is still reasonably robust to noise.

3.2 Related Works

3.2.1 Attention Mechanisms for MRC Models

The key components of many existing MRC models are their attention mechanisms [10], which are aimed at integrating the representations of associated words in one abstraction level so as to generate the representations in the next abstraction level. These attention mechanisms generally fall into two categories, namely mutual attention and self attention. Mutual attention, which was originally proposed in BiDAF [41] and has been adopted in several other MRC models [43, 45, 46, 47, 48, 42], is aimed at fusing the question representations into the passage representations so as to generate the question-aware passage representations. Self attention, which was originally proposed in R-NET [42] and and has been adopted in several other MRC models [45, 49, 47], is aimed at fusing the question-aware passage representations into themselves so as to generate the final passage representations. Besides, to effectively capture the complicated interactions among the words in each given context-question pair, the above two attention mechanisms can also be performed repeatedly, which is known as multi-round attention. For example, FusionNet [49] and DCN+ [48], implement multi-round attention by using the word representations output by the previous attention round (i.e. mutual-matching attention followed by self-matching attention) as the inputs to the current attention round. However, since different attention rounds share attentive information only through word representations, it is easy

to cause attention redundancy and attention deficiency. For this reason, R.M-Reader [46] directly uses the attentive information obtained in the previous attention round to refine the attentive information in the current attention round.

3.2.2 Performance-boosting Approaches for MRC Models

Linguistic representations. It is both simple and effective to incorporate linguistic representations into the word representations of MRC models. For example, DrQA [50] and SAN [47] first perform Part-of-Speech (POS) tagging and Named Entity Recognition (NER) on each given passage-question pair, and thereby extend their word embeddings with POS embeddings and entity embeddings. Besides, SEST [51] first constructs a constituency tree and a dependency tree on each given passage-question pair, and thereby extends its word embeddings with structural embeddings of the tree nodes.

Multi-round reasoning. Given a difficult passage, human beings can finally understand it by reading it over and over again. This strategy can also be implemented to better address difficult MRC tasks, which is known as multi-round reasoning. For example, ReasoNet [52] uses reinforcement learning to dynamically determine the number of reasoning rounds. Besides, SAN [47] fixes the number of reasoning rounds but uses stochastic dropout in the output layer to avoid step bias.

Reinforcement learning. To reward the answers that are textually similar to but positionally different from the golden answer, a reinforcement learning loss, which is measured

according to the overlap between the predicted answer and the golden answer, can be combined with the traditional cross entropy loss to form a joint loss to optimize. This strategy was originally proposed in DCN+ [48] and has been adopted in R.M-Reader [46]. Besides, SLQA+ [53] optimizes the traditional cross entropy loss in its pre-training stage and the joint loss in its fine-tuning stage.

Data augmentation. Augmenting training examples can improve the performance of MRC models. For example, GDANs [54] uses a separate generative model to generate questions based on unlabeled text, which substantially improves its performance. Besides, QANet [43] uses a separate back-and-forth translation model to paraphrase training examples, which brings its performance a significant gain.

3.3 Data Enrichment Method

In this section, we elaborate a WordNet-based data enrichment method, which is aimed at extracting inter-word semantic connections from each passage-question pair in our MRC dataset. The extraction is performed in a controllable manner, and the extracted results are provided as general knowledge to our MRC model.

3.3.1 Semantic Relation Chain

WordNet is a lexical database of English, where words are organized into synsets according to their senses. A synset is a set of words expressing the same sense so that a word having

multiple senses belongs to multiple synsets, with each synset corresponding to a sense. Synsets are further related to each other through semantic relations. According to the WordNet interface provided by NLTK [55], there are totally sixteen types of semantic relations (e.g. hypernyms, hyponyms, holonyms, meronyms, attributes, etc.). Based on synset and semantic relation, we define a new concept: semantic relation chain. A semantic relation chain is a concatenated sequence of semantic relations, which links a synset to another synset. For example, the synset "keratin.n.01" is related to the synset "feather.n.01" through the semantic relation "substance holonym", the synset "feather.n.01" is related to the synset "bird.n.01" through the semantic relation "part holonym", and the synset "bird.n.01" is related to the synset "parrot.n.01" through the semantic relation "hyponym", thus "substance holonym $\rightarrow$ part holonym $\rightarrow$ hyponym" is a semantic relation chain, which links the synset "keratin.n.01" to the synset "parrot.n.01". We name each semantic relation in a semantic relation chain as a hop, therefore the above semantic relation chain is a 3-hop chain. By the way, each single semantic relation is equivalent to a 1-hop chain.

3.3.2 Inter-word Semantic Connection

The key problem in the data enrichment method is determining whether a word is semantically connected to another word. If so, we say that there exists an inter-word semantic connection between them. To solve this problem, we define another new concept: the

extended synsets of a word. Given a word w, whose synsets are represented as a set S_w, we use another set S_w^* to represent its extended synsets, which includes all the synsets that are in S_w or that can be linked to from S_w through semantic relation chains. Theoretically, if there is no limitation on semantic relation chains, S_w^* will include all the synsets in WordNet, which is meaningless in most situations. Therefore, we use a hyper-parameter $\kappa \in \mathbb{N}$ to represent the permitted maximum hop count of semantic relation chains. That is to say, only the chains having no more than κ hops can be used to construct S_w^* so that S_w^* becomes a function of κ: $S_w^*(\kappa)$ (if $\kappa = 0$, we will have $S_w^*(0) = S_w$). Based on the above statements, we formulate a heuristic rule for determining inter-word semantic connections: a word w_1 is semantically connected to another word w_2 if and only if $S_{w_1}^*(\kappa) \cap S_{w_2} \neq \emptyset$.

3.3.3 General Knowledge Extraction

Given a passage-question pair, the inter-word semantic connections that connect any word to any passage word are regarded as the general knowledge we need to extract. Considering the requirements of our MRC model, we only extract the positional information of such inter-word semantic connections. Specifically, for each word w, we extract a set E_w, which includes the positions of the passage words that w is semantically connected to (if w itself is a passage word, we will exclude its own position from E_w). We can control the amount of the extracted results by setting the hyper-parameter κ: if we set κ to 0, inter-word semantic connections will only exist between synonyms; if we increase κ, inter-

word semantic connections will exist between more words. That is to say, by increasing κ within a certain range, we can usually extract more inter-word semantic connections from a passage-question pair, and thus can provide the MRC model with more general knowledge. However, due to the complexity and diversity of natural languages, only a part of the extracted results can serve as useful general knowledge, while the rest of them are useless for the prediction of answer spans, and the proportion of the useless part always rises when κ is set larger. Therefore we set κ through cross validation (i.e. according to the performance of the MRC model on the development examples).

3.4 Knowledge Aided Reader

In this section, we elaborate our MRC model: Knowledge Aided Reader (KAR). Just like many existing MRC models, KAR is equipped with attention mechanisms, including both mutual attention and self attention. However, the most remarkable feature of KAR is that it explicitly uses the general knowledge extracted by the data enrichment method to assist its attention mechanisms. Therefore, we separately name the attention mechanisms of KAR as knowledge aided mutual attention and knowledge aided self attention.

3.4.1 Task Definition

Given a passage $P = \{p_1, \ldots, p_n\}$ and a relevant question $Q = \{q_1, \ldots, q_m\}$, the task is to predict an answer span $[a_s, a_e]$, where $1 \leq a_s \leq a_e \leq n$, so that the resulting

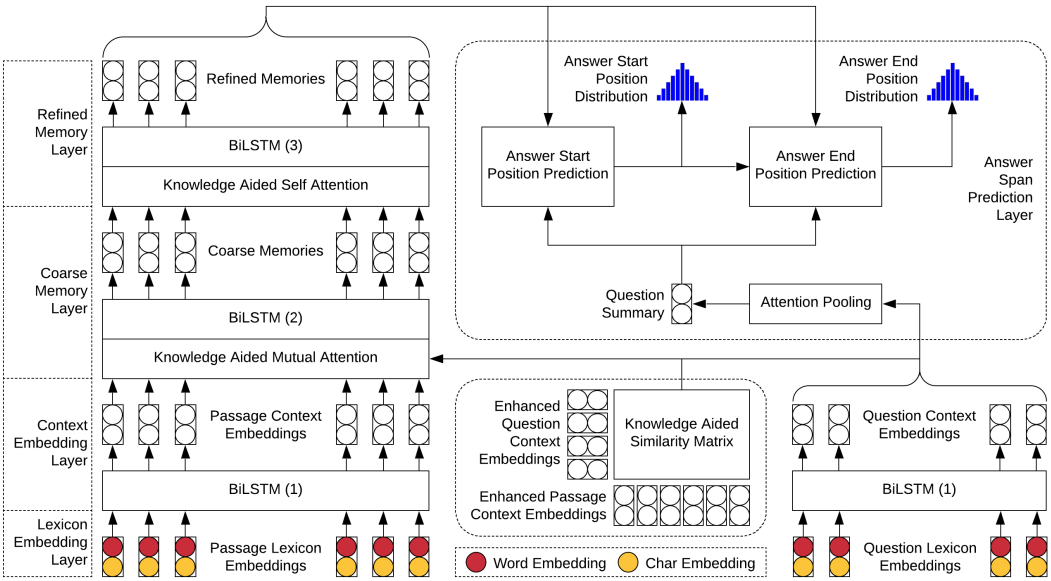

Figure 3.1: An end-to-end MRC model: Knowledge Aided Reader (KAR)

subsequence $\{p_{a_s}, \ldots, p_{a_e}\}$ from P is an answer to Q.

3.4.2 Overall Architecture

As shown in Figure 3.1, KAR is an end-to-end MRC model consisting of five layers:

Lexicon Embedding Layer. This layer maps the words to the lexicon embeddings. The lexicon embedding of each word is composed of its word embedding and character embedding. For each word, we use the pre-trained GloVe [56] word vector as its word embedding, and obtain its character embedding with a Convolutional Neural Network (CNN) [57]. For both the passage and the question, we pass the concatenation of the word embeddings and the character embeddings through a shared dense layer with ReLU

activation, whose output dimensionality is d. Therefore we obtain the passage lexicon embeddings $L_P \in \mathbb{R}^{d \times n}$ and the question lexicon embeddings $L_Q \in \mathbb{R}^{d \times m}$.

Context Embedding Layer. This layer maps the lexicon embeddings to the context embeddings. For both the passage and the question, we process the lexicon embeddings (i.e. L_P for the passage and L_Q for the question) with a shared bidirectional LSTM (BiLSTM) [8], whose hidden state dimensionality is $\frac{1}{2}d$. By concatenating the forward LSTM outputs and the backward LSTM outputs, we obtain the passage context embeddings $C_P \in \mathbb{R}^{d \times n}$ and the question context embeddings $C_Q \in \mathbb{R}^{d \times m}$.

Coarse Memory Layer. This layer maps the context embeddings to the coarse memories. First we use knowledge aided mutual attention (introduced later) to fuse C_Q into C_P, the outputs of which are represented as $\tilde{G} \in \mathbb{R}^{d \times n}$. Then we process $\tilde{G}$ with a BiLSTM, whose hidden state dimensionality is $\frac{1}{2}d$. By concatenating the forward LSTM outputs and the backward LSTM outputs, we obtain the coarse memories $G \in \mathbb{R}^{d \times n}$, which are the question-aware passage representations.

Refined Memory Layer. This layer maps the coarse memories to the refined memories. First we use knowledge aided self attention (introduced later) to fuse G into themselves, the outputs of which are represented as $\tilde{H} \in \mathbb{R}^{d \times n}$. Then we process $\tilde{H}$ with a BiLSTM, whose hidden state dimensionality is $\frac{1}{2}d$. By concatenating the forward LSTM outputs and the backward LSTM outputs, we obtain the refined memories $H \in \mathbb{R}^{d \times n}$, which are the final passage representations.

Answer Span Prediction Layer. This layer predicts the answer start position and the answer end position based on the above layers. First we obtain the answer start position distribution o_s:

$$t_i = v_s^\top \tanh(W_s h_{p_i} + U_s r_Q) \in \mathbb{R}$$

$$o_s = \mathrm{softmax}(\{t_1, \ldots, t_n\}) \in \mathbb{R}^n$$

where v_s, W_s, and U_s are trainable parameters; h_{p_i} represents the refined memory of each passage word p_i (i.e. the i-th column in H); r_Q represents the question summary obtained by performing an attention pooling over C_Q. Then we obtain the answer end position distribution o_e:

$$t_i = v_e^\top \tanh(W_e h_{p_i} + U_e[r_Q; Ho_s]) \in \mathbb{R}$$

$$o_e = \mathrm{softmax}(\{t_1, \ldots, t_n\}) \in \mathbb{R}^n$$

where v_e, W_e, and U_e are trainable parameters; $[;]$ represents vector concatenation. Finally we construct an answer span prediction matrix $O = \mathrm{uptri}(o_s o_e^\top) \in \mathbb{R}^{n \times n}$, where $\mathrm{uptri}(X)$ represents the upper triangular matrix of a matrix X. Therefore, for the training, we minimize $-\log(O_{a_s,a_e})$ on each training example whose labeled answer span is $[a_s, a_e]$; for the inference, we separately take the row index and column index of the maximum element in O as a_s and a_e.

3.4.3 Knowledge Aided Mutual Attention

As a part of the coarse memory layer, knowledge aided mutual attention is aimed at fusing the question context embeddings C_Q into the passage context embeddings C_P, where the key problem is to calculate the similarity between each passage context embedding c_{p_i} (i.e. the i-th column in C_P) and each question context embedding c_{q_j} (i.e. the j-th column in C_Q). To solve this problem, [41] proposed a similarity function:

$$f(c_{p_i}, c_{q_j}) = v_f^\top [c_{p_i}; c_{q_j}; c_{p_i} \odot c_{q_j}] \in \mathbb{R}$$

where v_f is a trainable parameter; $\odot$ represents element-wise multiplication. This similarity function has also been adopted by several other works [43, 45]. However, since context embeddings contain high-level information, we believe that introducing the pre-extracted general knowledge into the calculation of such similarities will make the results more reasonable. Therefore we modify the above similarity function to the following form:

$$f^*(c_{p_i}, c_{q_j}) = v_f^\top [c^*_{p_i}; c^*_{q_j}; c^*_{p_i} \odot c^*_{q_j}] \in \mathbb{R}$$

where c^*_x represents the enhanced context embedding of a word x. We use the pre-extracted general knowledge to construct the enhanced context embeddings. Specifically, for each word w, whose context embedding is c_w, to construct its enhanced context embedding c^*_w, first recall that we have extracted a set E_w, which includes the positions of the passage words that w is semantically connected to, thus by gathering the columns in C_P whose

indexes are given by E_w, we obtain the matching context embeddings $Z \in \mathbb{R}^{d \times |E_w|}$. Then by constructing a c_w-attended summary of Z, we obtain the matching vector c_w^+ (if $E_w = \emptyset$, which makes $Z = \{\}$, we will set $c_w^+ = 0$):

$$t_i = v_c^\top \tanh(W_c z_i + U_c c_w) \in \mathbb{R}$$

$$c_w^+ = Z \, \mathrm{softmax}(\{t_1, \ldots, t_{|E_w|}\}) \in \mathbb{R}^d$$

where v_c, W_c, and U_c are trainable parameters; z_i represents the i-th column in Z. Finally we pass the concatenation of c_w and c_w^+ through a dense layer with ReLU activation, whose output dimensionality is d. Therefore we obtain the enhanced context embedding $c_w^* \in \mathbb{R}^d$.

Based on the modified similarity function and the enhanced context embeddings, to perform knowledge aided mutual attention, first we construct a knowledge aided similarity matrix $A \in \mathbb{R}^{n \times m}$, where each element $A_{i,j} = f^*(c_{p_i}, c_{q_j})$. Then following [43], we construct the passage-attended question summaries R_Q and the question-attended passage summaries R_P:

$$R_Q = C_Q \, \mathrm{softmax}_r^\top(A) \in \mathbb{R}^{d \times n}$$

$$R_P = C_P \, \mathrm{softmax}_c(A) \, \mathrm{softmax}_r^\top(A) \in \mathbb{R}^{d \times n}$$

where $\mathrm{softmax}_r$ represents softmax along the row dimension and $\mathrm{softmax}_c$ along the column dimension. Finally following [45], we pass the concatenation of C_P, R_Q, $C_P \odot R_Q$,

and $R_P \odot R_Q$ through a dense layer with ReLU activation, whose output dimensionality is d. Therefore we obtain the outputs $\tilde{G} \in \mathbb{R}^{d \times n}$.

3.4.4 Knowledge Aided Self Attention

As a part of the refined memory layer, knowledge aided self attention is aimed at fusing the coarse memories G into themselves. If we simply follow the self attentions of other works [42, 45, 49, 47], then for each passage word p_i, we should fuse its coarse memory g_{p_i} (i.e. the i-th column in G) with the coarse memories of all the other passage words. However, we believe that this is both unnecessary and distracting, since each passage word has nothing to do with many of the other passage words. Thus we use the pre-extracted general knowledge to guarantee that the fusion of coarse memories for each passage word will only involve a precise subset of the other passage words. Specifically, for each passage word p_i, whose coarse memory is g_{p_i}, to perform the fusion of coarse memories, first recall that we have extracted a set E_{p_i}, which includes the positions of the other passage words that p_i is semantically connected to, thus by gathering the columns in G whose indexes are given by E_{p_i}, we obtain the matching coarse memories $Z \in \mathbb{R}^{d \times |E_{p_i}|}$. Then by constructing a g_{p_i}-attended summary of Z, we obtain the matching vector $g_{p_i}^+$ (if $E_{p_i} = \emptyset$, which makes $Z = \{\}$, we will set $g_{p_i}^+ = 0$):

$$t_i = v_g^\top \tanh(W_g z_i + U_g g_{p_i}) \in \mathbb{R}$$

$$g_{p_i}^+ = Z \operatorname{softmax}(\{t_1, \ldots, t_{|E_{p_i}|}\}) \in \mathbb{R}^d$$

where v_g, W_g, and U_g are trainable parameters. Finally we pass the concatenation of g_{p_i} and $g_{p_i}^+$ through a dense layer with ReLU activation, whose output dimensionality is d. Therefore we obtain the fusion result $\tilde{h}_{p_i} \in \mathbb{R}^d$, and further the outputs $\tilde{H} = \{\tilde{h}_{p_1}, \ldots, \tilde{h}_{p_n}\} \in \mathbb{R}^{d \times n}$.

3.5 Experiments

3.5.1 Experimental Settings

MRC Dataset. The MRC dataset used in this chapter is SQuAD 1.1, which contains over $100,000$ passage-question pairs and has been randomly partitioned into three parts: a training set (80%), a development set (10%), and a test set (10%). Besides, we also use two of its adversarial sets, namely AddSent and AddOneSent [44], to evaluate the robustness to noise of MRC models. The passages in the adversarial sets contain misleading sentences, which are aimed at distracting MRC models. Specifically, each passage in AddSent contains several sentences that are similar to the question but not contradictory to the answer, while each passage in AddOneSent contains a human-approved random sentence that may be unrelated to the passage.

Implementation Details. We tokenize the MRC dataset with spaCy [58], manipulate WordNet with NLTK, and implement KAR with TensorFlow [59]. For the data enrichment method, we set the hyper-parameter κ to 3. For the dense layers and the BiLSTMs, we

set the dimensionality unit d to 600. For model optimization, we apply the Adam [60] optimizer with a learning rate of 0.0005 and a mini-batch size of 32. For model evaluation, we use Exact Match (EM) and F1 score as evaluation metrics. To avoid overfitting, we apply dropout [61] to the dense layers and the BiLSTMs with a dropout rate of 0.3. To boost the performance, we apply exponential moving average with a decay rate of 0.999.

3.5.2 Model Comparison in both Performance and the Robustness to Noise

We compare KAR with other MRC models in both performance and the robustness to noise. Specifically, we not only evaluate the performance of KAR on the development set and the test set, but also do this on the adversarial sets. As for the comparative objects, we only consider the single MRC models that rank in the top 20 on the SQuAD 1.1 leader board and have reported their performance on the adversarial sets. There are totally five such comparative objects, which can be considered as representatives of the state-of-the-art MRC models. As shown in Table 3.2, on the development set and the test set, the performance of KAR is on par with that of the state-of-the-art MRC models; on the adversarial sets, KAR outperforms the state-of-the-art MRC models by a large margin. That is to say, KAR is comparable in performance with the state-of-the-art MRC models, and significantly more robust to noise than them.

To verify the effectiveness of general knowledge, we first study the relationship between the amount of general knowledge and the performance of KAR. As shown in Table 3.3, by

Single MRC model	**Dev set (EM / F1)**	**Test set (EM / F1)**	**AddSent (F1)**	**AddOneSent (F1)**
FusionNet [49]	75.3 / 83.6	76.0 / 83.9	51.4	60.7
RaSoR+TR+LM [62]	77.0 / 84.0	77.6 / 84.2	47.0	57.0
SAN [47]	76.2 / 84.1	76.8 / 84.4	46.6	56.5
R.M-Reader [46]	**78.9 / 86.3**	79.5 / 86.6	58.5	67.0
QANet (with data augmentation) [43]	75.1 / 83.8	**82.5 / 89.3**	45.2	55.7
KAR (ours)	76.7 / 84.9	76.1 / 83.5	**60.1**	**72.3**

Table 3.2: Model comparison based on SQuAD 1.1 and two of its adversarial sets: AddSent and AddOneSent. All the numbers are up to date as of October 18, 2018. Note that SQuAD 2.0 [1] is not involved in this chapter, because it requires MRC models to deal with the problem of answer triggering, but this chapter is aimed at improving the hunger for data and robustness to noise of MRC models.

κ	**Average number of inter-word semantic connections per word**	**Dev set (EM / F1)**
0	0.39	74.2 / 82.8
1	0.63	74.6 / 83.1
2	1.24	75.1 / 83.5
3	**2.21**	**76.7 / 84.9**
4	3.68	75.9 / 84.3
5	5.58	75.3 / 83.8

Table 3.3: With κ set to different values in the data enrichment method, we calculate the average number of inter-word semantic connections per word as an estimation of the amount of general knowledge, and evaluate the performance of KAR on the development set.

increasing κ from 0 to 5 in the data enrichment method, the amount of general knowledge rises monotonically, but the performance of KAR first rises until κ reaches 3 and then drops down. Then we conduct an ablation study by replacing the knowledge aided attention mechanisms with the mutual attention proposed by [41] and the self attention proposed by [42] separately, and find that the F1 score of KAR drops by 4.2 on the development set, 7.8 on AddSent, and 9.1 on AddOneSent. Finally we find that after only one epoch of training, KAR already achieves an EM of 71.9 and an F1 score of 80.8 on the development set, which is even better than the final performance of several strong baselines, such as DCN (EM / F1: 65.4 / 75.6) [48] and BiDAF (EM / F1: 67.7 / 77.3) [41]. The above empirical findings imply that general knowledge indeed plays an effective role in KAR.

To demonstrate the advantage of our explicit way to utilize general knowledge over the existing implicit way, we compare the performance of KAR with that reported by [23], which used an encoding-based method to utilize the general knowledge dynamically retrieved from Wikipedia and ConceptNet. Since their best model only achieved an EM of 69.5 and an F1 score of 79.7 on the development set, which is much lower than the performance of KAR, we have good reason to believe that our explicit way works better than the existing implicit way.

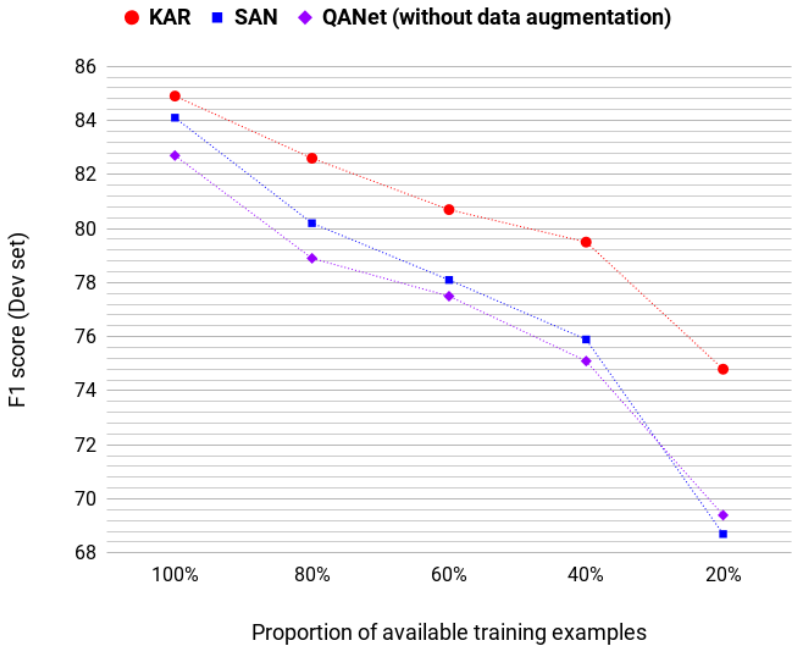

Figure 3.2: With KAR, SAN, and QANet (without data augmentation) trained on the training subsets, we evaluate their performance on the development set.

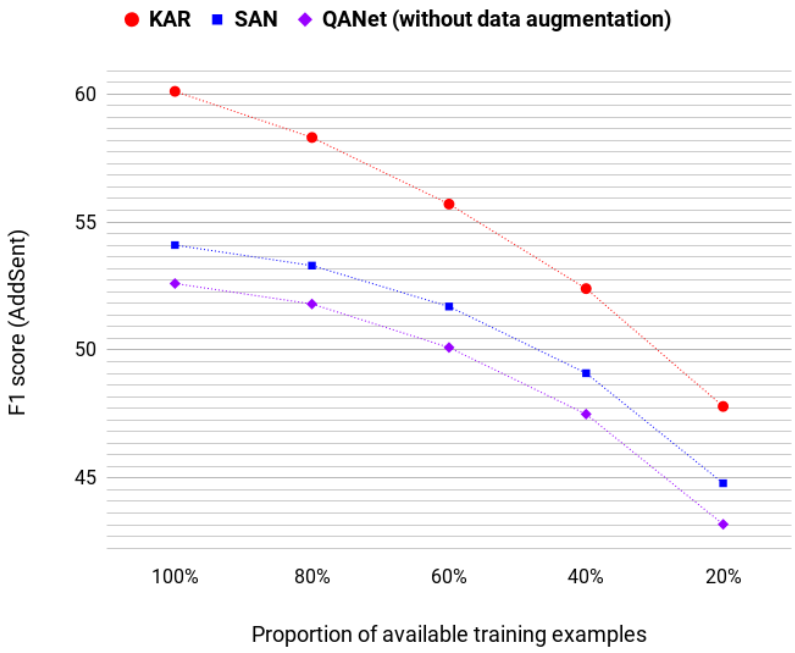

Figure 3.3: With KAR, SAN, and QANet (without data augmentation) trained on the training subsets, we evaluate their performance on AddSent.

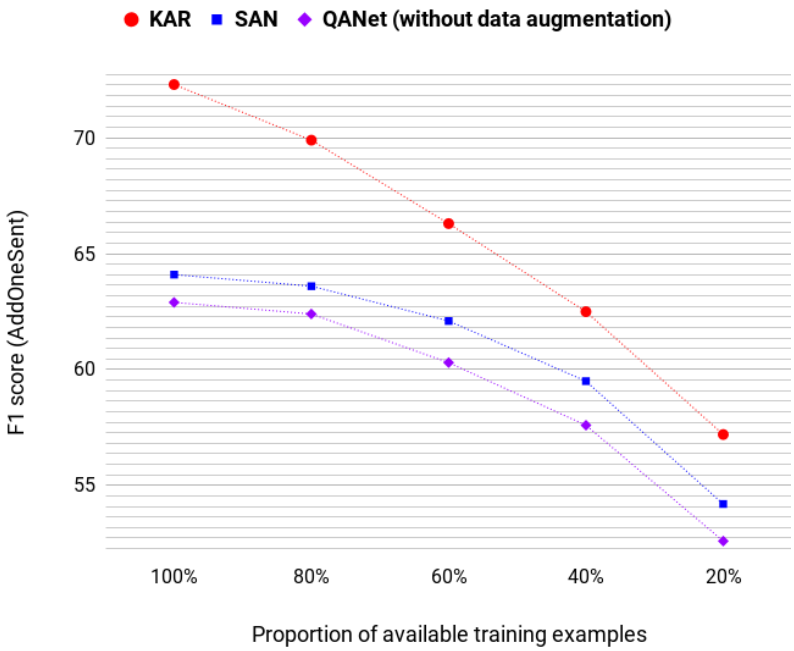

Figure 3.4: With KAR, SAN, and QANet (without data augmentation) trained on the training subsets, we evaluate their performance on AddOneSent.

3.5.3 Model Comparison in the Hunger for Data

We compare KAR with other MRC models in the hunger for data. Specifically, instead of using all the training examples, we produce several training subsets (i.e. subsets of the training examples) so as to study the relationship between the proportion of the available training examples and the performance. We produce each training subset by sampling a specific number of questions from all the questions relevant to each passage. By separately sampling 1, 2, 3, and 4 questions on each passage, we obtain four training subsets, which separately contain 20%, 40%, 60%, and 80% of the training examples. As shown in Figure 3.2, with KAR, SAN (re-implemented), and QANet (re-implemented without data augmentation) trained on these training subsets, we evaluate their performance on the development set, and find that KAR performs much better than SAN and QANet. As shown in Figure 3.3 and Figure 3.4, with the above KAR, SAN, and QANet trained on the same training subsets, we also evaluate their performance on the adversarial sets, and still find that KAR performs much better than SAN and QANet. That is to say, when only a subset of the training examples are available, KAR outperforms the state-of-the-art MRC models by a large margin, and is still reasonably robust to noise.

3.6 Analysis

According to the experimental results, KAR is not only comparable in performance with the state-of-the-art MRC models, but also superior to them in terms of both the hunger for data and the robustness to noise. The reasons for these achievements, we believe, are as follows:

- KAR is designed to utilize the pre-extracted inter-word semantic connections from the data enrichment method. Some inter-word semantic connections, especially those obtained through multi-hop semantic relation chains, are very helpful for the prediction of answer spans, but they will be too covert to capture if we simply leverage recurrent neural networks (e.g. BiLSTM) and pre-trained word vectors (e.g. GloVe).

- An inter-word semantic connection extracted from a passage-question pair usually also appears in many other passage-question pairs, therefore it is very likely that the inter-word semantic connections extracted from a small amount of training examples actually cover a much larger amount of training examples. That is to say, we are actually using much more training examples for model optimization than the available ones.

- Some inter-word semantic connections are distracting for the prediction of answer spans. For example, the inter-word semantic connection between “bank” and

"waterside" makes no sense given the context "the bank manager is walking along the waterside". It is the knowledge aided attention mechanisms that enable KAR to ignore such distracting inter-word semantic connections so that only the important ones are used.

3.7 Conclusion

In this chapter, we integrate the neural networks of MRC models with the general knowledge of human beings embodied in knowledge bases. Specifically, we use WordNet as our knowledge base, and propose a data enrichment method and an end-to-end MRC model KAR. The data enrichment method is used to extract inter-word semantic connections from each given passage-question pair. The extraction results are provided as general knowledge to KAR, which explicitly uses such general knowledge to assist its attention mechanisms. The experimental results show that KAR is not only comparable in performance with the state-of-the-art MRC models, but also superior to them in both the hunger for data and the robustness to noise. The limitation of this work is that WordNet is a small knowledge base, and it only embodies semantic knowledge. In the future, we will adopt more knowledge bases, such as ConceptNet and Freebase, to improve the amount and scope of the available general knowledge.

4 Probing the Strength of Graph Neural Networks in Multi-hop Machine Reading Comprehension

4.1 Introduction

As a sub-field of Question Answering (QA), Machine Reading Comprehension (MRC) requires a machine to read a context and answer a relevant question based on its comprehension to the context. In this chapter, we focus on multi-hop MRC tasks, where each context includes multiple passages, and the question needs to be answered by performing multi-hop reasoning on at least two of them. As shown in Table 4.1, the challenge of multi-hop MRC tasks is three-fold. First of all, for each given context-question pair, the Supporting Facts (SFs), which refer to the sentences necessary for answering the question, are scattered across two or more passages in the context, thus the question is unanswerable if any of these passages is ignored. In the next place, besides the passages containing SFs, each context also includes some distracting passages, which do not contain any SF. Last but not least, besides performing QA for each question, it is also required to extract the

Question: What was the former band of the member of Mother Love Bone who died just before the release of "Apple"?
Passage 1: Return to Olympus **Sentence 1:** Return to Olympus is the only album by . . . **Sentence 2:** It was released after the band had broken up and after lead singer Andrew Wood (later of Mother Love Bone) had died of a drug overdose in 1990. **Sentence 3:** Stone Gossard, of Pearl Jam, had compiled . . .
Passage 2: Mother Love Bone **Sentence 4:** Mother Love Bone was an American rock band that formed . . . **Sentence 5:** The band was active from 1987 to 1990. **Sentence 6:** Frontman Andrew Wood's personality and compositions helped to catapult the group to the top of the burgeoning . . . **Sentence 7:** Wood died only days before the scheduled release of the band's debut album, "Apple", thus ending the group's hopes of success. **Sentence 8:** The album was finally released a few months later.
Distracting Passages: Passage 3, . . . , Passage 10.
Supporting Facts: Sentence 1, Sentence 2, Sentence 4, Sentence 6, Sentence 7.
Answer: Malfunkshun

Table 4.1: A multi-hop MRC example provided by [2].

SFs so that the QA process is explainable.

To address multi-hop MRC tasks, many multi-hop MRC models have been proposed, where the state-of-the-art models have achieved great performance in both QA and SF extraction. According to the leaderboard of HotpotQA [2], there are two trends in developing multi-hop MRC models. The first trend is to apply pre-trained language models, such as BERT [18] and RoBERTa [63]. A pre-trained language model is usually pre-trained on a large-scale corpus for some self-supervised tasks, such as masked token prediction and next sentence prediction. By applying pre-trained language models, multi-hop MRC models can obtain rich contextual information. However, since the pre-training tasks of pre-trained language models are constrained to the token and sentence levels, the above obtained contextual information is mainly helpful for the within-passage reasoning rather than cross-passage reasoning of multi-hop MRC models. The second trend is to construct filter-reader pipelines, where the filter tries to exclude the distracting passages from each context and the reader operates on the remaining passages. The filter reduces the difficulty for the reader to perform multi-hop reasoning, thus multi-hop MRC models constructed as such pipelines usually outperform those constructed in an end-to-end manner. However, the training of these pipeline models is not only troublesome but also dependent on the ground-truth SFs, which are much more expensive to collect than the ground-truth answers and thus unavailable in the low-resource cases.

In many multi-hop MRC models, multi-hop reasoning is realized by applying Graph

Neural Networks (GNNs). This is a two-stage process, where the first stage is to construct a graph for each given context-question pair, and the second stage is to pass messages on the constructed graph [64, 65, 66, 67, 68, 69, 70]. The constructed graph is expected to embody the clues for multi-hop reasoning, thus applying GNNs in this way can be seen as utilizing the knowledge embodied in a dynamically created knowledge base. Most of the existing GNN-based multi-hop MRC models heavily rely on pre-trained language models and filter-reader pipelines. Although these models are indeed good in performance, it is still not clear how much GNNs contribute to their performance. To probe the strength of GNNs in multi-hop MRC, we propose a novel multi-hop MRC model named Graph Aided Reader (GAR), which uses GNN methods to perform multi-hop reasoning, but is free of any pre-trained language model and completely end-to-end. For graph construction, GAR utilizes the **topic-referencing relations between passages** and the **entity-sharing relations between sentences**, which is aimed at obtaining the most sensible reasoning clues. For message passing, GAR simulates a **top-down reasoning** and a **bottom-up reasoning**, which is aimed at making the best use of the above obtained reasoning clues. According to the experimental results, GAR even outperforms several competitors relying on pre-trained language models and filter-reader pipelines, which implies that GAR benefits a lot from its GNN methods. On this basis, GAR can further benefit from applying pre-trained language models, but pre-trained language models can mainly facilitate the within-passage reasoning rather than cross-passage reasoning of

GAR. Moreover, compared with the competitors constructed as filter-reader pipelines, GAR is not only easier to train, but also more applicable to the low-resource cases.

4.2 Related Works

4.2.1 Multi-passage Single-hop MRC Models

Although multi-passage single-hop MRC tasks do not require multi-hop reasoning, the approaches to developing multi-passage single-hop MRC models are useful for developing multi-hop MRC models. Most passages in each multi-passage context are distracting passages, which are irrelevant to the corresponding question, therefore it is necessary to screen them out and only reason on the relevant passages. To this end, a confidence approach [45] first uses a passage ranking model to select the most relevant passages from each context, then samples a subset from the selected passages for training (the full set is used for inference), next generates an unnormalized answer span distribution over each sampled passage, and finally uses a shared normalization to generate a normalized answer span distribution over all the sampled passages. A deep cascade approach [71] first uses a document ranking model and a paragraph ranking model to select the most relevant passages (i.e. the most relevant paragraphs in the most relevant documents), then separately reasons on the selected passages based on the question, and finally performs document extraction and paragraph extraction as two auxiliary tasks of answer extraction.

The above approaches are aimed at screening out distracting passages in advance through a ranking process, which is necessary when each context consists of many passages. But if each context only consists of a few passages, the ranking process can be skipped and distracting passages can be screened out by simply performing passage extraction together with answer extraction in a multi-task manner [72].

Another difficulty in multi-passage single-hop MRC tasks is that the answer to each question may appear in multiple passages of the corresponding context, since the passages of a context are independent from each other. Besides, the appearance of the answer in a passage does not imply that the passage alone is sufficient for answering the question, since answer extraction may require evidences from other passages. Therefore, reasoning across passages to collect evidences usually improves the performance of multi-passage single-hop MRC models. An answer re-ranking approach [73] first collects the most probable answer candidates from a baseline MRC model, and then uses two proposed re-rankers, namely strength-based re-ranker and coverage-based re-ranker, to re-score the answer candidates by aggregating evidences from the corresponding passages for each answer candidate. A cross-passage answer verification approach [53] performs answer content modeling and answer verification as two auxiliary tasks of answer extraction, where answer content modeling is estimating whether each word should be included in the answer, and answer verification is verifying the answer from each passage with those from the other passages to estimate its correctness.

4.2.2 Multi-hop MRC Models

The key to address multi-hop MRC tasks is to perform multi-hop reasoning on each multi-passage context with the corresponding question. Many multi-hop MRC models perform multi-hop reasoning by applying GNNs, and the difference between them mainly lies in how to construct a graph and how to pass messages on the constructed graph. KGNN [68] passes messages on a knowledge-enhanced entity graph, which is constructed according to the co-reference relations between entities and the relational facts in a knowledge base. DFGN [67] passes messages on a mention-based entity graph through a dynamic graph attention mechanism, where in each reasoning hop, irrelevant entities are softly masked out of the graph. C2F Reader [64] is a re-implementation of DFGN, but it turns out that when the pre-trained language model in DFGN is applied in a fine-tuning manner, the dynamic graph attention mechanism contributes little to the performance. CogQA [70] coordinates an extraction module and a reasoning module to maintain a cognitive graph as its working memory, where in each reasoning hop, the extraction module adds next-hop entities and answer candidates to the graph, and the reasoning module passes messages on the graph to prepare guiding clues for the extraction module. SAE [65] passes messages on a sentence graph, which is constructed according to three types of relations between sentences. Transformer-XH [69] passes messages on a passage graph through an eXtra Hop attention mechanism, where contextual information is propagated between the

interconnected passages in the graph to generate globally contextualized representations. HGN [66] passes messages on a hybrid graph with a passage-sentence-entity hierarchy, which allows it to utilize the relations at different levels.

Besides applying GNNs, there are also several other ways for multi-hop MRC models to perform multi-hop reasoning. QFE [74] applies an iterative evidence collection process, where each iteration is aimed at extracting one evidence sentence from the remaining sentences in the context, and thereby performs evidence collection as an auxiliary task of answer extraction. TAP [75] extends the pre-training of BERT [18] with a span selection auxiliary task, and thereby applies the pre-trained BERT first in a filter to collect evidences, and then in a reader to extract the answer from the collected evidences. LQR-net [76] applies an iterative reasoning process based on a reading module and a reformulation module, where in each iteration the reading module produces a question-aware context representation, and the reformulation module uses this context representation to update the question representation. DrKIT [77] uses a large corpus as a virtual knowledge base, and iteratively queries it by applying sparse-matrix TF-IDF indices and Maximum Inner Product Search (MIPS), which is aimed at extracting a reasoning chain.

4.3 Graph Aided Reader

An ambitious goal for developing multi-hop MRC models is to make them perform multi-hop reasoning like human beings. In cognitive science, the reasoning of human beings is

usually interpreted as searching on an abstract graph composed of real-world concepts [78]. Inspired by this notion, in many multi-hop MRC models, multi-hop reasoning is realized by applying GNNs, which refer to a family of methods for performing relational reasoning on graph-structured representations [79]. Since the underlying data of multi-hop MRC tasks is of a sequential structure, the first stage of applying GNNs is to construct a graph for each given context-question pair. The constructed graph could be an entity graph [64, 67, 68, 70], a sentence graph [65], a passage graph [69], or a hybrid graph with certain hierarchy [66], and it is expected to embody the clues for multi-hop reasoning. To integrate the neural networks of multi-hop MRC models with these reasoning clues, the second stage of applying GNNs is to pass messages on the constructed graph. This is usually performed in an iterative manner, where in each iteration the representation of each graph node (i.e. entity, sentence, passage, etc.) is updated with the representations of its connected nodes [64, 65, 66, 67, 68, 69, 70]. The resulting final node representations can be used to facilitate the predictions for QA and SF extraction.

While following the above two-stage process, most of the existing GNN-based multi-hop MRC models also heavily rely on pre-trained language models and filter-reader pipelines. In these models, pre-trained language models are applied so that rich contextual information is injected into the representations of the graph nodes, and filter-reader pipelines are constructed so that most of the distracting passages are excluded from multi-hop reasoning. As a result, these models are indeed good in performance. However,

it is still not clear how much GNNs contribute to their performance. To prob the strength of GNNs in multi-hop MRC, we propose a novel multi-hop MRC model named Graph Aided Reader (GAR), which uses GNN methods to perform multi-hop reasoning, but is free of any pre-trained language model and completely end-to-end.

4.3.1 Task Definition

A multi-hop MRC task is a combination of QA and SF extraction. Given a context C and a question Q, each of which is a sequence of words, suppose that C is divided into m passages $\{P_1, \ldots, P_m\}$ and further divided into n sentences $\{S_1, \ldots, S_n\}$, then QA is to predict an answer to Q, which is either "yes" or "no" or a span $[a_s, a_e]$ referring to the sub-sequence $\{c_{a_s}, \ldots, c_{a_e}\}$ in C, and SF extraction is to predict whether each passage P_i contains any SF and whether each sentence S_j is an SF. According to the example in Table 4.1, entities play a very important role in multi-hop reasoning. Since there have been many well-established techniques for entity recognition and entity linking, we assume that the entities in each given context-question pair have been pre-extracted. Besides, we also assume that a topic entity has been pre-specified for each passage in the context.

4.3.2 Graph Construction

The existing graph construction methods are based on various heuristic rules, such as interconnecting the entities in the same sentence, interconnecting the sentences in the same

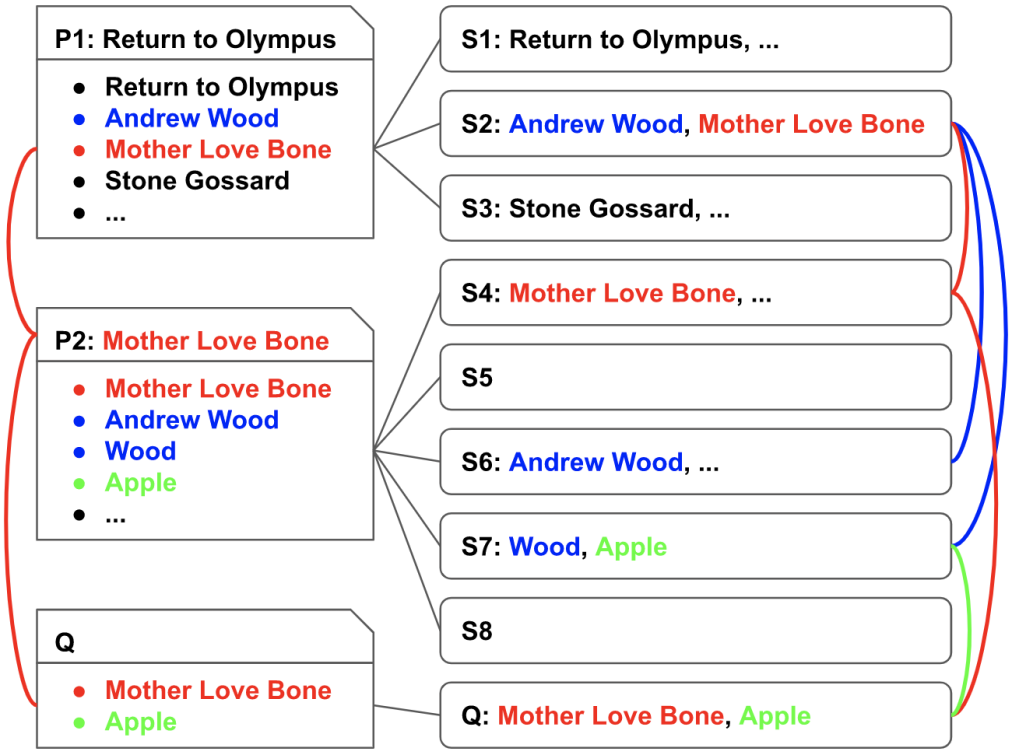

Figure 4.1: A part of the graph constructed for the example in Table 4.1. The distracting passages are not shown, but they are involved in the constructed graph.

passage, and interconnecting all the passages in the context. However, most of these rules are too rigid to match the reasoning of human beings, thus the reasoning clues embodied in a constructed graph do not make much sense. To obtain the most sensible reasoning clues, we adopt some humanoid elements when designing our graph construction method. Specifically, according to our observation and experience, human beings tend to perform multi-hop reasoning at the passage and sentence levels. On the one hand, human beings tend to reason between two passages if one refers to the other by the topic (e.g. passage 1 and passage 2 in Table 4.1). On the other hand, when reasoning between such two passages, human beings especially tend to reason between two sentences if they are in different passages but share at least one entity (e.g. sentence 2 and sentence 6 in Table 4.1). This implies that the **topic-referencing relations between passages** and the **entity-sharing relations between sentences** are essential for human beings to perform multi-hop reasoning. To utilize these two types of relations, we formulate two heuristic rules:

- For any two passages P_i and P_j, suppose that they separately hold two entity sets Z_{P_i} and Z_{P_j}, and separately hold two topic entities $\tilde{z}_{P_i}$ and $\tilde{z}_{P_j}$, then P_i and P_j are interconnected if $i \neq j$ and $Z_{P_i} \cap Z_{P_j} \cap \{\tilde{z}_{P_i}, \tilde{z}_{P_j}\} \neq \varnothing$.

- For any two sentences S_i and S_j, suppose that they are separately in two passages $P_{i'}$ and $P_{j'}$, and separately hold two entity sets Z_{S_i} and Z_{S_j}, then S_i and S_j are

interconnected if $P_{i'}$ and $P_{j'}$ are interconnected according to the above rule and $Z_{S_i} \cap Z_{S_j} \neq \varnothing$.

For each given context-question pair, to enforce the above two heuristic rules, we treat the question as an additional and special passage in the context, where there is no topic entity and only a single sentence. Besides, considering the inherent affiliations between passages and sentences, we also interconnect any passage-sentence pair if the sentence is in the passage. In this way, we finally construct a hybrid graph with a passage-sentence hierarchy. As shown in Figure 4.1, the reasoning clues embodied in such a constructed graph make a lot of sense.

4.3.3 Message Passing

A graph constructed using the above method embodies not only the reasoning clues at the passage and sentence levels, but also those across the two levels. To make the best use of these reasoning clues, we design our message passing method as an iterative process, where each iteration simulates a **top-down reasoning** and a **bottom-up reasoning**. Specifically, for each given context-question pair, we assume that the final word representations have been obtained. Meanwhile, to be compatible with the constructed graph, we still treat the question as a single-sentence passage appended to the context. On top of that, to initialize the passage representations, we perform an attention pooling on the final word representations corresponding to each passage, and to initialize the sentence

representations, we perform another attention pooling on the final word representations corresponding to each sentence. Suppose that the context is divided into m passages and further divided into n sentences, then we obtain $m + 1$ initial passage representations $U_P^0 \in \mathbb{R}^{d\times(m+1)}$ and $n + 1$ initial sentence representations $U_S^0 \in \mathbb{R}^{d\times(n+1)}$, where d is a hyper-parameter representing the dimensionality, and the last column in each matrix corresponds to the question. On this basis, we start the iterative message passing process, where in each iteration t we take four steps to update the current passage and sentence representations U_P^{t-1} and U_S^{t-1}:

Step 1: passage-passage updating. In this step, we perform a passage-passage graph attention to update the passage representations:

$$\tilde{U}_P^{t-1} = \mathrm{fuse}\big(U_P^{t-1}, U_P^{t-1}\mathrm{softmax_c}(A_{PP}^{t-1})\big)$$

where A_{PP}^{t-1} is a similarity matrix of size $(m + 1) \times (m + 1)$, $\mathrm{softmax_c}(\cdot)$ represents a column-wise softmax normalization, and $\mathrm{fuse}(X, Y)$ represents fusing Y into X. In A_{PP}^{t-1}, each element $A_{PP}^{t-1}[i, j]$ is a real number indicating the similarity between the i-th passage P_i and the j-th passage P_j. If P_i and P_j are interconnected in the constructed graph, we calculate $A_{PP}^{t-1}[i, j]$ by applying the similarity function proposed by [41]:

$$A_{PP}^{t-1}[i, j] = v_A^\top[u_{P_i}^{t-1}; u_{P_j}^{t-1}; u_{P_i}^{t-1} \odot u_{P_j}^{t-1}]$$

where v_A is a trainable vector, $u_{P_i}^{t-1}$ and $u_{P_j}^{t-1}$ are separately the representations of P_i and P_j (i.e. the i-th and j-th columns in U_P^{t-1}), $\odot$ represents an element-wise multiplication,

and $[;]$ represents a column-wise concatenation. Otherwise, we mask $A_{PP}^{t-1}[i,j]$ by setting it to $-\infty$ (if an entire column in A_{PP}^{t-1} is masked, we set the corresponding column in the result of $\mathrm{softmax_c}(A_{PP}^{t-1})$ to all zeros). As for $\mathrm{fuse}(X,Y)$, we implement it as the fusion function proposed by [53]:

$$\mathrm{fuse}(X,Y) = M_{XY} \odot N_{XY} + (1 - M_{XY}) \odot X$$

$$M_{XY} = \mathrm{sigmoid}(W_M^\top [X; Y; X \odot Y; X - Y] + b_M)$$

$$N_{XY} = \tanh(W_N^\top [X; Y; X \odot Y; X - Y] + b_N)$$

where W_M and W_N are trainable weights, and b_M and b_N are trainable biases.

Step 2: passage-sentence updating. In this step, we perform a passage-sentence graph broadcasting to update the sentence representations:

$$\tilde{U}_S^{t-1} = \mathrm{fuse}(U_S^{t-1}, \tilde{U}_P^{t-1} B_{PS})$$

where B_{PS} is an adjacency matrix of size $(m+1) \times (n+1)$. In B_{PS}, each element $B_{PS}[i,j]$ is a binary number indicating whether the i-th passage P_i and the j-th sentence S_j are interconnected in the constructed graph. It is obvious that for each sentence, the corresponding column in B_{PS} actually points out which passage the sentence is in, thus the corresponding column in the result of $\tilde{U}_P^{t-1} B_{PS}$ is the representation of this passage.

Step 3: sentence-sentence updating. In this step, we perform a sentence-sentence graph attention to update the sentence representations:

$$U_S^t = \mathrm{fuse}\big(\tilde{U}_S^{t-1}, \tilde{U}_S^{t-1} \mathrm{softmax_c}(A_{SS}^{t-1})\big)$$

where A_{SS}^{t-1} is a similarity matrix of size $(n+1) \times (n+1)$. The formation of A_{SS}^{t-1} is the same as that of A_{PP}^{t-1} in step 1, except that everything occurs at the sentence level and the calculation is based on the sentence representations $\tilde{U}_S^{t-1}$.

Step 4: sentence-passage updating. In this step, we perform a sentence-passage graph broadcasting to update the passage representations:

$$U_P^t = \text{fuse}(\tilde{U}_P^{t-1}, U_S^t B_{PS}^\top D_{PS}^{-1})$$

where D_{PS} is a degree matrix of size $(m+1) \times (m+1)$. In D_{PS}, each diagonal element $D_{PS}[i,i]$ is a natural number indicating how many sentences are interconnected with the i-th passage P_i in the constructed graph, and the off-diagonal elements are all zeros. It is obvious that for each passage, the corresponding column in $B_{PS}^\top$ actually points out which sentences are in the passage, and can be normalized through $B_{PS}^\top D_{PS}^{-1}$, thus the corresponding column in the result of $U_S^t B_{PS}^\top D_{PS}^{-1}$ is the average representation of these sentences.

During the above four steps, each passage and sentence receives one and only one message from each passage and sentence interconnected with it in the constructed graph. Therefore, each message passing iteration is actually a simulation to a reasoning hop. According to the hierarchical structure of the constructed graph, this reasoning hop is a combination of two reasoning operations, namely a **top-down reasoning**, which corresponds to step 1 and step 2, and a **bottom-up reasoning**, which corresponds to step 3 and step 4. Since the number of required reasoning hops varies between multi-hop MRC tasks, we use a

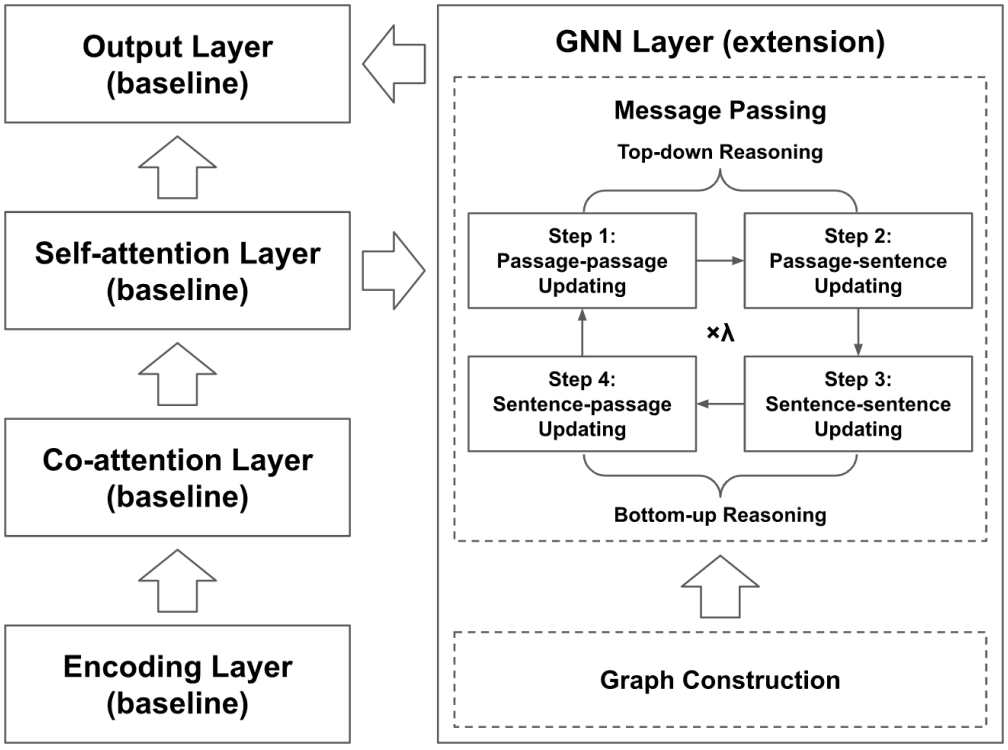

Figure 4.2: Our multi-hop MRC model: Graph Aided Reader (GAR).

hyper-parameter $\lambda \in \mathbb{N}$ to represent it. That is to say, for each given context-question pair, we perform message passing for λ iterations to obtain the final passage and sentence representations U_P^λ and U_S^λ.

4.3.4 Overall Architecture

Instead of constructing our multi-hop MRC model from scratch, we do that by extending an existing baseline model with our graph construction method and message passing method. To highlight the performance contribution from these two GNN methods, we

expect the baseline model to be a multi-passage single-hop MRC model that is free of any pre-trained language model and completely end-to-end. The multi-task deep attention model proposed by [71] is just such a model, thus we re-implement it as our baseline model. In this re-implementation, we replace each bi-directional LSTM [8] in the original model with a transformer encoder [12] to improve the training efficiency. On this basis, by inserting a GNN layer before the output layer of the baseline model, we extend the baseline model into a multi-hop MRC model, which we name as Graph Aided Reader (GAR). As shown in Figure 4.2, GAR consists of five layers:

Encoding layer. In this layer, we map the words in each given context-question pair to a vector space. First, for each of the words, we pass the concatenation of its GloVe embedding [56] and character-level CNN embedding [57] through a linear layer of output dimensionality d to obtain its merged embedding. Then, for each passage in the context C, we pass its corresponding merged word embeddings through a transformer encoder configured with a position embedding layer, and for the question Q, we also pass its corresponding merged word embeddings through this transformer encoder. As a result, we obtain the contextualized context word representations $G_C \in \mathbb{R}^{d \times |C|}$ and the contextualized question word representations $G_Q \in \mathbb{R}^{d \times |Q|}$.

Co-attention layer. In this layer, we align the context words with the question words. First, we make G_C attend to G_Q and vice versa:

$$\tilde{G}_C = \mathrm{fuse}\big(G_C, G_Q \mathrm{softmax}_{\mathrm{c}}(E_{CQ}^{\top})\big)$$

$$\tilde{G}_Q = \mathrm{fuse}\big(G_Q, G_C \mathrm{softmax_c}(E_{CQ})\big)$$

$$E_{CQ} = \mathrm{ReLU}(W_E^\top G_C)^\top \mathrm{ReLU}(W_E^\top G_Q)$$

where $\mathrm{fuse}(X, Y)$ is the fusion function introduced earlier, E_{CQ} is a word-level similarity matrix, and W_E is a trainable weight. Then, for each passage in C, we pass its corresponding columns in the fusion result $\tilde{G}_C$ through a transformer encoder, and for Q, we pass the fusion result $\tilde{G}_Q$ through another transformer encoder. As a result, we obtain the question-aware context word representations $H_C \in \mathbb{R}^{d \times |C|}$ and the context-aware question word representations $H_Q \in \mathbb{R}^{d \times |Q|}$.

Self-attention layer. In this layer, we further align the context words with themselves. First, we make H_C attend to itself:

$$\tilde{H}_C = \mathrm{fuse}\big(H_C, H_C \mathrm{softmax_c}(F_{CC})\big)$$

$$F_{CC} = H_C^\top W_F^\top H_C$$

where F_{CC} is a word-level similarity matrix, and W_F is a trainable weight. Then, for each passage in C, we pass its corresponding columns in the fusion result $\tilde{H}_C$ through a transformer encoder. As a result, we obtain the self-aware context word representations $L_C \in \mathbb{R}^{d \times |C|}$.

GNN layer. In this layer, we perform multi-hop reasoning at the passage and sentence levels. By applying our graph construction method in advance, we have already constructed a hybrid graph with a passage-sentence hierarchy. Given this graph, we apply our

message passing method, where we use L_C and H_Q as the final word representations. As a result, we obtain the final passage and sentence representations U_P^λ and U_S^λ.

Output layer. In this layer, we perform the predictions for QA and SF extraction. To predict whether the answer is "yes" or "no" or a span, we calculate the answer type distribution:

$$o_{type} = \text{softmax}(W_{type}^\top [r_C; r_Q])$$

where W_{type} is a trainable weight, r_C is the context representation obtained through an attention pooling on L_C, and r_Q is the question representation obtained through another attention pooling on H_Q. To deal with different answer types in a uniform manner, we use a trainable vector to represent the non-span types, and append it to the end of L_C to obtain L_C^+. In this way, all the answer types can be treated as the span type. To predict the answer span, we calculate the answer start distribution and the answer end distribution:

$$o_{start} = \text{softmax}(L_C^{+\top} W_{start}^\top r_Q)$$

$$o_{end} = \text{softmax}(L_C^{+\top} W_{end}^\top r_Q)$$

where W_{start} and W_{end} are trainable weights. To predict whether each passage in C contains any SF and whether each sentence in C is an SF, we calculate the passage extraction probabilities and the sentence extraction probabilities:

$$o_{pasg} = \text{sigmoid}(U_P^\lambda[:-1]^\top W_{pasg}^\top U_P^\lambda[-1])$$

$$o_{senc} = \text{sigmoid}(U_S^\lambda[:-1]^\top W_{senc}^\top U_S^\lambda[-1])$$

where $W_{pasg}^{\top}$ and $W_{senc}^{\top}$ are trainable weights. On this basis, for the training, we minimize a joint loss through stochastic gradient descent:

$$\mathcal{L} = \mathrm{CE}(o_{start}) + \mathrm{CE}(o_{end}) + \alpha \mathrm{CE}(o_{type}) + \beta \mathrm{CE}(o_{pasg}) + \gamma \mathrm{CE}(o_{senc})$$

where α, β, and γ are hyper-parameters, and $\mathrm{CE}(\cdot)$ represents a cross-entropy loss. For the inference, we first derive the answer type from o_{type}. Then, if the answer is a span, we go through all the within-sentence spans in C, and take the span $[i, j]$ that maximizes $o_{start}[i] \cdot o_{end}[j] \cdot o_{senc}[k_{i,j}] \cdot o_{pasg}[k'_{i,j}]$ as the answer span $[a_s, a_e]$, where $k_{i,j}$ and $k'_{i,j}$ are separately the sentence index and passage index corresponding to $[i, j]$. Finally, for each sentence S_i in C, we extract it as an SF if $o_{senc}[i] \cdot o_{pasg}[i'] > \delta$, where i' is the passage index corresponding to S_i, and δ is a hyper-parameter.

4.4 Experiments

4.4.1 Experimental Settings

Multi-hop MRC task. We adopt HotpotQA as our multi-hop MRC task. The dataset of HotpotQA contains 112,779 examples, each of which consists of a context-question pair and its ground-truth answer and ground-truth SFs. The passages in each context are sampled from the Wikipedia lead paragraphs, each of which comes with a topic entity. HotpotQA provides two settings, namely the distractor setting, where each context includes two passages containing SFs and up to eight distracting passages, and the fullwiki

setting, where the Wikipedia lead paragraphs are merged into a global context. For model evaluation, HotpotQA not only adopts EM and F1 score as the metrics for QA and SF extraction, but also defines a joint EM and a joint F1 score as the overall metrics.

Implementation details. We use TAGME [80] for entity recognition and entity linking, use spaCy [58] for word tokenization, and use TensorFlow [59] for model implementation. For the transformer encoders, we implement each of them as a stack of three encoder blocks according to the block-level configuration used in [12]. For the hyper-parameters, we set d to 256, λ to 2, α to 1, β to 1, γ to 5, and δ to 0.4. For model optimization, we apply an AdamW optimizer [81] with an initial learning rate of 0.0001, a weight decay factor of 0.01, and a batch size of 32. Besides, to boost the performance, we also apply exponential moving average with a decay factor of 0.999.

4.4.2 Model Comparison

We train GAR on HotpotQA for both the distractor setting and the fullwiki setting. In the fullwiki setting, for each question, we use an information retrieval system to retrieve a ten-passage relevant context from the global context, and feed the resulting context-question pair into GAR. As shown in Table 4.2 and Table 4.3, although GAR is free of any pre-trained language model and completely end-to-end, it still achieves strong performance in both QA and SF extraction. In particular, GAR even outperforms several competitors relying on pre-trained language models and filter-reader pipelines. Therefore,

Model	**PLM**	**FRP**	**GSFs**	**QA (EM / F1)**	**SF Extr (EM / F1)**	**Joint (EM / F1)**
Competitors						
KGNN [68]	No	No	Yes	50.81 / 65.75	38.74 / 76.79	22.40 / 52.82
QFE [74]	No	No	Yes	53.86 / 68.06	57.75 / 84.49	34.63 / 59.61
DFGN [67]	Yes	Yes	Yes	56.31 / 69.69	51.50 / 81.62	33.62 / 59.82
TAP [75]	Yes	Yes	Yes	58.63 / 71.48	46.84 / 82.98	32.03 / 61.90
LQR [76]	Yes	No	Yes	60.20 / 73.78	56.21 / 84.09	36.56 / 63.68
SAE [65]	Yes	Yes	Yes	60.36 / 73.58	56.93 / 84.63	38.81 / 64.96
HGN [66]	Yes	Yes	Yes	66.07 / 79.36	60.33 / 87.33	43.57 / 71.03
C2F [64]	Yes	Yes	Yes	67.98 / 81.24	60.81 / 87.63	44.67 / 72.73
Our Models						
GAR	No	No	Yes	56.61 / 71.40	58.36 / 87.27	36.79 / 64.01
GAR-BERT	Yes	No	Yes	62.67 / 76.35	59.50 / 87.98	40.64 / 68.74
GAR-NOSF	No	No	No	56.20 / 71.17	N/A	N/A

Table 4.2: Model comparison on the test set of HotpotQA for the distractor setting. PLM means relying on a pre-trained language model. FRP means relying on a filter-reader pipeline. GSFs means being trained with ground-truth SFs.

Model	**PLM**	**FRP**	**GSFs**	**QA (EM / F1)**	**SF Extr (EM / F1)**	**Joint (EM / F1)**
Competitors						
QFE [74]	No	No	Yes	28.66 / 38.06	14.20 / 44.35	8.69 / 23.10
KGNN [68]	No	No	Yes	27.65 / 37.19	12.65 / 47.19	7.03 / 24.66
CogQA [70]	Yes	No	Yes	37.12 / 48.87	22.82 / 57.69	12.42 / 34.92
DrKIT [77]	Yes	No	Yes	42.13 / 51.72	37.05 / 59.84	24.69 / 42.88
Transformer-XH [69]	Yes	No	Yes	51.60 / 64.07	40.91 / 71.42	26.14 / 51.29
HGN [66]	Yes	Yes	Yes	56.71 / 69.16	49.97 / 76.39	35.63 / 59.86
Our Models						
GAR	No	No	Yes	48.22 / 61.33	48.34 / 73.89	30.61 / 52.95
GAR-BERT	Yes	No	Yes	52.28 / 64.84	49.00 / 74.73	33.00 / 56.10
GAR-NOSF	No	No	No	47.50 / 60.62	N/A	N/A

Table 4.3: Model comparison on the test set of HotpotQA for the fullwiki setting. The competitors are the published fully-functional single basic models that focus on multi-hop MRC.

GAR benefits a lot from its GNN methods.

Besides GAR, we also train two variants of it on HotpotQA for both of the two settings. The first variant of GAR is GAR-BERT, where we apply a pre-trained language model BERT to GAR in a feature-based manner. Specifically, we replace the GloVe word embeddings with the word embeddings generated by BERT. As shown in Table 4.2 and Table 4.3, GAR-BERT further outperforms GAR, but the performance gain is mainly reflected in QA rather than SF extraction. Therefore, GAR can further benefit from applying pre-trained language models, but pre-trained language models can mainly facilitate the within-passage reasoning rather than cross-passage reasoning of GAR.

The second variant of GAR is GAR-NOSF, where we remove the ground-truth SFs from the training data of GAR. By doing so, we are deliberately creating a low-resource case that is beyond the ability of filter-reader pipelines. In this case, we use the answer-containing sentences as a substitute for the ground-truth SFs. As shown in Table 4.2 and Table 4.3, the QA performance of GAR-NOSF is just slightly worse than that of GAR and still comparable to that of several strong competitors. Therefore, compared with the competitors constructed as filter-reader pipelines, GAR is not only easier to train, but also more applicable to the low-resource cases.

Model	**QA (F1)**	**SF Extr (F1)**	**Joint (F1)**
GAR	71.80	87.52	64.77
w/o top-down reasoning	69.23	84.97	61.95
w/o bottom-up reasoning	68.62	82.79	60.98
w/o both (baseline)	66.17	79.66	58.20

Table 4.4: Ablation results on the development set of HotpotQA for the distractor setting.

4.4.3 Ablation Study

To investigate the performance contribution from the GNN methods of GAR, we perform ablations on the development set of HotpotQA for the distractor setting. Specifically, we separately ablate the two reasoning operations constituting each reasoning hop, namely the **top-down reasoning** and the **bottom-up reasoning**. To ablate each of these operations, we skip its corresponding steps in each message passing iteration. Besides, we also ablate both of these operations by skipping the entire message passing process, which actually leads to the baseline model. As shown in Table 4.4, each of these operations contributes a substantial portion of the performance, and the performance contribution from the **bottom-up reasoning** is slightly larger than that from the **top-down reasoning**.

4.5 Conclusion

In this chapter, we try to probe the strength of GNNs in multi-hop MRC, and thereby propose a novel multi-hop MRC model GAR. GAR is constructed by extending an existing multi-passage single-hop MRC model with two GNN methods, namely a graph construction method and a message passing method. Unlike most of the existing GNN-based multi-hop MRC models, which heavily rely on pre-trained language models and filter-reader pipelines, GAR is free of any pre-trained language model and completely end-to-end. However, by comparing GAR with the competitors relying on pre-trained language models and filter-reader pipelines, we verify that the GNN methods of GAR is an effective way to develop GNN-based multi-hop MRC models. The limitation of this work is that when each context includes too many passages (e.g. in the fullwiki setting of HotpotQA), GAR cannot work unless there is an information retrieval system. In the future, we will perform information retrieval by applying GNNs.

5 Exploring Cross-Lingual Transfer Learning with Unsupervised Machine Translation

5.1 Introduction

Virtual assistants, such as Amazon Alexa, Apple Siri, and Google Assistant, are increasingly popular due to the convenience they bring to customers. A core function of virtual assistants is Natural Language Understanding (NLU), which typically includes slot filling and intent classification. NLU models behind virtual assistants are generally trained in a supervised manner, which requires a large amount of annotated data. Collecting annotated data is not a big deal for high-resource languages, but difficult or even impossible for low-resource languages. As a result, when ported to a low-resource language, an NLU model may suffer from the so-called "data hungriness" [82]. This problem can be alleviated by conducting Cross-Lingual Transfer Learning (CLTL) [83], where annotated data in a high-resource source language is used to bootstrap an NLU model aimed at a low-resource target language.

CLTL can be seen as the neural-based knowledge transfer between languages. Given a source-target language pair, the key to CLTL is to learn a shared representation space between them. A traditional way to achieve this goal is to leverage cross-lingual word embeddings, which are obtained by mapping the words in both languages to a shared word embedding space [84, 85, 86, 87, 88, 89]. However, most studies on this topic only consider similar languages (e.g. English-German) but ignore distant languages (e.g. English-Japanese), since it is more challenging to conduct CLTL between distant languages than between similar languages. Recently, contextualized word embeddings generated by pre-trained language models have shown significant advantages over ordinary word embeddings [17, 18, 63]. For the purpose of CLTL, many efforts have been made to develop multilingual variants of pre-trained language models. These efforts have in turn brought about pre-trained multilingual language models, each of which is pre-trained on a multilingual corpus so that the learned representation space is not only rich in contextual clues but also shared by all the involved languages [90, 91, 92]. However, in this pre-training, the collection of the multilingual corpus is not obviously biased to any language, thus in the learned representation space, similar languages are still similar to each other, and distant languages are still distant from each other. As a result, although pre-trained multilingual language models have greatly promoted CLTL, it is still more challenging to conduct CLTL between distant languages than between similar languages. This opinion has been verified by several empirical studies on a popular pre-trained

multilingual language model named Multilingual BERT (M-BERT) [18], where the CLTL performance of M-BERT between similar languages is decent, but that between distant languages is still far from satisfactory [93, 94, 95].

From our point of view, CLTL can be analogized to the process of a human being learning a foreign language, where the prior knowledge on the native language plays an important role. Language educators believe that a foreign language learner can benefit a lot from translation, since translation not only involves all aspects of foreign language learning but also helps to enhance the correlation between the native language and the foreign language [96]. According to our observation and experience, this is especially the case when the native language and the foreign language are distant from each other. Inspired by these thoughts, to facilitate the CLTL of NLU models, especially the CLTL between distant languages, we propose a novel CLTL model named Translation Aided Language Learner (TALL), where CLTL is integrated with MT. Specifically, we adopt a pre-trained multilingual language model, which is the state of the art in CLTL, as our baseline model, and construct TALL by appending a decoder to it. On this basis, we directly fine-tune the baseline model as an NLU model to conduct CLTL, but put TALL through an MT-oriented pre-training before its NLU-oriented fine-tuning. We believe that the MT-oriented pre-training can help TALL to enhance the correlation between the given source-target language pair in its representation space, and thus can make CLTL easier to conduct in its NLU-oriented fine-tuning, especially when the source language and the

target language are distant from each other. To make use of unannotated data, which is not only large in amount but also available for every language, we implement the recently proposed Unsupervised Machine Translation (UMT) [97, 98, 99, 100, 101] technique in the MT-oriented pre-training of TALL.

To verify the effectiveness of TALL, we carry out a series of experiments to compare the CLTL performance of TALL with that of the baseline model. In these experiments, we address not only CLTL tasks between similar languages but also those between distant languages. For each given CLTL task, we separately use two popular pre-trained multilingual language models for model construction. To implement UMT, we collect unannotated sentences from Wikipedia dumps. To conduct CLTL, we separately collect annotated sentences from two multilingual NLU datasets. According to the experimental results, the application of UMT enables TALL to consistently achieve better CLTL performance than the baseline model without using more annotated data, and the performance gain is relatively prominent in the case of distant languages.

5.2 Related Works

Cross-lingual word embeddings. A traditional way to conduct CLTL is to leverage cross-lingual word embeddings, which are usually learned in an unsupervised manner. For example, [84] formulate the learning of cross-lingual word embeddings as an adversarial game, and explore several adversarial training methods to implement it. [85] first use

adversarial training to learn a linear mapping from the word embeddings of a source language to those of a target language, and then use a Procrustes solution to refine it. [86] first use an unsupervised initialization scheme to create an initial mapping, and then use a self-learning procedure to iteratively improve it. [87] propose a language-adversarial training method, and use it to address cross-lingual sentiment classification. Besides, there are also several studies on multilingual word embeddings. For example, [88] propose an unsupervised approach to learning multilingual word embeddings, which directly exploits the relations between all the involved languages. On this basis, [89] propose a multi-source CLTL model, which not only uses adversarial training to learn language-invariant features, but also uses a mixture-of-experts method to dynamically exploit the similarity between a target language and multiple source languages.

Pre-trained multilingual language models. The currently dominant way to conduct CLTL is to fine-tune pre-trained multilingual language models, which are multilingual variants of pre-trained language models, and are each pre-trained on a multilingual corpus. For example, [90] propose Rosita as a multilingual variant of ELMo, and pre-train it on a multilingual corpus covering 3 languages. [18] propose M-BERT as a multilingual variant of BERT, and pre-train it on a multilingual corpus covering 104 languages. [91] propose XLM as a multilingual variant of BERT, and pre-train it on a multilingual corpus covering 15 languages. [92] propose XLM-R as a multilingual variant of RoBERTa, and pre-train it on a multilingual corpus covering 100 languages. Besides, there are also several empirical

studies on M-BERT. For example, [93] carry out a large number of probing experiments to verify and interpret the zero-shot CLTL performance of M-BERT. [94] explore the zero-shot CLTL potential of M-BERT on 5 downstream tasks covering 39 languages. [95] provide a comprehensive study on the contribution of each component of M-BERT to its CLTL ability, which focuses on the impact of linguistic properties of the languages, the architecture of the model, and the learning objectives.

UMT technique. The UMT technique is aimed at reducing the reliance of MT models on parallel corpora. For example, [97] construct an MT model consisting of a language-invariant encoder and two language-specific decoders, and train it on a non-parallel corpus through denoising auto-encoding and back-translation. [98] construct an MT model consisting of a language-invariant pair of encoder and decoder, and train it on a non-parallel corpus not only through denoising auto-encoding and back-translation but also through adversarial training. [99] construct an MT model consisting of two pairs of encoder and decoder, which partially share their parameters, and train it on a non-parallel corpus not only through denoising auto-encoding and back-translation but also through adversarial training. [100] propose a simple but effective approach based on the above works, where the constructed MT model only consists of a language-invariant pair of encoder and decoder, and its training on a non-parallel corpus only requires denoising auto-encoding and back-translation. [101] first pre-train BART on a non-parallel multilingual corpus through denoising auto-encoding, and then fine-tune the pre-trained BART for

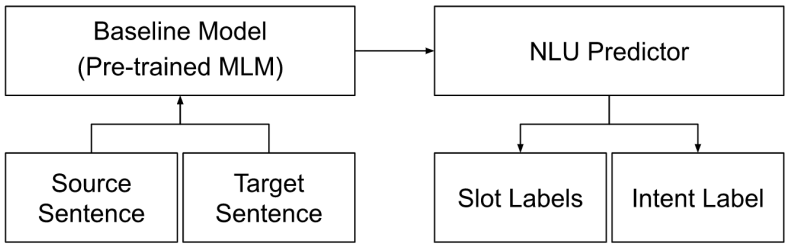

Figure 5.1: The NLU-oriented fine-tuning of our baseline model. MLM means multilingual language model.

downstream MT tasks.

5.3 Translation Aided Language Learner

5.3.1 Task Definition

An NLU task is a combination of slot filling and intent classification. Given a sentence x consisting of m words $\{w_1, \ldots, w_m\}$, slot filling is to predict a slot label y_i^σ for each word w_i, and intent classification is to predict an intent label y^ι for x. In this chapter, NLU models are required to be trained under a zero-shot CLTL scenario, where annotated sentences in the given source language are used for model optimization, while those in the given target language are used for model evaluation.

5.3.2 Baseline Model

A transformer [12] is a sequence-to-sequence model consisting of an encoder and a decoder. A main feature of transformers is that they use multi-head self-attention and multi-head cross-attention to model dependencies in sequential data. These attention mechanisms enable transformers to extract long-term contextual clues from text. As a result, transformers have been intensively used in transfer learning to develop pre-trained language models, which generate contextualized word embeddings. For example, some pre-trained language models, such as BERT [18] and RoBERTa [63], are implemented as transformer encoders, and some other ones, such as the GPT family [19, 20], are implemented as transformer decoders.

As a sub-field of transfer learning, CLTL has witnessed the wide application of transformers in developing pre-trained multilingual language models. Most of the existing pre-trained multilingual language models, such as M-BERT, XLM [91], and XLM-RoBERTa (XLM-R) [92], are implemented as transformer encoders. Actually, these pre-trained multilingual language models are the multilingual variants of BERT and RoBERTa, since each of them is identical to either BERT or RoBERTa except being pre-trained on a multilingual corpus. The representation space learned through this pre-training is not only rich in contextual clues but also shared by all the involved languages. Therefore, in theory, each of these pre-trained multilingual language models can be simply fine-tuned

to address any CLTL task between its involved languages.

The pre-trained multilingual language models mentioned above are now recognized as the state of the art in CLTL. To push the state of the art, we adopt one of them as our baseline model, and fine-tune it as an NLU model to conduct CLTL. As shown in Figure 5.1, in this NLU-oriented fine-tuning, we feed each given sentence to the baseline model, and feed the final hidden states of the baseline model to an NLU predictor. Since the baseline model is fitted with a sub-word tokenizer, a given sentence x consisting of m words $\{w_1, \ldots, w_m\}$ is tokenized into n tokens ($n \geqslant m$) such that the baseline model generates n final hidden states $\{h_1, \ldots, h_n\}$. For slot filling, the NLU predictor first performs an average pooling on the final hidden states related to each word w_i, and then uses a dense layer with a softmax normalization to map the pooling result to a slot distribution for w_i:

$$\mathrm{p}(y_i^{\sigma} \mid x) = \mathrm{softmax}\big(W^{\sigma}\mathrm{f_a}(h_{k_i}, \ldots, h_{l_i}) + b^{\sigma}\big)$$

where W^{σ} is a trainable weight, b^{σ} is a trainable bias, k_i and l_i separately represent the start position and end position of the final hidden states related to w_i, and $\mathrm{f_a}(\cdot)$ represents average pooling. For intent classification, the NLU predictor first performs an average pooling on all the final hidden states, and then uses another dense layer with another softmax normalization to map the pooling result to an intent distribution for x:

$$\mathrm{p}(y^{\iota} \mid x) = \mathrm{softmax}\big(W^{\iota}\mathrm{f_a}(h_1, \ldots, h_n) + b^{\iota}\big)$$

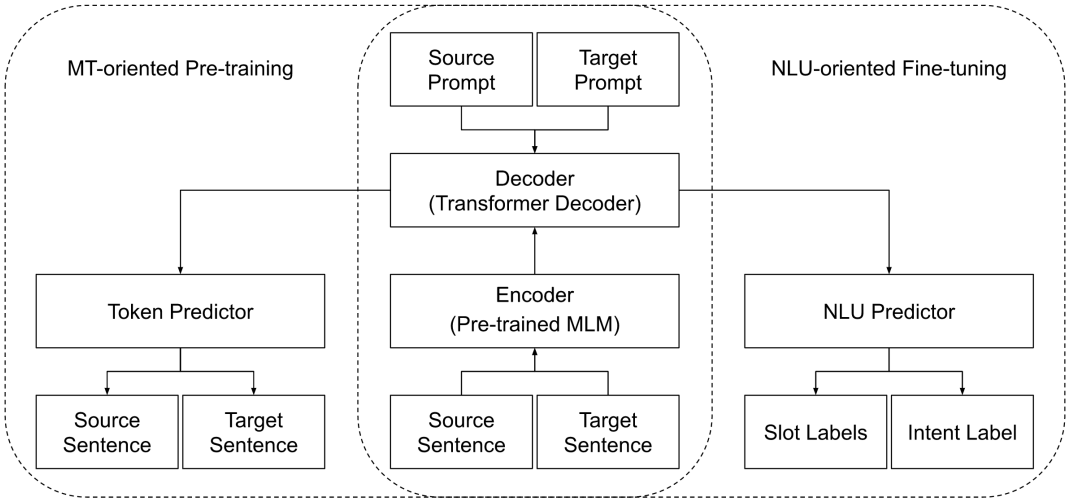

Figure 5.2: The MT-oriented pre-training and NLU-oriented fine-tuning of our proposed Translation Aided Language Learner (TALL).

where W^{ι} is a trainable weight, and b^{ι} is a trainable bias. On this basis, for model optimization, we minimize the following joint loss through stochastic gradient descent on annotated sentences in the given source language:

$$\mathcal{L}_{nlu} = -\log\big(\prod_{i=1}^{m} \mathrm{p}(y_i^{\sigma} \mid x) \cdot \mathrm{p}(y^{\iota} \mid x)\big)$$

For model evaluation, we infer the baseline model on annotated sentences in the given target language to measure three evaluation metrics, namely Slot F1, Intent Accuracy, and Semantic Accuracy (i.e. sentence-level joint accuracy).

5.3.3 Proposed Model

Since the baseline model is pre-trained on a multilingual corpus, all its involved languages are correlated with each other in its representation space. Normally, the larger such correlation between languages, the easier it is to conduct CLTL. To equally treat all possible CLTL tasks, the multilingual corpus used in the pre-training of the baseline model is collected in a subtle way that is not obviously biased to any language. However, there are two side effects of doing so. On the one hand, instead of focusing on a specific CLTL task, the baseline model pays equal attention to all possible CLTL tasks. On the other hand, in the representation space of the baseline model, the correlation between languages is proportional to their linguistic similarity, or in other words, similar languages are still similar to each other, and distant languages are still distant from each other. This implies that the CLTL ability of the baseline model can be pertinently improved for each given CLTL task, and the room for improvement is relatively large when the CLTL task is between distant languages.

To pertinently improve the CLTL ability of the baseline model for each given CLTL task, we would like to transform its representation space, which is used for all possible CLTL tasks, into a specialized one, where the correlation between the given source-target language pair is expressly enhanced. This goal can be achieved by resorting to MT, since translation is the most direct way to correlate languages with each other. As shown in

Figure 5.2, for MT to be workable, we treat the baseline model as an encoder and append a decoder to it. Considering that the encoder is implemented as a transformer encoder, we implement the decoder as a transformer decoder to keep the model architecture consistent. Besides, as in [12], we also share the token embeddings between the encoder and the decoder. The resulting new model can be seen as a standard transformer, where the encoder is a pre-trained multilingual language model. We expect this model to learn the correlation between the given source-target language pair by addressing a two-way MT task, and thus name it Translation Aided Language Learner (TALL).

Before conducting CLTL with TALL, we need to pre-train it as a two-way MT model that translates between the given source-target language pair. As shown in Figure 5.2, in this MT-oriented pre-training, we feed each given sentence to the encoder, feed a prompt for the translated sentence to the decoder, and feed the final hidden states of the decoder to a token predictor. Given a sentence x and a prompt x' for the translated sentence, suppose the decoder generates a final hidden state h'_i for the i-th token in x', then h'_i can be seen as a memory of both x and the first i tokens in x'. The token predictor uses a dense layer with a softmax normalization to map this memory to a token distribution for the position i in the translated sentence:

$$\mathrm{p}(y_i^\tau \mid x, x') = \mathrm{softmax}(W^\tau h'_i + b^\tau)$$

where W^τ is a trainable weight tied to the token embeddings, and b^τ is a trainable bias. Since two-way MT requires the translated sentence to be in either the source language

or the target language, which depends on the current direction, we extend the token vocabulary with two language identifiers, which separately represent the two languages, and thereby inform the decoder about the currently required language by setting the first token of x' to the corresponding language identifier. By the way, since the token vocabulary is highly multilingual, most probabilities in the above token distribution are for the tokens beyond the given source-target language pair and thus make no sense. Therefore, we ignore these probabilities when inferring TALL to generate translated sentences.

By convention, the training of MT models is supervised and thus requires parallel corpora. However, parallel corpora are generally expensive to collect, which makes them scarce or even unavailable for many source-target language pairs. Since TALL is designed to be a general-purpose CLTL model, a supervised training on parallel corpora is not applicable to its MT-oriented pre-training. Recently, an unsupervised training technique for MT models, which is named Unsupervised Machine Translation (UMT), has been proposed. Instead of relying on parallel corpora, UMT relies on monolingual corpora of unannotated sentences. This is attractive to us, since a large amount of unannotated sentences are always available for every language. Therefore, we implement the UMT training recipe proposed by [100] in the MT-oriented pre-training of TALL. Specifically, for model optimization, we collect a source-language corpus S and a target-language corpus T, each of which is a set of unannotated sentences. On this basis, we measure the following two losses:

- **Denoising auto-encoding loss.** As in [98], we implement a noise injector $\mathrm{f_n}(\cdot)$, which injects noise to each given sentence by randomly dropping and swapping its tokens. For each source-language sentence $s \in S$, we first run the noise injector to obtain a noise-injected sentence $\mathrm{f_n}(s)$, which can be seen as a sentence in a different language, and then use TALL to translate $\mathrm{f_n}(s)$ to the source language, the expected result of which is s. Besides, we also perform this process on each target-language sentence $t \in T$. This is the so-called "denoising auto-encoding", whose loss is defined as the cross-entropy loss on recovering the original sentences from the noise-injected sentences:

$$\mathcal{L}_{dae} = \mathbb{E}_{s \in S}\big[-\log \mathrm{p}\big(s \mid \mathrm{f_n}(s)\big)\big] + \mathbb{E}_{t \in T}\big[-\log \mathrm{p}\big(t \mid \mathrm{f_n}(t)\big)\big]$$

- **Back-translation loss.** Let us use $\mathrm{f_m}(\cdot)$ to represent the inference of TALL, which translates each given sentence to its opposite language in the given source-target language pair. For each source-language sentence $s \in S$, we first infer TALL to obtain a TALL-translated sentence $\mathrm{f_m}(s)$, which is in the target language, and then use TALL to translate $\mathrm{f_m}(s)$ to the source language, the expected result of which is s. Besides, we also perform this process on each target-language sentence $t \in T$. This is the so-called "back-translation", whose loss is defined as the cross-entropy loss on recovering the original sentences from the TALL-translated sentences:

$$\mathcal{L}_{bt} = \mathbb{E}_{s \in S}\big[-\log \mathrm{p}\big(s \mid \mathrm{f_m}(s)\big)\big] + \mathbb{E}_{t \in T}\big[-\log \mathrm{p}\big(t \mid \mathrm{f_m}(t)\big)\big]$$

We sum up the above two losses to obtain a joint loss, and thereby minimize the joint loss through stochastic gradient descent. For model evaluation, we collect another source-language corpus and another target-language corpus, each of which is also a set of unannotated sentences. On this basis, we implement the round-trip translation trick proposed by [98], where we first translate each given sentence to its opposite language in the current source-target language pair, and then translate the resulting sentence to the original language. By inferring TALL, we perform this process on all the sentences in the above two corpora to obtain two reconstructed corpora. Thereby, we measure the BLEU score between the two original corpora and the two reconstructed corpora to evaluate the translation performance of TALL.

The above MT-oriented pre-training guarantees that TALL can learn a representation space, where the given source-target language pair are expressly correlated with each other. As a result, it will be easier to conduct CLTL with the pre-trained TALL than with the baseline model. This is especially the case when the given source-target language pair are distant from each other, since translating between distant languages reveals more knowledge than translating between similar languages. However, considering that the representation space of TALL is co-carried by the encoder and the decoder, we have to fine-tune them together as an NLU model when we conduct CLTL with the pre-trained TALL. To this end, we implement the fine-tuning approach of BART [102] in the NLU-oriented fine-tuning of TALL. Specifically, as shown in Figure 5.2, we feed each given sentence to the encoder,

CLTL Task	Pre-trained MLM	BLEU Score	Slot F1			Intent Accuracy			Semantic Accuracy		
			BSL	TALL	Gain	BSL	TALL	Gain	BSL	TALL	Gain
EN-JA	M-BERT	41.19	56.78	60.87	7.20%	80.37	83.21	3.53%	14.56	16.39	12.57%
(distant)	XLM-R	39.83	58.21	63.19	8.56%	81.19	83.92	3.36%	16.58	18.47	11.40%
DE-JA	M-BERT	38.54	51.28	54.56	6.40%	79.08	81.54	3.11%	11.71	13.24	13.07%
(distant)	XLM-R	35.11	50.36	53.68	6.59%	78.43	81.12	3.43%	12.76	14.61	14.50%
EN-DE	M-BERT	71.21	70.42	72.39	2.80%	89.39	91.16	1.98%	36.53	38.64	5.78%
(similar)	XLM-R	72.91	75.29	77.14	2.46%	92.82	94.33	1.63%	44.86	47.25	5.33%

Table 5.1: The translation performance of TALL on Wikipedia and the CLTL performance of both the baseline model and TALL on MultiATIS++. EN means English. JA means Japanese. DE means German. BSL means the baseline model. Gain means the CLTL performance gain of TALL over the baseline model. The gain numbers are in percentage and calculated as $(\mathrm{TALL} - \mathrm{BSL}) \div \mathrm{BSL}$.

feed this sentence again as a prompt to the decoder with the corresponding language identifier prefixed to it, and feed the final hidden states of the decoder except the last one to the NLU predictor. On this basis, both the model optimization and the model evaluation remain the same as in the NLU-oriented fine-tuning of the baseline model.

CLTL Task	Pre-trained MLM	Slot F1 Gain	Intent Accuracy Gain	Semantic Accuracy Gain
EN-JA (distant)	M-BERT	53.61%	36.75%	59.45%
	XLM-R	50.96%	31.60%	71.86%
DE-JA (distant)	M-BERT	47.75%	31.46%	55.55%
	XLM-R	58.49%	34.45%	69.19%
EN-DE (similar)	M-BERT	10.42%	9.69%	18.00%
	XLM-R	12.75%	7.81%	23.87%

Table 5.2: The CLTL performance gain of TALL over the baseline model on the multi-domain multilingual NLU dataset.

5.4 Verification Experiments

5.4.1 Experimental Settings

CLTL tasks. For generality, we address not only CLTL tasks between distant languages but also those between similar languages. Specifically, we separately conduct CLTL between three source-target language pairs, which include two distant language pairs, namely English-Japanese and German-Japanese, and one similar language pair, namely English-German.

Pre-trained multilingual language models. For compatibility, we use different pre-trained multilingual language models for model construction. Specifically, for each given

CLTL task, we separately use two popular pre-trained multilingual language models, namely M-BERT (base and cased) and XLM-R (base), to construct both the baseline model and TALL.

Training data. For practicality, we adopt large-scale corpora and real-world datasets as training data. Specifically, to implement UMT, we collect a source-language corpus of 1M unannotated sentences and a target-language corpus of 1M unannotated sentences from Wikipedia dumps for model optimization, and also collect a source-language corpus of 10K unannotated sentences and a target-language corpus of 10K unannotated sentences from Wikipedia dumps for model evaluation. To conduct CLTL, we collect annotated sentences in real-world domains from two multilingual NLU datasets. The first multilingual NLU dataset is MultiATIS++ [40], which is an extension to Multilingual ATIS [39]. It provides 5K annotated sentences for each language, which are all in the domain of airline travel. The second multilingual NLU dataset is a multi-domain dataset collected from a virtual assistant. It provides 100K annotated sentences for each language, which are evenly distributed in five domains, namely music, notifications, smart home, weather, and books. By the way, in the above two multilingual NLU datasets, each word is annotated with a slot label in the B-I-O format, and each sentence is annotated with an intent label.

5.4.2 Implementation details.

We use WikiExtractor [103] to extract paragraphs from Wikipedia dumps, use Stanza [104] to split paragraphs into sentences, use HuggingFace's Transformers [105] to tokenize sentences into tokens and load pre-trained multilingual language models, and use PyTorch [106] to implement both the baseline model and TALL. For model optimization, we apply an AdamW optimizer [81] with an initial learning rate of 0.0001, a weight decay factor of 0.01, and a batch size of 64 in the MT-oriented pre-training of TALL, and apply another AdamW optimizer with an initial learning rate of 0.00005, a weight decay factor of 0.01, and a batch size of 256 in the NLU-oriented fine-tuning of both the baseline model and TALL. After each epoch, we evaluate the validation performance, which refers to BLEU score in the MT-oriented pre-training of TALL and Semantic Accuracy in the NLU-oriented fine-tuning of both the baseline model and TALL. If the obtained performance number is improved, we save the model, otherwise we cancel the finished epoch by restoring the model to the last saved version. We decay the learning rate by 0.5 after each cancelled epoch, and terminate the model optimization after the 5th cancelled epoch. For model evaluation, we use NLTK [55] to measure BLEU score, and use the evaluation script for the CoNLL-2000 shared task to measure Slot F1.

5.4.3 Experimental Results

As shown in Table 5.1, we carry out a series of experiments on the unannotated sentences collected from Wikipedia and the annotated sentences collected from MultiATIS++. Each of these experiments is aimed at a different combination of CLTL task and pre-trained multilingual language model, and includes the corresponding MT-oriented pre-training of TALL and the corresponding NLU-oriented fine-tuning of both the baseline model and TALL. On this basis, we first evaluate the translation performance of TALL in its MT-oriented pre-training, then evaluate the CLTL performance of both the baseline model and TALL in their NLU-oriented fine-tuning, and finally calculate the CLTL performance gain of TALL over the baseline model in percentage. Besides, as shown in Table 5.2, we also repeat the NLU-oriented fine-tuning of both the baseline model and TALL on the annotated sentences collected from the multi-domain multilingual NLU dataset, and thereby obtain another CLTL performance gain of TALL over the baseline model. The experimental results show that due to the application of UMT in the MT-oriented pre-training, TALL consistently achieves better CLTL performance than the baseline model in the NLU-oriented fine-tuning without using more annotated data, and the performance gain is relatively prominent in the case of distant languages.

5.4.4 Ablation Study

Denoising auto-encoding. In the MT-oriented pre-training of TALL, we try to discard the denoising auto-encoding loss and only minimize the back-translation loss in the UMT training. As a result, we observe that TALL achieves a very poor translation performance and a very poor CLTL performance. This implies that TALL learns little cross-lingual knowledge through the UMT training without denoising auto-encoding.

Back-translation. In the MT-oriented pre-training of TALL, we also try to discard the back-translation loss and only minimize the denoising auto-encoding loss in the UMT training. As a result, we observe that TALL achieves an almost perfect translation performance but a very poor CLTL performance. This is because the UMT training without back-translation makes TALL a copying model instead of an MT model, and a copying model can work perfectly in the model evaluation based on round-trip translation.

BART-style fine-tuning. In the NLU-oriented fine-tuning of TALL, instead of following the fine-tuning approach of BART, we try to discard the decoder and only fine-tune the encoder following the way we fine-tune the baseline model. As a result, we observe a very poor CLTL performance. This implies that the decoder of TALL is necessary for its NLU-oriented fine-tuning.

5.5 Further Discussion

Is a startup supervision necessary for the back-translation? In several existing UMT training recipes, the back-translation is supervised during its startup stage, where the supervision is provided by replacing the inference of TALL with a bilingual dictionary [98, 97]. This startup supervision is aimed at initializing a shared representation space for the given source-target language pair. However, since the encoder of TALL is a pre-trained multilingual language model, TALL already possesses a properly initialized representation space, which is shared by all the involved languages, and thus does not need a startup supervision. Actually, we tried to use a parallel corpus generated by a naive MT model to provide a startup supervision, which is equivalent to using a bilingual dictionary, but did not observe any translation performance gain.

How does the UMT training affect the CLTL performance? The UMT training uses the denoising auto-encoding and the back-translation to enhance the correlation between the given source-target language pair in the representation space of TALL. Since the encoder of TALL is a pre-trained multilingual language model, the representation space of TALL can be seen as an extension to that of the pre-trained multilingual language model. In the representation space of the pre-trained multilingual language model, similar languages have been more correlated with each other than distant languages. That is to say, in the representation space of TALL, there is more potential to enhanced the correlation

between distant languages than between similar languages. As a result, although the CLTL performance between similar languages is better than that between distant languages, the CLTL performance gain between distant languages is larger than that between similar languages.

5.6 Conclusion

The contribution of this chapter is three-fold. First of all, we construct a novel CLTL model TALL based on a pre-trained multilingual language model. In the next place, we train TALL to conduct CLTL through an MT-oriented pre-training and an NLU-oriented fine-tuning. Last but not least, we implement UMT in the MT-oriented pre-training of TALL to make use of unannotated data. Compared with the baseline model, which is the pre-trained multilingual language model used to construct TALL, TALL consistently achieves better CLTL performance without using more annotated data, and the performance gain is relatively prominent in the case of distant languages. The limitation of this work is that the unannotated sentences used for the UMT training are collected from Wikipedia, and thus not linguistically compatible with the downstream NLU tasks. In the future, we will use unannotated NLU sentences for the UMT training, which we believe can further boost the CLTL performance of TALL.

6 Conclusion

In this dissertation, we explore how to better perform neural-based knowledge transfer in NLP tasks, which is aimed at transferring out-of-domain (OOD) knowledge to task-specific neural networks. For structured OOD knowledge, we deal with Machine Reading Comprehension (MRC). In single-passage MRC tasks, we explicitly utilize the general knowledge extracted from knowledge bases to assist the attention mechanisms of MRC models, which makes MRC models less hungry for data and more robust to noise. In multi-hop MRC tasks, we transfer the knowledge in heuristically constructed graphs to multi-hop MRC models by applying GNN methods, where the strength of GNNs is probed through a comparison with pre-trained language models and filter-reader pipelines. For unstructured OOD knowledge, we deal with Natural Language Understanding (NLU). In Cross-Lingual Transfer Learning (CLTL) tasks for NLU, we perform cross-lingual knowledge transfer by applying Unsupervised Machine Translation (UMT) in an MT-oriented pre-training, which facilitates the CLTL in the following NLU-oriented fine-tuning, especially in the case of distant languages. Since both MRC and NLU are representative NLP tasks, we

believe that the neural knowledge transfer approaches proposed in this dissertation are also applicable to other NLP tasks.

Bibliography

[1] Rajpurkar, P., R. Jia, P. Liang. 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)*. 2018.

[2] Yang, Z., P. Qi, S. Zhang, et al. Hotpotqa: a dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018.

[3] Young, T., D. Hazarika, S. Poria, et al. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 2018.

[4] Mikolov, T., K. Chen, G. Corrado, et al. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations*. 2013.

[5] Mikolov, T., I. Sutskever, K. Chen, et al. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing*

Systems. 2013.

[6] Collobert, R., J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*. 2008.

[7] Kalchbrenner, N., E. Grefenstette, P. Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2014.

[8] Hochreiter, S., J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[9] Cho, K., B. van Merriënboer, D. Bahdanau, et al. 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*. 2014.

[10] Bahdanau, D., K. Cho, Y. Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*. 2015.

[11] Luong, M.-T., H. Pham, C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015.

[12] Vaswani, A., N. Shazeer, N. Parmar, et al. Attention is all you need. In *Advances in Neural Information Processing Systems*. 2017.

[13] Miller, G. A. Wordnet: a lexical database for english. *Communications of the ACM*, 1995.

[14] Speer, R., J. Chin, C. Havasi. Conceptnet 5.5: an open multilingual graph of general knowledge. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. 2017.

[15] Bollacker, K., C. Evans, P. Paritosh, et al. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. 2008.

[16] McCann, B., J. Bradbury, C. Xiong, et al. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*. 2017.

[17] Peters, M., M. Neumann, M. Iyyer, et al. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of*

the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). 2018.

[18] Devlin, J., M.-W. Chang, K. Lee, et al. 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)*. 2019.

[19] Radford, A., K. Narasimhan, T. Salimans, et al. Improving language understanding by generative pre-training. *Technical report, OpenAI*, 2018.

[20] Radford, A., J. Wu, R. Child, et al. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

[21] Nair, V., G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*. 2010.

[22] Bengio, Y., P. Simard, P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 1994.

[23] Weissenborn, D., T. Kočiskỳ, C. Dyer. Dynamic integration of background knowledge in neural nlu systems. *arXiv preprint arXiv:1706.02596*, 2017.

[24] Mihaylov, T., A. Frank. 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). 2018.

[25] Long, T., E. Bengio, R. Lowe, et al. World knowledge for reading comprehension: Rare entity prediction with hierarchical lstms using external descriptions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017.

[26] Richardson, M., C. J. Burges, E. Renshaw. 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*. 2013.

[27] Yang, Y., W.-t. Yih, C. Meek. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2015.

[28] Welbl, J., P. Stenetorp, S. Riedel. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 2018.

[29] Hermann, K. M., T. Kocisky, E. Grefenstette, et al. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 2015.

[30] Hill, F., A. Bordes, S. Chopra, et al. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*, 2015.

[31] Rajpurkar, P., J. Zhang, K. Lopyrev, et al. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016.

[32] Trischler, A., T. Wang, X. Yuan, et al. Newsqa: a machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. 2017.

[33] Dunn, M., L. Sagun, M. Higgins, et al. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*, 2017.

[34] Joshi, M., E. Choi, D. S. Weld, et al. 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)*. 2017.

[35] Nguyen, T., M. Rosenberg, X. Song, et al. Ms marco: a human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016)*. 2016.

[36] He, W., K. Liu, J. Liu, et al. Dureader: a chinese machine reading comprehension dataset from real-world applications. *arXiv preprint arXiv:1711.05073*, 2017.

[37] Kočiskỳ, T., J. Schwarz, P. Blunsom, et al. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 2018.

[38] Price, P. Evaluation of spoken language systems: The atis domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*. 1990.

[39] Upadhyay, S., M. Faruqui, G. Tür, et al. (almost) zero-shot cross-lingual spoken language understanding. In *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018.

[40] Xu, W., B. Haider, S. Mansour. End-to-end slot alignment and recognition for cross-lingual nlu. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020.

[41] Seo, M., A. Kembhavi, A. Farhadi, et al. Bidirectional attention flow for machine comprehension. In *Proceedings of the 5th International Conference on Learning Representations*. 2017.

[42] Wang, W., N. Yang, F. Wei, et al. 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)*. 2017.

[43] Yu, A. W., D. Dohan, M.-T. Luong, et al. Qanet: Combining local convolution with global self-attention for reading comprehension. In *Proceedings of the 6th International Conference on Learning Representations*. 2018.

[44] Jia, R., P. Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017.

[45] Clark, C., M. Gardner. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018.

[46] Hu, M., Y. Peng, Z. Huang, et al. Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2018.

[47] Liu, X., Y. Shen, K. Duh, et al. Stochastic answer networks for machine reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018.

[48] Xiong, C., V. Zhong, R. Socher. Dynamic coattention networks for question answering. In *Proceedings of the 5th International Conference on Learning Representations*. 2017.

[49] Huang, H.-Y., C. Zhu, Y. Shen, et al. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. In *Proceedings of the 6th International Conference on Learning Representations*. 2018.

[50] Chen, D., A. Fisch, J. Weston, et al. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2017.

[51] Liu, R., J. Hu, W. Wei, et al. Structural embedding of syntactic trees for machine comprehension. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017.

[52] Shen, Y., P.-S. Huang, J. Gao, et al. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017.

[53] Wang, W., M. Yan, C. Wu. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018.

[54] Yang, Z., J. Hu, R. Salakhutdinov, et al. Semi-supervised qa with generative domain-adaptive nets. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2017.

[55] Bird, S. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*. 2006.

[56] Pennington, J., R. Socher, C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014.

[57] Kim, Y. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014.

[58] Honnibal, M., I. Montani. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To Appear*, 2017.

[59] Abadi, M., P. Barham, J. Chen, et al. Tensorflow: a system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 2016.

[60] Kingma, D. P., J. Ba. Adam: a method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*. 2015.

[61] Srivastava, N., G. Hinton, A. Krizhevsky, et al. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.

[62] Salant, S., J. Berant. Contextualized word representations for reading comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 2018.

[63] Liu, Y., M. Ott, N. Goyal, et al. Roberta: a robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[64] Shao, N., Y. Cui, T. Liu, et al. Is graph structure necessary for multi-hop reasoning? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020.

[65] Tu, M., K. Huang, G. Wang, et al. Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020.

[66] Fang, Y., S. Sun, Z. Gan, et al. Hierarchical graph network for multi-hop question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020.

[67] Qiu, L., Y. Xiao, Y. Qu, et al. Dynamically fused graph network for multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019.

[68] Ye, D., Y. Lin, Z. Liu, et al. Multi-paragraph reasoning with knowledge-enhanced graph neural network. *arXiv preprint arXiv:1911.02170*, 2019.

[69] Zhao, C., C. Xiong, C. Rosset, et al. Transformer-xh: Multi-evidence reasoning with extra hop attention. In *Proceedings of the 8th International Conference on Learning Representations*. 2019.

[70] Ding, M., C. Zhou, Q. Chen, et al. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019.

[71] Yan, M., J. Xia, C. Wu, et al. A deep cascade model for multi-document reading comprehension. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 2019.

[72] Tan, C., F. Wei, N. Yang, et al. S-net: From answer extraction to answer synthesis for machine reading comprehension. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. 2018.

[73] Wang, S., M. Yu, J. Jiang, et al. Evidence aggregation for answer re-ranking in open-domain question answering. In *Proceedings of the 6th International Conference on Learning Representations*. 2018.

[74] Nishida, K., K. Nishida, M. Nagata, et al. Answering while summarizing: Multi-task learning for multi-hop qa with evidence extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019.

[75] Glass, M., A. Gliozzo, R. Chakravarti, et al. Span selection pre-training for question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020.

[76] Grail, Q., J. Perez, E. Gaussier. Latent question reformulation and information accumulation for multi-hop machine reading. *OpenReview*, 2019.

[77] Dhingra, B., M. Zaheer, V. Balachandran, et al. Differentiable reasoning over a virtual knowledge base. In *Proceedings of the 8th International Conference on Learning Representations*. 2020.

[78] Buzan, T., B. Buzan. *Mind Map Book*. Pearson Education, 2006.

[79] Battaglia, P. W., J. B. Hamrick, V. Bapst, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[80] Ferragina, P., U. Scaiella. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. 2010.

[81] Loshchilov, I., F. Hutter. Decoupled weight decay regularization. In *Proceedings of the 7th International Conference on Learning Representations*. 2019.

[82] van der Ploeg, T., P. C. Austin, E. W. Steyerberg. Modern modelling techniques are data hungry: a simulation study for predicting dichotomous endpoints. *BMC Medical Research Methodology*, 2014.

[83] Yarowsky, D., G. Ngai, R. Wicentowski. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the 1st International Conference on Human Language Technology Research*. 2001.

[84] Zhang, M., Y. Liu, H. Luan, et al. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2017.

[85] Conneau, A., G. Lample, M. Ranzato, et al. Word translation without parallel data. In *Proceedings of the 6th International Conference on Learning Representations*. 2018.

[86] Artetxe, M., G. Labaka, E. Agirre. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018.

[87] Chen, X., Y. Sun, B. Athiwaratkun, et al. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*, 2018.

[88] Chen, X., C. Cardie. Unsupervised multilingual word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018.

[89] Chen, X., A. H. Awadallah, H. Hassan, et al. Multi-source cross-lingual model transfer: Learning what to share. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019.

[90] Mulcaire, P., J. Kasai, N. A. Smith. Polyglot contextual representations improve crosslingual transfer. 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). 2019.

[91] Conneau, A., G. Lample. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*. 2019.

[92] Conneau, A., K. Khandelwal, N. Goyal, et al. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020.

[93] Pires, T., E. Schlinger, D. Garrette. How multilingual is multilingual bert? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019.

[94] Wu, S., M. Dredze. Beto, bentz, becas: The surprising cross-lingual effectiveness of bert. 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)*. 2019.

[95] Karthikeyan, K., Z. Wang, S. Mayhew, et al. Cross-lingual ability of multilingual bert: An empirical study. In *Proceedings of the 8th International Conference on Learning Representations*. 2020.

[96] Witte, A., T. Harden, A. R. de Oliveira Harden. *Translation in Second Language Learning and Teaching*. Peter Lang, 2009.

[97] Artetxe, M., G. Labaka, E. Agirre, et al. Unsupervised neural machine translation. In *Proceedings of the 6th International Conference on Learning Representations*. 2018.

[98] Lample, G., A. Conneau, L. Denoyer, et al. Unsupervised machine translation using monolingual corpora only. In *Proceedings of the 6th International Conference on Learning Representations*. 2018.

[99] Yang, Z., W. Chen, F. Wang, et al. Unsupervised neural machine translation with weight sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018.

[100] Lample, G., M. Ott, A. Conneau, et al. Phrase-based neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018.

[101] Liu, Y., J. Gu, N. Goyal, et al. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 2020.

[102] Lewis, M., Y. Liu, N. Goyal, et al. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020.

[103] Attardi, G. Wikiextractor. *GitHub*, 2015.

[104] Qi, P., Y. Zhang, Y. Zhang, et al. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 2020.

[105] Wolf, T., L. Debut, V. Sanh, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[106] Paszke, A., S. Gross, F. Massa, et al. Pytorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*. 2019.