Multi-hop Question Answering with Link Completed Graph

Yangguang Li[1]

MSc Computational Statistics and Machine Learning

Supervisor: Prof. Philip Treleaven

September 2019

[1]**Disclaimer:** This report is submitted as part requirement for the MSc degree in Computational Statistics and Machine Learning at University College London. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Abstract

Question answering (QA) requires the system to answer a given question by choosing a candidate among a given list, extract the answer from the given text, or recall from the system's knowledge bases. Previously researchers were focusing on answering questions with only one single supporting text, while in the real world, people usually find the answer for a particular question by looking through and combining information from several sources. Hence recently, more researchers are putting efforts into tacking multi-hop QA (MHQA): questions that can only be answered by combining factoid information from multiple sources. Among them, some researchers have successfully used the relational graph and contextual embedder to achieve state-of-the-art (SOTA) result.

This dissertation investigates the possibility of utilizing link prediction on graphs as an auxiliary task of MHQA problem, i.e. whether the performance on MHQA tasks can be improved by completing the graphs used for inference. This is important because graphs are known to be incomplete, and filling in those missing information by link prediction may help to capture the reasoning chains in the propagation steps in graphs. Though people have achieved near human-level performance in single-document QA, most of the architectures designed for that setting fall behind in the multi-document setting. Some researches have started to tackle this problem, but they are still far from achieving human-level performance, not to mention solving it. One of the reasons that stops graph-based methods to get better performance maybe the missing of information in the graphs.

This research comprises building models with and without a link predictor and then comparing the results of them to verify the effect of performing link prediction on MHQA problems. Both of the models consists of an encoder to summarise the local context of named entities, the graph constructor that build the entity graph out of the supporting documents, a message propagation network to accumulate evidence in the nodes' representations, and an output network to predict the answer. For the model with link predictor, the predictor is inserted between the graph constructor and message propagation network.

The implementation utilises mixed precision training to enable the graphs built to be processed by GPU. Implementing them this way shortens the running time significantly comparing to the papers which did their experiments on CPU [41, 9]. Also, mixed precision training allows the algorithm to process some graphs that are too large to put in the memory of GPU.

The testing is done with the QAngaroo WikiHop [54] dataset, which is explicitly designed to challenge the algorithm to aggregate evidence among passages before it can predict the correct answer. The dataset contains 43738 training instances and 5129 development instances. As the testing set is not published, the evaluation is done on the development set only, i.e. the development set is blinded.

Contents

List of Figures 3

List of Tables 4

1 Introduction 5

1.1 Problem and Motivation 5

1.2 Objective 6

1.3 Dataset 7

1.4 Structure of the Dissertation 8

2 Background and Related Work 9

2.1 Background 9

2.1.1 Natural Language Processing 9

2.1.2 Graph Neural Network 15

2.1.3 Link Prediction 19

2.2 Related Work 23

2.2.1 MHQA on Knowledge Bases and Sentences 23

2.2.2 MHQA with Attention-Based Methods 23

2.2.3 MHQA with Graph-Based Methods 24

3 Methodology 26

3.1 Model Design 26

3.1.1 Model Without Link Predictor 26

3.1.2 Model with Link Predictor 29

3.2 Implementation 31

3.3 Testing 31

3.4 Contribution 32

4 Experimental Results **33**
4.1 Set-Up of Experiments 33
4.2 Results 34
4.3 Assessment 36

5 Conclusion **37**
5.1 Summary 37
5.2 Future Work 38
5.2.1 Encoder 38
5.2.2 Deeper GNN 38
5.2.3 Link Prediction 39
5.2.4 HotpotQA 39

A Codes **40**

Bibliography **41**

List of Figures

1.1 An example from the WikiHop dataset 8

2.1 BiLSTM 12
2.2 Skip-gram 14
2.3 Knowledge Graph 18
2.4 The Link Prediction Model with R-GCN 22

4.1 Training loss for model without link predictor 34
4.2 Training loss for model with link predictor 35

List of Tables

3.1 Statistics of Entity Graphs Built 30

4.1 Accuracies for each model 35

Chapter 1

Introduction

This chapter presents an overview of the dissertation. It starts with introducing the research topic and state the motivation of this project. Then the objective of this project is further explained, which is followed by the discussion of the dataset and the structure of this dissertation.

1.1 Problem and Motivation

Question answering is about building a system that can read in a textual corpus, reason on it, and achieve a conclusion. It is a longstanding goal in the fields of information retrieval and natural language processing (NLP). With such a system, we will be able to input text files, audio, or videos to the system, and get the answer to questions concerning those files that have not been processed manually.

A typical way to assess such a system is by evaluating its performance on question answering datasets. Such a dataset is explicitly prepared as a set of questions $\{A\}$, where each question A consists of a query q and the correct answer $\bar{c}$. Sometimes there may also be supporting document(s) $\{d\}$ and a list of candidates $\{c\}$ being provided. The system is required to answer the query based on the supporting document(s) or some common knowledge/knowledge bases it has in itself. If the candidates are provided, the system needs to pick one of them; otherwise, it needs to extract the correct answer of free-form text from the whole span of the given text.

Previous research mainly focused on QA based on only a single document or paragraph. Boosted by the availability of large-scale datasets like SQuAD [38] and CNN/Daily Mail [20] in which most questions can be answered by attending the information in only one single sentence, many end-to-end neural models [42, 43] have been proposed and achieved good performances on these datasets. To overcome the limit of these datasets [53]: requiring only the information from one single sentence to answer the questions, people have proposed NarrativeQA [27], CoQA [39], TriviaQA [24], and RACE [28] which have questions that can only be answered if information from multiple sentences within the same document is gathered together. Though being more challenging, the several sentences for each document are still strongly related as there is usually some logical flows among the sentences in the same document.

In the real world, when given a question, people usually read up sources of different topics, written by people of various backgrounds and for audiences of diverse interests. Hence the capability to combine sources that are so heterogeneous is desired for QA systems. With such a system, we are able to input unstructured data that remain unprocessed to the system and ask arbitrary questions of interest. Attracted by this topic, a dataset named QAngaroo [54] which will be discussed in more details later is explicitly constructed for tackling multi-document QA problems. Researchers started from simply concatenating the documents into one and then applied the architectures, namely attention-based methods [42, 53], developed for single-document QA. The results were promising but far from human-level performance.

Another branch of research on this topic utilized the graph extracted from the query and its supporting documents [9, 45, 49]. They then propagated the information along the edges extracted with Graph Neural Network (GNN) [64] namely Graph Convolutional Network [26], Graph Recurrent Network, and Graph Attention Network. This research suggests that the reasoning chain could be captured by the propagation steps. This kind of approaches has improved the performance on QAngaroo WikiHop and remain the SOTA.

However, the performances achieved by these graph-based methods are still far from human-level. One factor that led to the deficiency may be the missing of information from the constructed graph as the way how the graphs were constructed is quite limiting: the nodes are the exact match of candidates and query entity in the supporting documents, and the edges are just four kinds predefined relationships between those named entities.

1.2 Objective

The research objective is to investigate whether the MHQA performance on QAngaroo WikiHop dataset can be improved by introducing link prediction into the graph-based method, given the hypothesis that this method is limited by the missing of information from the constructed graphs. To verify the hypothesis, the results of the methods with/without link prediction between graph construction and information propagation are compared.

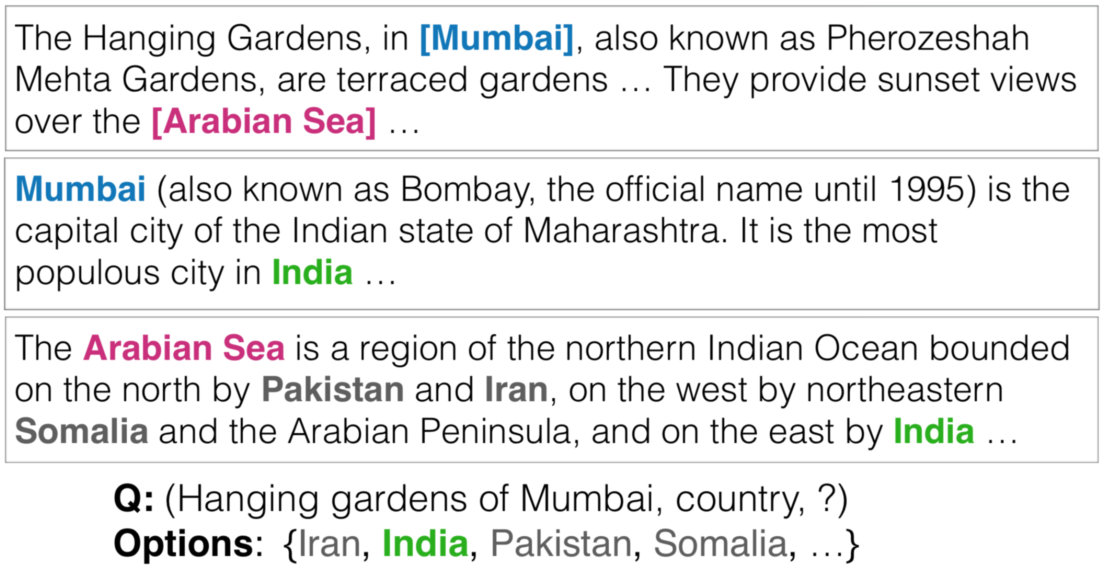
1.3 Dataset

Although remaining challenging, [27, 39, 24, 28] are created for reasoning within the same document, which is not the case for many real-world applications that require aggregating information from multiple documents, or even from multimodal sources. In this dissertation, we only focus on text question answering, hence only tackling the case of multi-document QA. Seeing the need for a dataset to facilitate research in MHQA, Welbl et al. released a new dataset named QAngaroo [54] which consists of WikiHop and MedHop.

The dataset requires the system to reason across multiple supporting documents before it can achieve the conclusion to pick the right answer among the candidates. WikiHop consists of questions generated from the user-created multi-domain unstructured text corpus Wikipedia and the structured information set Wikidata. All the information needed for answering a particular query is contained in the supporting documents, but there may also be some misleading documents which penalise the systems that perform only exact matches without comprehending the context in the reasoning procedure.

In Figure 1.1, we show an example of WikiHop dataset, where the system needs to combine information from several documents to get the correct answer to the question. We can see the appearance of misleading documents in the example. In the first document, we get *Mumbai* being adjacent to the *Arabian Sea*. Whereas in the last document, we find that *Arabian Sea* is closer to *Pakistan* and *Iran* than the correct answer, *India*, does in terms of the relational positions in the text. In such a scenario, a system that does not have a semantic understanding of the context may fall into the trap and pick *Iran* or *Pakistan* as the answer.

The MedHop dataset in QAngaroo is about finding the drug that will interact with a given drug according to the provided text from PubMed, a dataset of biomedical literature. The supporting documents are abstracts of research papers, and al candidates are drugs' names extracted from the whole text corpus. This dataset is much smaller comparing to WikiHop: 2,508 instances versus the 51,318 ones of the WikiHop. Also, the text is closed-domain now, hence the requirement on the ability to do reading comprehension is lower than that of WikiHop. Therefore we are only focusing on WikiHop instead of MedHop.


The Hanging Gardens, in **[Mumbai]**, also known as Pherozeshah Mehta Gardens, are terraced gardens … They provide sunset views over the **[Arabian Sea]** …

Mumbai (also known as Bombay, the official name until 1995) is the capital city of the Indian state of Maharashtra. It is the most populous city in **India** …

The **Arabian Sea** is a region of the northern Indian Ocean bounded on the north by **Pakistan** and **Iran**, on the west by northeastern **Somalia** and the Arabian Peninsula, and on the east by **India** …

Q: (Hanging gardens of Mumbai, country, ?)
Options: {Iran, **India**, Pakistan, Somalia, …}

Figure 1.1: An example from the WikiHop dataset. Reprinted from [54].

1.4 Structure of the Dissertation

Structure of this dissertation is as follows,

- Chapter 2 reviews the background in several key areas including NLP, GNNs, and link prediction. Then the related work is briefly described.
- Chapter 3 articulates the models to be compared and then discusses the implementation and testing plan for the project.
- Chapter 4 presents the set-up of the experiments and the experimental results with the assessment of them.
- Chapter 5 concludes this dissertation and its contribution. Then a few possible future works are discussed.

Chapter 2

Background and Related Work

This chapter starts with introducing the background knowledge needed to understand the related work. Then it goes to describe the literature related to my work briefly.

2.1 Background

This section includes critical concepts needed in NLP, GNNs, and link prediction.

2.1.1 Natural Language Processing

NLP is a subfield of computer science concerned with processing and analyzing a large amount of natural language data. It can be used to translate a novel in English to Chinese or generate responses to customer enquiries automatically. Inspired by the famous paper of Alan Turing on the Turing test, researchers have put many efforts into solving NLP problems relating to:

- Syntax: part-of-speech tagging, lemmatization, and parsing;
- Semantics: machine translation, named-entity recognition (NER) [46], natural language understanding, QA [2], and textual entailment;
- Discourse: coreference resolution [57].

In the early days, many NLP systems were constructed with hand-crafted rules, which imposed a consequential limit on their abilities: the systems were not able to handle the variations that had not been encoded into the system by a human. In order to equip those systems with the capacity for solving cases they have not encountered, researchers introduced machine learning (ML), a field that studies how to automate computer programs, into the systems.

Neural Network

ML studies how to find patterns in data efficiently and exploit the finding to perform a specific task automatically. ML algorithms assume a mathematical model over the data and then learn the parameters and infer the latent variables from the data. There are various families of ML models ranging from linear regression to Gaussian process available for solving different problems. In the field of NLP, one of the most important kinds of data is unstructured data which is text-heavy. In order to process them, we usually treat those data as sequential data and use Markov models such as n-gram model which assumes that the next word/letter statistically depends only on the past n words:

$$p(x_i | x_{i-(n-1)}, \dots, x_{i-1})$$

This assumption considerably simplified the search space of the language model and led to many successful applications.

More recently, Recurrent Neural Network (RNN) [16], an extension of standard multi-layer perceptron, had been proposed to deal with sequential data and time-series data. The network maintains a hidden state that can change over steps, and takes in input, like tokens or price of the stock, one at a time. Then it applies a learned transformation to the concatenation of input and hidden state to produce the output and hidden state for the next step:

$$\begin{aligned} h_t &= \sigma(W^x x_t + W^h h_{t-1} + b) \\ &= \sigma(W[x_t; h_{t-1}] + b), \\ o_t &= \tau(h_t), \end{aligned}$$

where h_t, h_{t-1} are hidden states at the end of steps t and $t-1$, x_t is the input at step t, W, b are weights to learn, and σ, usually *tanh* or *relu*, is the non-linearity applied. Usually chosen as identity mapping, τ is the function that maps hidden state to output. As a general function approximator, RNN is more powerful then n-gram models and can theoretically process sequences, i.e. sentences, of arbitrary size.

However, at the time when RNN was proposed, people were suffering from the inefficiency of computing power that was needed to train the network with Backpropagation Through Time (BPTT). Even if computing BPTT was feasible, they still faced the problem of gradient exploding and gradient vanishing. To solve these problems, Hochreiter and Schmidhuber proposed Long Short-Term Memory (LSTM) network which introduced gating mechanism to control how much the hidden state, i.e. the memory, is updated given the input:

$$\begin{aligned}
h_{t-1}, c_{t-1} &= s_{t-1}, \\
z_t &= [x_t; h_{t-1}], \\
i_t &= \sigma(W^i z_t + b^i), \text{ input gate} \\
f_t &= \sigma(W^f z_t + b^f), \text{ forget gate} \\
o_t &= \sigma(W^o z_t + b^o), \text{ output gate} \\
c_t &= f_t \odot c_{t-1} + i_t \odot tanh(W^c z_t + b^c), \\
h_t &= o_t \odot tanh(c_t), \\
s_t &= [h_t; c_t],
\end{aligned}$$

where h_t and c_t is the hidden state and cell state at step t, x_t is the input at step t, σ is the sigmoid function, and all $W^{\cdot}, b^{\cdot}$s are the weight to be learned.

Given the chance that the LSTM can retain some of the memory instead of updating it completely as did in vanilla RNN, it introduces a 'short-cut' for the backpropagation of gradient and gets rid of the exploding/vanishing gradient caused by the weight matrix having its largest eigenvalue being not equal to one. In this dissertation, I have used Bidirectional LSTM (BiLSTM) to encode the query. Proposed by Graves et al. in 2013 [18], this model adds a pipeline that processes the sequence in the reversed order as shown in Figure 2.1; hence it can combine not only the information from the past but also from the future for each token. Empirically this bidirectional mechanism usually provides a better result than its unidirectional counterpart.

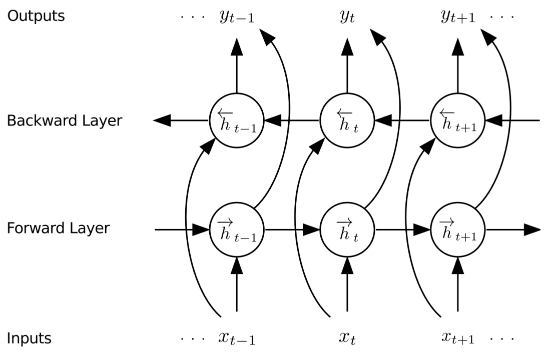


Figure 2.1: BiLSTM. Reprinted from [18].

Loss

For most ML algorithms, we train the model by either maximising the likelihood or posterior or minimising an expected loss. In this work, the final metric is the accuracy of the model on the development set of WikiHop, and the way how the model makes a prediction for each instance is by choosing the node with the highest probability. Hence it seems like we can regard this problem as a classification problem, where we naturally go to cross-entropy (CE) loss:

$$loss(x, class) = -\log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right) = -x[class] + \log\left(\sum_j \exp(x[j])\right),$$

where x is a vector of logits produced by the output layer for each class i.e. each node in our problem, and *class* is the index of the correct node.

However, for a single instance, there may be more than one correct node, which indicates the answer has appeared in supporting documents for more than once. It is then hard to define the target vector. For CE loss, there should be only one correct class, Hence for the target vector, there is only one non-zero value on the position corresponding to the correct class. However, there would be multiple non-zero values in the target vector in our problem, and if treated equally, they will be $1/n$ given n nodes are corresponding to the answer. This way, we have a target that is changing its amplitude across instances and being inconsistent with the definition of CE loss.

Therefore I have used another loss function that is more suitable for this problem which may have multiple correct answers: Binary Cross-Entropy (BCE) loss. The loss function can be written as:

$$loss(x, y) = \square_{i=1}^{n}[y_n log(x_n) + (1 - y_n) log(1 - x_n)],$$

where x is the logits aforementioned, y is the target, n is the number of nodes, and $\square$ is the reduction function which can be mean or sum. This way, the target can be constructed as a vector of 0s and 1s, where the non-zero values are on the positions corresponding to the correct nodes. Now the problem has transformed into a number of subproblems: whether a particular node is a correct answer or not.

Optimisation

People use optimisation methods to find optimal or near-optimal solutions for problems. The problems in machine learning can be generalised as: given the hypothesised model over the dataset, what are the parameters that make the model optimally 'fits' the data. In order to find those parameters, researchers have developed various methods that can be categorised into three main kinds: zero-order methods, first-order methods, and second-order methods.

Among first-order methods, Stochastic gradient descent (SGD) is the most popular optimisation method being used. It randomly samples a mini-batch from the whole dataset and uses the gradient computed on that small batch to approximate the gradient. Therefore it generalises pretty well and may be able to get rid of some of the local minimums. However, one of the drawbacks of SGD is that it uses a fixed learning rate for all steps, which easily make the learning rate too big in the earlier stage and too small in the later stage. Moreover, the update variance of SGD is high because of the lack of momentum. It also converges slowly if local curvature is too small, where momentum may assist in convergence.

Another first-order optimisation method, Adaptive Moment Estimation (Adam) [25], is also commonly used. However, despite having faster convergence and lower variance, adaptive optimisation methods, such as Adam, has been found to generalize poorly comparing to SGD. These methods tend to perform well in the initial portion of the training but are outperformed by SGD at the later stage of training [56]. Since the batch size is very small in my work due to the limit of GPU memory, Adam was used to stabilising the training procedural.

Word Embedding

A technique in NLP where words are mapped to numerical vectors, word embedding has boosted the performance in many downstream tasks such as semantic analysis and machine translation. The relative locations of the vectors in the feature space seem to conform with the syntactic and semantic relationships among the corresponding words. Hence, this mapping effectively reduced the high dimensional space of words into continuous space with much lower dimensions.

In the early days, people use singular value decomposition to find the singular vectors of the term-document occurrence matrix, which is typically constructed with tf-idf. This method is called the Latent Semantic Analysis (LSA). More recently, word2vec [33], a research bases on neural network, was proposed by Mikolov et al. This method consists of two models of which both are trained in an unsupervised manner. The Continuous Bag of Word model trains the neural network to recover the masked word given the context around it. While the Skip-gram model, as shown in Figure 2.2, trains the network to recover the context given a single word. After training, people throw away the decoding part and only use the encoder to assist the downstream tasks. Word2vec helped many NLP problems to achieve better performances.

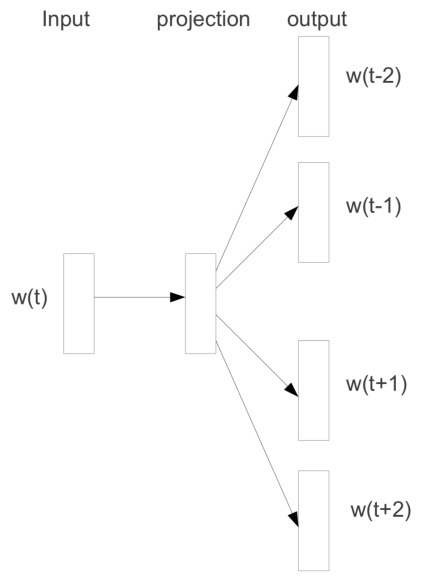


Figure 2.2: Skip-gram model. Reprinted from [33].

Though being very useful, both LSA and word2vec have the same limit: they do not take into account the context when encoding a word. In graph-based methods for solving MHQA problem, the initial node embeddings should encode not only the mentions themselves but also their context. The graph structure captures the global information in the supporting documents and their corresponding query but throws away all the text information. Hence, I would like to enhance the graph with the local context information contained in the text. In order to do so, ELMo [36] was deployed at the graph construction procedural. The model consists of character-level convolution and trained 2 BiLSTM layers to encode sentences on the fly. Therefore it solves the Out-of-Vocabulary (OOV) problem naturally and enables the network to utilise morphological clues. The word vectors from the network carry the context information passed by LSTMs forwardly and backwardly.

2.1.2 Graph Neural Network

Graph neural networks are NNs designed to process graph-structured data, or also called non-Euclidean data. Traditional NNs like fully-connected layer, CNN, and Recurrent Neural Network could not handle this kind of data, as each point in the dataset has a different shape. Every graph has a different number of nodes, and each node has a varying number of neighbours.

First proposed in 2008 [40], GNN was constructed to learn an embedding h_v for each node. The embedding should contain the information of its neighbourhood. To be able to do that, we have to learn a local mapping function that is sharing among all nodes for each kind of relationship (edge), and update the embedding given its neighbour. Let f be the local mapping function, and g be the output function, the propagation step of GNN can be defined as:

$$\begin{aligned} h_v &= f(x_v, x_{co[v]}, \bar{h}_{ne[v]}, \bar{x}_{ne[v]}) \\ o_v &= g(h_v, x_v) \end{aligned}$$

where x_v is the feature of node v, $x_{co[v]}$ is the feature of its edges; $\bar{h}_{ne[v]}$ and $\bar{x}_{ne[v]}$ are the embeddings and features of v's neighbour respectively. The output can be anything we care about, like node labels for node classification problem.

Although GNN was shown to be powerful in modelling relational data, it still has some limitations:

1. The iterative process for updating the embeddings to reach a fixed point is not efficient. Instead, we can relax the limitation and design an architecture that multiple layers of updates.

2. The parameters of the mapping functions are shared between each iteration. Instead, we can make each layer of propagation having its own parameters.

3. If the node embeddings do reach a fixed point, most of the embeddings would be quite similar, and it is hard to distinguish them. We would like each node's embedding contains the information of the neighbourhood but still remains unique.

In order to tackle these limitations and incorporate features that are suitable for modelling some particular kinds of data, researchers have proposed many variants of GNN including but not limited to Graph Convolutional Network (GCN), Graph Attention Network, Gated Graph Neural Network, GraphSAGE, ChebNet, and etc. The original GNN and its variants have achieved state-of-the-art performances in many areas including text classification, social relationship understanding, protein interface prediction, and knowledge base completion, just to name a few.

GCN

Graph Convolutional Network [26] is a successful adaptation of CNN into graph-structured data. In recent years, CNN has dominated almost all state-of-the-art leader boards and challenges in Computer Vision (CV). There are also some successful applications of CNN in NLP and speech. However, CNN is not suitable for processing non-Euclidean data, as the receptive field is hard to define due to the various number of neighbours each node has, i.e. CNN can not keep translation invariance which makes it so successful in CV.

Variants of GNN that use convolution to aggregate information can be categorised into two groups: spectral methods and non-spectral methods. The spectral methods work with a spectral representation of the graphs: the graph Laplacian. The most simple Laplacian, combinatorial Laplacian, is defined as $L = D - A$, where D is the degree matrix, and A is the adjacency matrix. However, most of the GCN papers utilise the concept of (symmetric) normalised Laplacian $\tilde{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ which extends Laplacian to irregular graphs. Once we get the normalised Laplacian, we can perform eigendecomposition on it: $\tilde{L} = U \Lambda U^\top$. The graph convolution operation is then defined in the Fourier domain, i.e. spectral domain, as

$$g_\theta \star x = U g_\theta(\Lambda) U^\top x,$$

where g_θ is the filter parameterised by $\theta \in R^N$ with $diag(\theta)$, and $x \in R^N$ is the signal. Hence $U^\top x$ is just the graph Fourier transform of x. However, this operation is quite computationally intensive, and the filter is not spatially localised.

In order to circumvent the problem, Defferrard et al. [10] proposed ChebNet where they approximated the filter with Chebyshev polynomials $T_k(x)$ up to order K:

$$g_\theta(\Lambda) \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{\Lambda}),$$

where $\tilde{\Lambda} = \frac{2}{\lambda_{max}}\Lambda - I$ and λ_{max} is the largest eigenvalue of L. The rescaled matrix $\tilde{\Lambda}$ has eigenvalues lie in $[-1, 1]$. This way, the introduced K-localised convolution removes the need to compute eigenvectors the Laplacian. Therefore we are able to stack multiple convolutional layers together to form a neural network.

Furthermore, Kipf and Welling [26] limits the layer-wise convolutional operation to $K = 1$ and approximates $\lambda_{max} = 2$ and get:

$$g_\theta \star x \approx \theta_0 x + \theta_1 (L - I)x = \theta_0 x - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x,$$

which eases the overfitting on local structures on graphs that have a wide range of node degrees. Finally, we get the layer-wise propagation rule in vector form:

$$H^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^l W^l),$$

where $\tilde{A} = A + I$ an adjacency matrix added with self-loops, $[\tilde{D}]_{ii} = \sum_j [\tilde{A}]_{ij}$ the degree matrix of the graph added with self-loops, H^l is the embedding at layer l, W^l is the layer parameter, and σ is the non-linearity.

Though shown to have a lot potential, spectral methods are limited by that their learning process required the whole graph structure via graph Laplacian, i.e. they only work on graphs that share the same Laplacian eigenbasis, and hence can not perform inductive inference on general graphs.

GraphSAGE [19] is a general inductive framework which generates the embeddings by sampling and aggregating information from a node's neighbourhood:

$$\begin{aligned} h^k_{\mathcal{N}(v)} &= AGGREGATE_k(\{h^{k-1}_u | \forall u \in \mathcal{N}(v)\}), \\ h^k_v &= \sigma(W^k \cdot [h^{k-1}_v || h^k_{\mathcal{N}(v)}]), \end{aligned}$$

where h^k_u is the embedding of node u at layer k, $\mathcal{N}(v)$ is the set of neighbours of node v, W^k is the parameter of layer k, $[\cdot||\cdot]$ is concatenation, and σ is the non-linearity. Note that $\{h^{k-1}_u | \forall u \in \mathcal{N}(v)\}$ is not the full-set neighbour but a fixed-size uniform sample. The sampling ensures that all instances in a batch are of the same size, and modifying the sample size accordingly can alleviate the computational and memory burden. The last thing that is worth attention is the $AGGREGATE_k$, a differential aggregator function, which can be the mean function, max-pooling after a fully-connected layer, or even an LSTM aggregator. When using the sum aggregator, GraphSAGE is an inductive variant of GCN.

Relational-GCN

All the GNNs mentioned above only deal with graphs that have a single relationship, i.e. single type of edge. However, there are many graph data involving multiple relationships. For instance, Figure 2.3 shows part of a knowledge graph involving four types of edges: 'played', 'characterIn', 'starredIn', and 'genre'. Intuitively the way how information is propagated through each kind of relationship should differ, but all the models mentioned in the previous section treat all the edges as the same and fail to distinguish them.

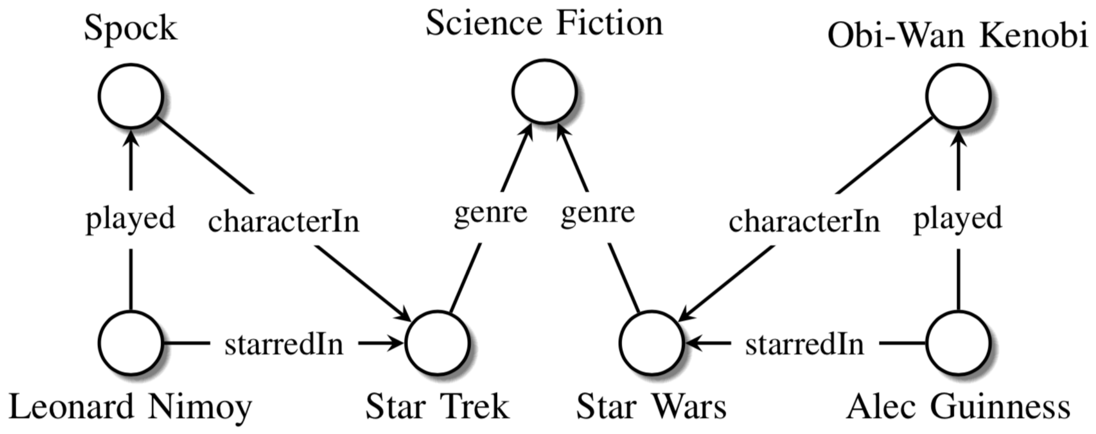


Figure 2.3: Knowledge Graph. Reprinted from [34].

Schlichtkrull et al. [41] have extended the inductive version of GCN onto multi-relational graphs. Let $\mathcal{R}$ be the set of relationships occur in a graph, the model's propagation rule is simply:

$$h_i^{l+1} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^l h_j^l + W_o^l h_i^l\right),$$

where $\mathcal{N}_i^r$ is the set of nodes linked to i with relationship r, $c_{i,r}$ is the problem-specific normalising constant, W_r^l and W_o^l are the parameters, h_j^l and h_i^l are the embeddings of nodes

j and i at layer l, and σ is the non-linearity. Therefore the model learns to propagation information differently in accordance with different relations.

However, as we can see from the update rule, there are $|\mathcal{R}|$ times more parameters in Relational-GCN (R-GCN) comparing to GCN, which significantly increases the risk of over-parameterisation. For mitigating the risk, regularisations including basis decomposition and block-diagonal decomposition are introduced. With the basis decomposition, each weight W_r^l is defined as:

$$W_r^l = \sum_{b=1}^{B} a_{rb}^l V_b^l,$$

where $B \leq |\mathcal{R}|$ is the number of bases, $V_b^l \in \mathbb{R}^{d^{l+1} \times d^l}$ is the b-th basis, and a_{rb}^l controls how the base are combined to form the parameters for relationship r. In block-diagonal decomposition, we have

$$W_r^l = \bigoplus_{b=1}^{B} Q_{br}^l,$$

a direct sum over a set of lower-dimensional matrices $\{Q_{br}^l \in \mathbb{R}^{(d^{l+1}/B) \times (d^l/B)}\}$.

2.1.3 Link Prediction

Given an arbitrary graph, Link Prediction (LP) tries to find out whether there are links between two or more entities. As graphs are omnipresent in real-world, LP has a wide range of applications including friend recommendation, product recommendation, knowledge base completion, and crime prevention by predicting missing links in the crime network. In the early days, researchers have developed a bunch of straightforward yet competent heuristics for evaluating the similarity of nodes such as common neighbours, preferential attachment [5], Adamic-Adar [1], resource allocation [65], and SimRank [23]. Based on how similar two nodes are, we can then set or train a threshold to determine if there is a link between them.

All the methods mentioned above utilizes the local or global graph structures, but do not take into account the differences between nodes and the type of the edges. In order to encode useful information other than the graph structures, people have proposed methods incorporating graph embedding. This kind of methods can work on graphs with node embedding initialised with information retrieval or NLP methods like word vectors [33]. With word embedding, the contextual information is projected into a lower-dimensional space at where the distance/relationship between two nodes can be easily resolved. Below some of these methods are introduced and compared.

TransE

Bordes et al. [7] proposed to model entities and relationships in the same embedding space $\mathbb{R}^d$. Therefore all the vectors can operate with each other directly, and the relationship is just a vector that linearly interacts with the vectors of entities. Given a triplet (s, r, t) from the knowledge base representing nodes s and t are related by edge r, we want to have $\mathbf{s} + \mathbf{r} \approx \mathbf{t}$, i.e. $\mathbf{t}$ is the nearest neighbour of $\mathbf{s} + \mathbf{r}$. Otherwise, if the triplet is from negative sampling, we want them to be far apart. Note that all the entity vectors are unit vectors to narrow the embedding space. For learning such embeddings, a margin-based ranking loss is minimized over the train set:

$$\mathcal{L} = \sum_{(s,r,t)\in S} \sum_{(s',r,t')\in S_{(s,r,t)}} [\gamma + d(\mathbf{s} + \mathbf{r}, \mathbf{t}) - d(\mathbf{s}' + \mathbf{r}, \mathbf{t}')]_+,$$

where S is the knowledge base, $S_{(s,r,t)}$ is the set of negative samples of (s, r, t) where the source or target is corrupted, γ is the margin we want between the positive and negative samples, $d(a, b)$ is the metric like Manhattan or Euclidean distances defined in $\mathbb{R}^d$, and $[x]_+ = max(0, x)$.

There are two extensions of TransE: TransH [52] learns different entity embeddings for a different kind of relationship, hence the goal is simply to make $\mathbf{s}_r + \mathbf{r}$ and $\mathbf{t}_r$ close to each other for positive samples and far away otherwise; TransR [31] learns a project operator M_r for relation r to project $\mathbf{s}$ and $\mathbf{t}$ onto $\mathbf{s}_r$ and $\mathbf{t}_r$, hence it does not need to keep $N \times |\mathcal{R}|$ distinct embeddings but just N embeddings and $|\mathcal{R}|$ operators.

However, the relations considered in these approaches are all linear operators represented by one-dimensional vectors which are very limiting. The model lacks the ability to model the interaction between entities. Hence some other researchers have proposed to model the relations bilinearly.

NTN

Socher et al. [44] proposed to use Neural Tensor Network (NTN) to do knowledge base completion. In NTN, one node's embedding is initialised by applying mean pooling to the word vectors corresponding to the entity of that node and so the embedding is not a unit vector like anymore. The scoring function, a bilinear tensor layer, is defined as:

$$g(s,r,t) = u_r^\top \sigma\left(s^\top W_r^{[1:k]} t + V_r \begin{bmatrix} s \\ t \end{bmatrix} + b_r\right),$$

where σ is the non-linearity which was *tanh* chosen by the authors, $W_r^{[1:k]} \in \mathbb{R}^{d\times d\times k}$ is a tensor such that $s^\top W_r^{[1:k]} t$ results in a vector in R^k, $V_r \in \mathbb{R}^{k\times 2d}$, and $u_r, b_r \in \mathbb{R}^k$. The objective is to train the scoring function so that it returns 1 for positive samples and -1 otherwise. This way, the model is capable of modelling the interaction between entity vectors and the linear matrix operator and non-linearity further enhance the model's expressiveness. Nonetheless, the model now suffers from the risk of over-parameterisation and higher computational burden.

DistMULT

DistMULT [59] was proposed by Yang et al. to simplify the scoring function of NTN yet achieve similar or even better performance in link prediction. The non-linearity and linear matrix in NTN are removed, and the bilinear tensor is replaced by a bilinear matrix such that the scoring function is now:

$$g(s,r,t) = \mathbf{s}^\top W_r \mathbf{t},$$

where (s,r,t) is the triplet representing source node s, relation (edge) r, and target node t. Similar to NTN, the scoring function is trained to produce 1 for positive samples and 0 for negative samples. To facilitate training, the margin-based ranking loss, which was used by TransE, is incorporated into the model. Also, note that the entity representations **s** and **t** can be n-hot vectors or vectors produced by word embedding techniques which capture local context to aid the inference on the global graph structure. Nevertheless, people have found that DistMULT works pretty well on standard graph data, but is not good at modelling asymmetric relations. To deal with that, people have introduced complex number into the embeddings and parameters, which shows prominent improvements over DistMULT on graph-structured data with asymmetric relations.

R-GCN

As detailed above, R-GCN is a GNN model that propagate information in multi-relational graphs. The ability to accumulate evidence through several inference steps on the graph can further improve the factorisation models for link prediction like DistMULT, NTN, and TransE described before. Schlichtkrull et al. chained an R-GCN as the encoder with a scoring function as the decoder to get a graph auto-encoder as shown in Figure 2.4. Evaluation on WN18, FB15K, and FB15K-237 datasets have shown improvements over other link prediction models without the encoder.

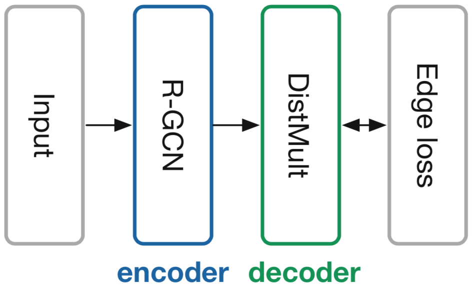


Figure 2.4: The Link Prediction Model with R-GCN. Reprinted from [41].

ConvE

Dettmers et al. [11] hypothesised that with a much deeper model, the learned feature might be more expressive than those learned with shallower models. Hence they proposed to use a multi-layer 2D convolutional neural network to perform link prediction tasks. The scoring function of their model is:

$$g(s, r, t) = \sigma(vec(\sigma([\bar{\mathbf{s}}||\bar{\mathbf{r}}] * \omega))W)\mathbf{t},$$

where σ is the non-linearity which was ReLU in the paper, $\mathbf{s}, \mathbf{t}$, and $\mathbf{t} \in \mathbb{R}^d$ are the entity vectors and relation vector respectively, $\bar{\mathbf{s}}, \bar{\mathbf{r}} \in \mathbb{R}^{d_w \times d_h}$ where $d_w \times d_h = d$ are the 2D reshaped $\mathbf{s}$ and $\mathbf{r}$, ω is a convolutional filter, and W is the layer projection parameter. Unlike all factorisation models mentioned above, which used margin-based ranking loss, a BCE loss is minimised over all edges (positive and negative) to train the model. As the convolutional operation is far more parameter efficient than linear and bilinear operations, the ConvE model produces a similar performance to DistMULT and R-GCN with 8 times and 17 times fewer parameters, i.e. it significantly soothes the risk of over-parameterisation.

2.2 Related Work

2.2.1 MHQA on Knowledge Bases and Sentences

Many existing works on multi-hop reasoning focused on hopping over knowledge bases [22, 62] which do not require the system to be able to comprehend text but only known relations with a well-defined structure. In order to inspire the creation of such a system that can perform multi-hop QA on textual data, many datasets which require the model to reason across multiple sentences including bAbI [55], OpenBookQA [32], CBT [21], and MultiRC [43] were introduced. Many systems [48, 58] have been proposed to solve these multi-hop reading comprehension tasks. However, the problem we are interested in is to aggregate information from multiple documents, since the most popular data interspersed in the real world are unprocessed, textual documents but not sentences or knowledge bases.

One line of approaches that deal with multiple passages tries to select the most relevant document first before performing reading comprehension to answer the question [14, 15]. Nonetheless, this kind of methods fail to accumulate information across documents, but reduce the problem into two easier problems: a classification problem to choose the most relevant document, and then a reading comprehension problem to answer the question.

To address the problem of combining information from different documents before making the decision, a few approaches [51, 30] have been proposed to aggregate the contexts of the same candidate in different passages to cover various aspect of the question. However, these approaches fail in combining evidence from the contexts of different entities which could be of concern for answering the question properly.

2.2.2 MHQA with Attention-Based Methods

Neural attention mechanism [4] trains a selector network to weight and combine the outputs from the last layer/encoder accordingly to form the input for the next layer/decoder. This design resembles the biological attention mechanism where our brains selectively concentrate on a distinct aspect of information while neglecting other perceivable information. With the help of attention, researchers have designed many approaches that properly accumulate information from multiple passages according to the query. Furthermore, multi-hop attention mechanism helps to focus on different aspects on different steps to simulate the reasoning chain for answering the question.

BiDAF [42] and FastQA [53] were introduced into the MHQA problem by Welbl et al.

[54]. Although originally designed for processing single document, these two methods were adapted successfully to work on a super-passage constructed by concatenating all documents randomly. MHPGM [6] proposed by Bauer et al. uses multi-attention mechanism to perform multi-hop reasoning and a pointer-generator network to synthesise the answer. Moreover, external commonsense information is extracted from ConceptNet to fill in the gaps of reasoning between hops using a gated attention mechanism. Weaver [37] performs co-encoding through several BiLSTM layers to relate the question to the documents, then use a memory network [47] to make the question attend to the context iteratively before making the decision.

More recently, Zhong et al. proposed CFC [63] which uses co-attention and self-attention to encode the documents with respect to the question and then score each candidate by comparing its occurrences throughout all documents with the question. DynSAN [66] approaches the problem from token-level by choosing the important tokens with a gated token selection mechanism first, then use the self-attention mechanism to model their dependencies with the help of convolutional layers.

The last work to be present is the Coref-GRU proposed by Dhingra et al. [13]. Although the method itself is not graph-based, it inspires the line of research on MHQA using GNN models. This method encodes the relation between mentions of entities with coreference, then a GRU augmented with jump links is used to aggregate the information across documents.

2.2.3 MHQA with Graph-Based Methods

This work follows the idea of graph-based MHQA methods and tries to improve them by augmenting the graph extracted from the documents with link prediction. The first attempt to use GNN for multi-hop reasoning was MHQA-GRN proposed by Song et al. [45]. This model consists of a BiLSTM (or Coref-BiLSTM which is the LSTM adaptation of Coref-GRU) to encode the local context of each entity mention, a few GNN layers to propagate information through the graph structure to perform the reasoning, and a fully-connected network to predict the answer. It outperformed all other models that do not utilise graph structure at the time it was published. However, this method does not take into account the query information when encoding the entity mentions and does not distinguish the types of edges which may be essential for reasoning over the relationships among the entities. Entity-GCN proposed by De Cao et al. [9] address the problems mentioned above. After encoding the query and entity mention with LSTM, query embedding is concatenated with each entity embedding and then pass through a feed-forward fully-connected network to combine the query information into the entity information. Also, all relationships are now treated differently with the help of R-GCN to propagate information differently through various types of edges.

HDEGraph proposed by Tu et al. [49] further improves the result by extending the graph which has all of its nodes being entity mentions into a heterogeneous graph: a graph consists of document nodes, candidate nodes, and entity nodes. Moreover, the number of types of edges is increased to seven from four. Finally, the co-attention and self-attention mechanisms for encoding the documents are adapted from CFC to get a better initialisation of node embeddings. BAG proposed by Cao et al. [8] tries to incorporate attention mechanism onto a graph by deferring the integration of query information into entity information until the R-GCN layers have accumulated the relational information over the entity graph to form relation-aware node embeddings. Then a bidirectional attention layer (node2query and query2node) is applied to produce the query-relation-aware node embeddings which are then passed into a fully-connected network to predict the answer. Note that the initial node features are a combination of ELMo and GloVe [35] embeddings.

Chapter 3

Methodology

This chapter consists of detailed descriptions of the design of the two models: one without link predictor and one that has. The two models are compared to investigate the effect of performing link prediction for MHQA tasks. Then the plan for implementation and testing of the models are articulated, with a discussion of the contribution of this dissertation follows.

3.1 Model Design

There are two models of consideration: one of them comprises only the entity graph extracted from the documents and a few R-GCN layers for propagating information, the other one consists of an additional link predictor to complete the entity graph before passing the graph into the R-GCN layers.

3.1.1 Model Without Link Predictor

The model without link predictor is very similar to that of Entity-GCN with a few modifications. It is built for providing a baseline for the one with facilitation of link prediction. This model consists of four parts: an encoder for summarising the local contexts of entity mentions, the entity graph extracted from the documents, an R-GCN network that propagates information on the entity graph, and a fully-connected network to predict the answer.

Encoder

As we discussed earlier, pre-trained word embedding techniques have boosted the performance of many downstream NLP tasks. The large corpus the embeddings are trained on and the tremendously many steps they are trained to help them to capture the statistical relationships between words and context. Since we have neither a large training corpus nor the necessary computational resources, a contextualised embedding, ELMo, but not a self-trained word encoder is used to obtain the initial word embeddings for entity mentions. Also, the lightweight encoding part achieved by using the pre-trained embedding makes the system to be able to run in real-time.

For an entity mention $[w_1, \ldots, w_n]$ of n words, we first get the pre-trained contextualised ELMo embeddings $[\mathbf{w_1}, \ldots, \mathbf{w_n} \in \mathbb{R}^{3072}]$. However, what we want is a single vector embedding for each entity mention. Unlike what was done in Entity-GCN where the first and last word vectors are concatenated to form $[\mathbf{w_1}||\mathbf{w_n}] \in \mathbb{R}^{6144}$ as the entity encoding, I take the element-wise mean of all word vectors to get $x' = (\mathbf{w_1} \oplus \cdots \oplus \mathbf{w_n}) \oslash n \in \mathbb{R}^{3072}$ as the initial entity encoding to reduce the number of parameters needed for the following encoding steps. Then x' is passed into a linear layer and a ReLU activation to get $x \in \mathbb{R}^{256}$.

For the query of n words, we also get the ELMo embeddings $[\mathbf{w_1}, \ldots, \mathbf{w_n} \in \mathbb{R}^{3072}]$ first. Then we do not perform the mean pooling, but the sequence is passed into two BiLSTM layers of which the hidden dimensions are 256 and 128 respectively. The last two outputs from the later BiLSTM layer, $\overrightarrow{q_n} \in \mathbb{R}^{128}$ and $\overleftarrow{q_n} \in \mathbb{R}^{128}$, are then concatenated to form the query encoding $q := [\overrightarrow{q_n}||\overleftarrow{q_n}] \in \mathbb{R}^{256}$.

This way we have the query encoding q and entities encodings $\{x_1, \ldots, x_N\}$ which summaries their local contexts. Furthermore, we would like the entity encodings to be query-aware so that the R-GCN layers propagates information with the query taken into concern. Hence we concatenate the query embedding with each entity encoding vector to get $\{[x_1||q], \ldots, [x_N||q] \in \mathbb{R}^{512}\}$ which are then passed into two layers of fully-connected network of which the hidden dimensions are 1024 and 512 respectively to get the query-aware entities encodings $\{\hat{x_1}, \ldots, \hat{x_N} \in \mathbb{R}^{512}\}$.

Entity Graph

The content of each training instance, i.e. the query and its supporting documents, is organised into a graph. The nodes are mentions of entities, the candidates and query entity, in all documents, note $\{v_1, \ldots, v_N\}$. All the queries are prepared in the form of $q =< s, r, ? >$ where s is the subject, r is the relation, and ? is the object which should be chosen among the candidates. Hence it is easy to separate the subject s in the query to be used as the query entity.

Apart from the exact matching of entities, a coreference resolution system [57] is used to extend the set of nodes in the graph. Specifically, for each coreference chain, if any of the entities are included in the chain, all other named entities in that chain will be added as new nodes. However, if any mention is resolved ambiguously into more than one coreference chains, it is discarded from the set of nodes to avoid the R-GCN network propagates vagueness.

All the nodes collected as described above in this subsection is then encoded as detailed in the last subsection to get the initial node embeddings $\{h_1^0, \ldots, h_N^0 \in \mathbb{R}^{512}\} = \{\hat{x_1}, \ldots, \hat{x_N} \in \mathbb{R}^{512}\}$. The edges defined below are then constructed to connect these mentions:

1. SAME-DOC edges: if they appear in the same document;
2. MATCH edges: if they are exact match;
3. COREF edges: if they occur within the same coreference chain;
4. COMPLEMENT edges: if they are not connected to any other nodes by the three kind of edges mentioned above, i.e. they are isolated.

Message Propagation Network

Since there are four types of edges, R-GCN is a natural choice for propagating information over this multi-relational graph. The reasoning chain is assumed to be captured by propagating local information along the edges, hence to simulate the reasoning chain properly we need a multi-hop propagation which leads to a multi-layer R-GCN network. As suggested by the original paper, $L = 3$ layers are used, since increasing the number of layers does not improve the performance anymore but cause the vanishing gradient problem.

Instead of using the vanilla R-GCN layer, a gated version of it is deployed so that the network learns to control how much information is propagated to the next step. If a reasoning chain has a length less than three, the gating mechanism should be able to stop the model from propagating information further away and blur the correct answer.

Recall that R-GCN layer updates node representations by:

$$u_i^{l+1} = \sigma\Big(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^l h_j^l + W_o^l h_i^l\Big).$$

Instead of setting $h_i^{l+1} = u_i^{l+1}$ directly, we train a linear layer f_a with sigmoid activation:

$$a_i^{l+1} = \sigma(f_a([u_i^{l+1}||h_i^l]))$$

to form the update gate and apply the update accordingly:

$$h_i^{l+1} = \phi(u_i^{l+1}) \odot a_i^{l+1} + h_i^l \odot (1 - a_i^{l+1}),$$

where ϕ is a *tanh* layer chosen as the non-linearity. Note that for reducing the number of parameters, all 3 R-GCN layers share the same parameters and hence the introduction of the gating mechanism is necessary for distinguishing each layer.

Output Network

After the graph is passed through the 3 R-GCN layers, we get the final node representations $\{h_1^3, \ldots, h_N^3 \in \mathbb{R}^{512}\}$. The Entity-GCN then computes the probability of choosing a candidate c as the answer by:

$$p(c|q) \propto exp\left(\underset{i \in \mathcal{M}_c}{max} f_o([q||h_i^3])\right),$$

where $\mathcal{M}_c$ is the set of indices such that node i is a mention of candidate c and f_o is the fully-connected output network. Instead, we remove the *max* operation and treat each node individually by computing:

$$p(i|q) = \sigma(f_o([q||h_i^3])),$$

where σ is the sigmoid activation, i.e. $p(i|q)$ is the probability whether the mention corresponding to node i is the correct answer. This way, we transformed the CE loss as a BCE loss and the multi-class classification problem is reduced as N binary classification problems. The output network f_o is a three layers fully-connected model with hidden dimensions 256, 128, 1 for each layer.

3.1.2 Model with Link Predictor

The only difference between this model with link predictor and the baseline model is that there is a link predictor, namely R-GCN link predictor, between graph construction and the message propagation network. Therefore, the description of an encoder, entity graph, message propagation network, and output network is omitted in this subsection.

Link Predictor

There is a choice between R-GCN and ConvE. As ConvE is deeper than R-GCN, it can produce better-learned features for complex graphs. More specifically, it outperforms R-GCN on graphs with node degrees greater than 10,000. Whereas for simpler graphs, R-GCN achieves better performance in link prediction than ConvE does. Hence an investigation into the entity graphs built for the QAngaroo WikiHop dataset is carried out to decide which model should be applied. The result of the investigation is shown in Table 3.1. As we can see, the entity graphs built are not very complex, and even the most complex graph has the maximum node degree being 317, which is much smaller than 10,000.

Table 3.1: Statistics of Entity Graphs Built

	# nodes	# edges	min degree	avg degree	max degree
Mean	67.41	2745.70	1.57	15.58	27.70
Max	611	137962	33	268.58	317

Therefore, R-GCN is deployed to perform graph completion. The predictor consists of two R-GCN layers for encoding the graph and a DistMULT decoder for scoring the triplets sampled. Two R-GCN layers both have the output dimension being 512, which is the same as that of the initial node embeddings. For simplicity, the basis decomposition layer is used, and the number of bases is the same as the number of relations, i.e. $B = |\mathcal{R}| = 4$. After the graph is passed through the encoder, we have the final node representations $\{h_1^2, \ldots, h_N^2 \in \mathbb{R}^{512}\}$ of which will then be fed into decoder for scoring the given triplets.

Like I mentioned before, the DistMULT decoder scores an edge bilinearly:

$$g(i, r, j) = \sigma\big((h_i^2)^\top W_r h_j^2\big),$$

where $W_r \in \mathbb{R}^{512\times512}$ is a diagonal matrix, and therefore W_r can be parameterised as $diag(\{W_r^{(1)}, \ldots, W_r^{(512)}\})$; σ is the sigmoid activation.

For training the predictor, negative sampling is used to construct the training instances. For each edge in the edge set $\{(s, r, t)\}$ of an entity graph built, we randomly corrupt its source or target to get the negative sample (s, r, t') or (s', r, t) which is not in $\{(s, r, t)\}$. For positive samples, i.e. edges/triplets in the original edge set, the labels are 1; for negative samples in the corrupted edge set $\{(s', r, t')\}$ the labels are 0. Hence BCE loss can be applied to train the predictor. During the testing phase, only edges with $g(s, r, t) > 0.5$ is accepted and added into the edge set of the entity graph.

3.2 Implementation

The original R-GCN and Entity-GCN papers run their experiments on CPU due to that a large amount of memory required to run vectorised computation cannot be fulfilled by GPU. In this project, we would like to build a pipeline which utilises the parallelism provided by GPU with the help of mixed precision training.

Firstly spaCy is used to perform named entity recognition which extracts the potential nodes of the entity graph. NeuralCoref built on top of spaCy is then deployed to resolve coreference which gives the coreference chains needed to decide which entities to add in the entity graph, and construct the COREF edges. Then AllenNLP with pre-trained ELMo embedding is applied to encode the local context of mentions of entities.

The edges constructed and the nodes being initialised by encoder are then packed into the graph Data object of PyTorch Geometric (PyG) [17], a geometric deep learning framework, which is built on top of PyTorch that supports tensor computation on GPU. All the models are then built with PyTorch and PyG. Apart from PyG, Deep Graph Library (DGL) has also been considered. DGL has a more elegant API for constructing and running GNN methods, but it does not support FP16, which abandons it to be incorporated into the mixed precision training pipeline of this work.

The instantiated models are then wrapped by Apex to allow automatic casting during the process of the forward pass to reduce the memory used, and loss scaling in the backward pass to preserve small gradient values.

3.3 Testing

Testing is performed on the development set of QAngaroo WikiHop dataset, as the testing set is not published. The accuracies achieved by these two models (with and without link predictor) on the development set are then compared to decide whether the use of link prediction helps to solve MHQA problems.

3.4 Contribution

This study investigates the effect of performing graph completion through link prediction on solving MHQA problems with graph-based methods. It also builds a pipeline that utilised mixed precision training to allow the model to be run on GPU to circumvent the limitation of memory size.

Chapter 4

Experimental Results

In this chapter, the set-up of the experiments like the hyperparameters used is discussed. Then the experimental results are presented with the assessment of the results.

4.1 Set-Up of Experiments

The number of R-GCN layers in the message propagation network and the link predictor are 3 and 2 respectively as discussed in the methodology chapter. As normalisation helps the training of neural networks, a model with normalisation but not link predictor is also evaluated to see if normalisation helps in MHQA task. Batch Normalisation is applied after each linear layer, Layer Normalisation [3] is deployed after every RNN layers, and Instance Normalisation [50] is used after each convolutional layer (like R-GCN layer). Hence there are four models to be compared in total:

A) Model with normalisation but not the link predictor which is trained for 3 epochs,

B) Model with neither normalisation nor the link predictor which is trained for 3 epochs,

C) Model with neither normalisation nor the link predictor which is trained for 4 epochs,

D) Model with the link predictor where both message propagation network and the link predictor are trained for 3 epochs.

For the training of message propagation network, query encoder, and entity encoder, the Adam optimiser with $lr = 10^{-5}$, $\alpha = 0.9$, and $\beta = 0.999$ is used. After these three networks are trained for 3 epochs, a copy of them is frozen. The query encoder and entity encoder are used to build the entity graphs which are needed to train the link predictor. An Adam optimiser with $lr = 10^{-2}$ is then deployed to train the link predictor. Meanwhile, the message propagation network is trained for one more epoch with the frozen query encoder and entity encoder.

As mentioned before, mixed precision training is used to reduce the memory needed in computation. Furthermore, we can enjoy the speed-up of mixed precision training with the help of Tensor Cores. Hence the models are trained on a GPU with Tensor Cores, namely NVIDIA Tesla T4.

4.2 Results

The plots of training losses and the table of accuracies achieved by each model are presented in this section. Figure 4.1 shows the training loss for the model without link predictor. We can see that the loss keeps going down throughout the 4 epochs of the training process. Hence we may expect that the accuracy on the testing set for the model without link predictor trained for 4 epochs being higher than that of the model trained for only 3 epochs.

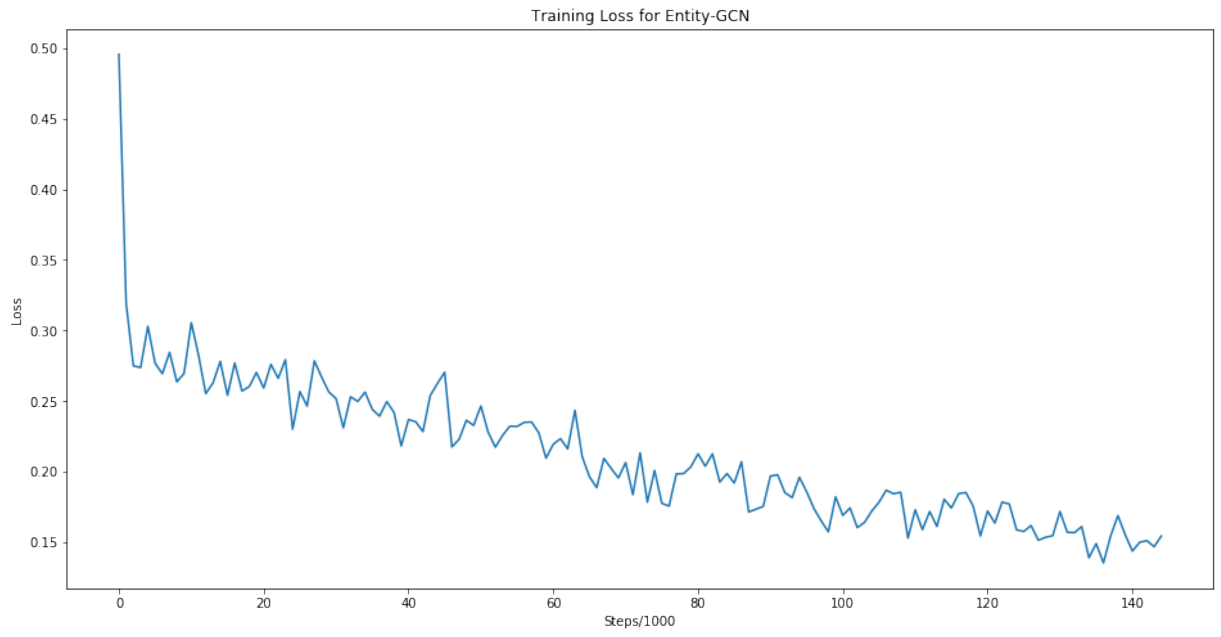


Figure 4.1: Training loss for model without link predictor.

Figure 4.2 displays the training loss of the model with link predictor. We can see that the model converges very quickly and the loss can no longer be pushed down after the first few hundred steps. It may indicate the occurrence of the vanishing gradient problem.

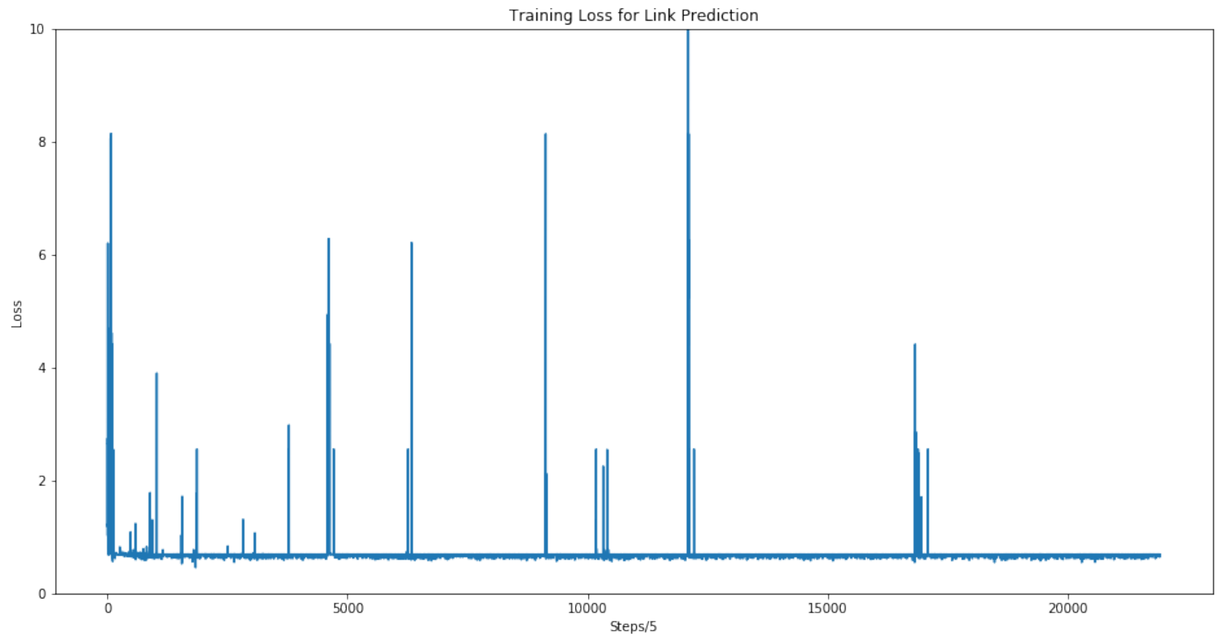


Figure 4.2: Training loss for model with link predictor.

Table 4.1 shows the accuracies achieved by each model mentioned in section 4.1. We can see that the normalisation layers applied have not improved the performance but degraded it severely. Also, the link predictor did not boost the model performance, yet dropped the generalisation capacity of the model.

Table 4.1: Accuracies for each model

	Accuracy
A	0.105
B	0.303
C	0.303
D	0.241

4.3 Assessment

It has shown by the result that adding this much normalisation layers could harm the model's performance instead of improving it. The first reason why it does not work is because of the batching of training data. Each entity graph extracted from documents varies greatly in terms of the graph structure like the number of nodes, number of edges, and the node degrees. Hence the statistics of one batch may not be similar to those of the other batches. Also, the batch size of 32 may be too small, which makes the statistics less representative. Furthermore, mixing different types of normalisations may not be appropriate. Although each normalization has shown to accelerate model training and improve model performance, there is rarely any study on combining them into the same model.

The model with neither normalisation nor the link predictor has not improved for training one more epoch (from Model B with 3 epochs to Model C with 4 epochs). This may be caused by over-fitting though the testing accuracy itself is not very high. Also, as the training progress, introducing a scheduler to perform learning rate decay may help the model in converging. Early stopping is another technique that can help us to determine when the model has over-fitted and need to break the training.

The link predictor has not enhanced the model but rather harms the performance on the test set. One of the reason might be that the graph completed is directed instead of undirected. The entity graphs built should be undirected, but PyG handles undirected graph by using two directed edges of opposite directions to represent an undirected edge. Hence, one predicted link is directed, and its counterpart may not be accepted by the link predictor. However, the message propagation network is trained on undirected graphs so it may be not able to propagate information properly on directed graphs. Also, after investigating the outputs of the link predictor, we found that most of them are near 0.5, i.e. the link predictor avoids to decide whether a given triplet should exist. Though the weights of DistMULT have magnitude far greater than 0, which means the model learns to balance h_i^2 and h_j^2 so that $(h_i^2)^\top W_r h_j^2 \to 0$. This may be prevented by adding a margin between positive and negative samples by using the margin-based ranking loss instead of BCE loss.

Chapter 5

Conclusion

5.1 Summary

This work investigates whether completing entity graphs with link prediction can boost the performance on MHQA problem. It has empirically shown that link prediction has not improved the performance but rather degraded it. Furthermore, a pipeline that utilises GPU as well as mixed precision training has been built to perform GNN methods to solve textual MHQA problem.

Chapter 2 introduces the background knowledge in NLP, GNN, and link prediction needed to understand the work. Specifically, contextualised word embedding, R-GCN, and DistMULT are the three key concepts required. Then other researchers' work on MHQA on KBs and short sentences are presented. Finally, the attention-based and graph-based methods to solve MHQA problem on long passages or documents are described to relate to this work. Particularly, Entity-GCN is the model from where I built my own method.

Chapter 3 describes the two lines of models: one without link predictor and the other one with a link predictor. Then the five core components including the encoder which captures the local textual context of mentions of entities, the entity graph constructor which extracts the graph for a query from its supporting documents, the link predictor which completes an entity graph, the message propagation network which accumulates the information on the graph along the global graph structure, and the output network which predicts the answer using the finalised node representations are articulated. Also, the implementation details and how the trained models are tested are explained.

Chapter 4 specifies the set-up of the experiments, including the hyperparameters used. Then the details of the four models to be compared are written down. After that, the experimental results are presented, which is followed by the assessment of the results.

In conclusion, this dissertation has shown that link prediction can not improve the performance of graph-based methods on textual MHQA tasks. Also, a system has been built to utilise the power of GPU with the help of mixed precision training.

5.2 Future Work

There are a few possibilities to improve the graph-based method used in this work. Furthermore, some challenging problems which may be tackle following this line of research will be discussed.

5.2.1 Encoder

In this work, a contextualised word embedding namely ELMo is used to form the input to the simple encoders: a BiLSTM layer and a linear layer. We may be able to enhance the encoding part by using more powerful embedding techniques like BERT [12] or XLNet [60]. These attention-based encoders with a tremendously large amount of parameters are trained on an amazingly large corpus for incredibly many steps. Hence they are a better language model then ELMo as they can represent a much larger space of embedded words and they have seen many more subtle statistical relations between words. Moreover, a more complex encoder like the one using co-attention and self-attention mechanisms in CFC can be trained to adapt the pre-trained embeddings to our dataset better.

5.2.2 Deeper GNN

As discussed before, the GNNs we have seen for now are very ‘shallow’ - around three or four layers due to vanishing gradient problem. As we know it now, the power of a neural network comes from building the network to be very deep. In order to tackle this problem, Li et al. [29] borrowed the idea of residual and dense connections from Convolutional Neural Networks (CNNs) and trained a GCN that is as deep as 56 layers. As a result that network has beaten other GCNs with only a few layers. Hence if we can extend R-GCN over 4 layers, the message propagation network may propagate information more appropriately.

5.2.3 Link Prediction

Though the graphs built do not show too many asymmetric relations, changing the decoder of RGCN from DistMULT to ComplEx may still improve the performance. Also, training the link predictor as well as the message propagation network at the same time and use the result of the output network as a signal for link predictor may additionally help the training of link predictor. By training these two modules together, we can also sooth the inconsistency between their goals.

5.2.4 HotpotQA

Proposed by Yang et al. [61], HotpotQA is a more challenging dataset then QAngaroo. The supporting documents here are the first paragraph of English Wikipedia's pages. The query is constructed such that at least two paragraphs from different pages are needed to answer the question. There are two settings in this dataset:

- Distractor: a list of supporting documents are given for each query. The two paragraphs that this query depends on are among those documents while there are an arbitrary number of 'distractor's documents to misguide the system;

- Full-Wiki: no supporting documents are given. The system needs to hop through the $5,000,000+$ wiki paragraphs to find out which passages are of interest.

Also, no candidate list is given in the HotpotQA dataset; hence, the system has to extract the correct named entity from the paragraphs itself.

It is challenging to apply the method designed in this work to the HotpotQA dataset, especially for the Full-Wiki setting. One main difficulty is to process the graph built for the Full-Wiki setting, which could contain thousands of nodes and edges. It is hard to put it into the memory, not to mention the system should run in real-time.

Appendix A

Codes

All the codes can be accessed from `https://github.com/yg-li/MHQA-with-LP`.

Bibliography

[1] Lada A Adamic and Eytan Adar. Friends and neighbors on the Web. *Social Networks*, 25(3):211–230, July 2003.

[2] A. Andrenucci and E. Sneiders. Automated question answering: review of the main approaches. In *Third International Conference on Information Technology and Applications (ICITA'05)*, volume 1, pages 514–519 vol.1, July 2005.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *arXiv:1607.06450 [cs, stat]*, July 2016. arXiv: 1607.06450.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]*, September 2014. arXiv: 1409.0473.

[5] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, October 1999.

[6] Lisa Bauer, Yicheng Wang, and Mohit Bansal. Commonsense for Generative Multi-Hop Question Answering Tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4220–4230, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[7] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc., 2013.

[8] Yu Cao, Meng Fang, and Dacheng Tao. BAG: Bi-directional Attention Entity Graph Convolutional Network for Multi-hop Reasoning Question Answering. *arXiv:1904.04969 [cs]*, April 2019. arXiv: 1904.04969.

[9] Nicola De Cao, Wilker Aziz, and Ivan Titov. Question Answering by Reasoning Across Documents with Graph Convolutional Networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2306–2317, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 3844–3852, USA, 2016. Curran Associates Inc. event-place: Barcelona, Spain.

[11] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d Knowledge Graph Embeddings. *arXiv:1707.01476 [cs]*, July 2017. arXiv: 1707.01476.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[13] Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. Neural Models for Reasoning over Multiple Mentions Using Coreference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 42–48, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[14] Bhuwan Dhingra, Kathryn Mazaitis, and William W. Cohen. Quasar: Datasets for Question Answering by Search and Reading. *ArXiv*, abs/1707.03904, 2017. arXiv: 1707.03904.

[15] Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Guney, Volkan Cirik, and Kyunghyun Cho. SearchQA: A New Q&A Dataset Augmented with Context from a Search Engine. *arXiv:1704.05179 [cs]*, April 2017. arXiv: 1704.05179.

[16] Jeffrey L. Elman. Finding Structure in Time. *Cognitive Science*, 14(2):179–211, 1990.

[17] Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric. *arXiv:1903.02428 [cs, stat]*, March 2019. arXiv: 1903.02428.

[18] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, May 2013.

[19] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1024–1034. Curran Associates, Inc., 2017.

[20] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching Machines to Read and Comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc., 2015.

[21] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations. *arXiv:1511.02301 [cs]*, November 2015. arXiv: 1511.02301.

[22] Sarthak Jain. Question Answering over Knowledge Base using Factual Memory Networks. In *Proceedings of the NAACL Student Research Workshop*, pages 109–115, San Diego, California, June 2016. Association for Computational Linguistics.

[23] Glen Jeh and Jennifer Widom. SimRank: A Measure of Structural-context Similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 538–543, New York, NY, USA, 2002. ACM. event-place: Edmonton, Alberta, Canada.

[24] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, July 2017.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, December 2014. arXiv: 1412.6980.

[26] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907 [cs, stat]*, September 2016. arXiv: 1609.02907.

[27] Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The NarrativeQA Reading Comprehension Challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018.

[28] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding Comprehension Dataset From Examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[29] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. DeepGCNs: Can GCNs Go as Deep as CNNs? *arXiv:1904.03751 [cs]*, April 2019. arXiv: 1904.03751.

[30] Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. Denoising Distantly Supervised Open-Domain Question Answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1736–1745, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[31] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning Entity and Relation Embeddings for Knowledge Graph Completion. *AAAI 15*, page 7, 2015.

[32] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[33] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[34] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE*, 104(1):11–33, January 2016.

[35] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[36] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 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)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[37] Martin Raison, Pierre-Emmanuel Mazaré, Rajarshi Das, and Antoine Bordes. Weaver: Deep Co-Encoding of Questions and Documents for Machine Reading. *arXiv:1804.10490 [cs]*, April 2018. arXiv: 1804.10490.

[38] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, November 2016.

[39] Siva Reddy, Danqi Chen, and Christopher D. Manning. CoQA: A Conversational Question Answering Challenge. *Transactions of the Association for Computational Linguistics*, 7(0):249–266, May 2019.

[40] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80, January 2009.

[41] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam, editors, *The Semantic Web*, Lecture Notes in Computer Science, pages 593–607. Springer International Publishing, 2018.

[42] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional Attention Flow for Machine Comprehension. *ArXiv*, abs/1611.01603, 2016.

[43] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. ReasoNet: Learning to Stop Reading in Machine Comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1047–1055, New York, NY, USA, 2017. ACM. event-place: Halifax, NS, Canada.

[44] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 926–934. Curran Associates, Inc., 2013.

[45] Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. Exploring Graph-structured Passage Representation for Multi-hop Reading Comprehension with Graph Neural Networks. *arXiv:1809.02040 [cs]*, September 2018. arXiv: 1809.02040.

[46] Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. Fast and Accurate Entity Recognition with Iterated Dilated Convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2670–2680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[47] Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. End-To-End Memory Networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc., 2015.

[48] Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. Improving Machine Reading Comprehension with General Reading Strategies. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2633–2643, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[49] Ming Tu, Guangtao Wang, Jing Huang, Yun Tang, Xiaodong He, and Bowen Zhou. Multi-hop Reading Comprehension across Multiple Documents by Reasoning over Heterogeneous Graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2704–2713, Florence, Italy, July 2019. Association for Computational Linguistics.

[50] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance Normalization: The Missing Ingredient for Fast Stylization. *arXiv:1607.08022 [cs]*, July 2016. arXiv: 1607.08022.

[51] Zhen Wang, Jiachen Liu, Xinyan Xiao, Yajuan Lyu, and Tian Wu. Joint Training of Candidate Extraction and Answer Selection for Reading Comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1724, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[52] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge Graph Embedding by Translating on Hyperplanes. *AAAI 14*, page 8, 2014.

[53] Dirk Weissenborn, Georg Wiese, and Laura Seiffe. Making Neural QA as Simple as Possible but not Simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 271–280, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[54] Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing Datasets for Multi-hop Reading Comprehension Across Documents. *Transactions of the Association for Computational Linguistics*, 6:287–302, 2018.

[55] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. *arXiv:1502.05698 [cs, stat]*, February 2015. arXiv: 1502.05698.

[56] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The Marginal Value of Adaptive Gradient Methods in Machine Learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4148–4158. Curran Associates, Inc., 2017.

[57] Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. Learning Global Features for Coreference Resolution. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 994–1004, San Diego, California, June 2016. Association for Computational Linguistics.

[58] Chien-Sheng Wu, Richard Socher, and Caiming Xiong. Global-to-local Memory Pointer Networks for Task-Oriented Dialogue. *arXiv:1901.04713 [cs]*, January 2019. arXiv: 1901.04713.

[59] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. *arXiv:1412.6575 [cs]*, December 2014. arXiv: 1412.6575.

[60] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv:1906.08237 [cs]*, June 2019. arXiv: 1906.08237.

[61] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[62] Pengcheng Yin, Zhengdong Lu, Hang Li, and Kao Ben. Neural Enquirer: Learning to Query Tables in Natural Language. In *Proceedings of the Workshop on Human-Computer Question Answering*, pages 29–35, San Diego, California, June 2016. Association for Computational Linguistics.

[63] Victor Zhong, Caiming Xiong, Nitish Shirish Keskar, and Richard Socher. Coarse-grain Fine-grain Coattention Network for Multi-evidence Question Answering. *arXiv:1901.00603 [cs]*, January 2019. arXiv: 1901.00603.

[64] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph Neural Networks: A Review of Methods and Applications. *arXiv:1812.08434 [cs, stat]*, December 2018. arXiv: 1812.08434.

[65] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *Eur. Phys. J. B*, 71(4):623–630, October 2009.

[66] Yimeng Zhuang and Huadong Wang. Token-level Dynamic Self-Attention Network for Multi-Passage Reading Comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2252–2262, Florence, Italy, July 2019. Association for Computational Linguistics.