A Modular and Efficient Approach for Question Answering Over Knowledge Bases

March 2022

Happy Buzaaba

A Modular and Efficient Approach for Question Answering Over Knowledge Bases

Graduate School of Systems and Information Engineering

University of Tsukuba

March 2022

Happy Buzaaba

Abstract

Question answering over knowledge base (KBQA) is an important and challenging task which aims at correctly answering natural language questions posed by humans. It has a wide range of application in natural language processing (NLP) and information retrieval (IR) such as search, dialogue systems, information extraction, and summarization. Several existing knowledge-based question answering systems exploit complex end-to-end neural network architectures to solve the QA task. While these methods achieve good results, this performance comes at a high memory and computational cost because data or tokens are represented as vectors or embeddings of high dimension to train the neural network. These high dimension representations require more computational resources and take long to execute when training the neural network. Further more, such end-to-end approaches, makes it even more difficult to conduct a detailed performance analysis.

In this dissertation, we focus on how to efficiently perform question answering over knowledge bases. This study consists of two parts; In the first part, the question answering task is decomposed into three different components of entity detection, entity linking, and relation prediction, and we solve each of the components separate to come up with the correct answer candidate to the question. The second part applies dimension reduction technique to generate low dimensional vectors that are used to train a similarity matching function between the question and the candidate answers from which the closest candidate answer is selected to be the correct answer to the question. Further more, knowledge based question answering faces an ambiguity challenge when identifying the matching entities and relations in the knowledge base. We take a step towards addressing these challenges.

In order to achieve efficient question answering over knowledge bases, We propose a simple yet efficient approach to address the knowledge base question answering task. We also propose and analyze an auto-encoder framework that can learn low dimension representations of the original input embeddings, and then reconstructs them back with a lesser dimension. Our proposed framework retains as much information as possible and minimize the reconstruction error between the original and the reconstructed data when training the question answering model. As a result, in the experiment and the analysis, this study indicates that we can achieve reasonable performance on the question answering task all the while preserving computational time and memory requirements.

Acknowledgements

This work was accomplished with the help and support from many people. I would like to extend my appreciation to everyone who has helped me or contributed in any away towards the accomplishment of this work.

First of all, my sincere appreciation goes to my supervisor Professor Toshiyuki Amagasa who gave me a chance to pursue the Ph.D degree. Without his unwavering support, advise and encouragement, this thesis would not have been possible. Also, I would like to thank great faculty members, Professor Hiroyuki Kitagawa, Associate Professor Hiroaki Shiokawa, Assistant Professor Yasuhiro Hayase, and Assistant Professor Kazumasa Horie. For their help and encouragement.

I am very grateful to my doctoral committee, Professor Toshiyuki Amagasa, Professor Hiroyuki Kitagawa, Professor Kazuhiro Fukui, Associate Professor Inui Takashi, and Professor Tetsuji Satoh for their valuable suggestions and constructive recommendations. Their comments greatly helped me to improve the quality of this dissertation.

Also, my thesis was supported by my colleagues from the RIKEN Advanced Intelligence project (AIP). I would like to extend my sincere appreciation to the Approximate Bayesian Inference Team at RIKEN AIP, Dr. Emtiyaz Khan, Dr. Pierre Alquero, Dr. Gianma Marcorni, Dr. Thomas Mollenf, Mr. Darmesh Tailor, Mr. Peter Nickel. Special Thanks goes to Gianma Marcoroni for always supporting me.

I would also like to thank Knowledge and Data Engineering Lab members. My seniors and friends, Assistant Professor Savong Bou, Assistant Professor Ito Hiroyashi, Dr. Leo Ota, Katia Bouhrouman, Miyamoto Tatsuro, Kento Miura, Okura Maiki, Vijdan Khariq, Prince Gashongore, Mehari Yohanese, Marie Grace Uwamariya, and John Bosco Mugeni for their support. Ms. Yumiko Hisamatsu and Ms. Shihoko Sekiya also helped me to accomplish this work.

In a special way, I would like dedicate this thesis to my mother the late Mukankaka Bonifirida who passed away three weeks before my final defense, and my father the late Kasumba Samuel who also passed away in the final year of my Ph.D. study. I was not able to attend their funerals because of both the covid19 travel restrictions and the tight schedule of my Ph.D. may their souls rest in eternal peace.

Finally, I would like to thank all my family members; my brothers, sisters, cousins, aunties, and uncles for being there for me all the time. Without your prayers and encouragement, I would not be here.

Happy Buzaaba

December 2021

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Question answering over knowledge base has been conducted using large scale knowledge bases (KB), such as Freebase [12], DBpedia [60], Wikidata [93], and Yago [44] to mention but a few. Such large scale knowledge bases, consist of real world entities as nodes and the relations between them as edges. Each directed edge along with its head and tail entity, constitute a triple that is to say (`head entity`, `relation`, `tail entity`) also known as a knowledge base fact. For example, when a natural language question such as `where was Barack Obama born?` is posed by a user, the aim of question answering over the knowledge base is to identify a triple or fact (`Barack Obama`, `people/person/place_of_birth`, `Honololu`) from the the knowledge base such that the tail entity (`Honololu`) is the answer to the question.

Knowledge bases have in recent years become pivotal in question answering because of their ability to provide precise answers to users questions. Users commonly use structured query languages like SPARQL for querying resource description framework (RDF) data in such knowledge bases. SPARQL is a powerful query language which requires expert knowledge that is hard to learn for non-programmers who would want to access the information in the KB. So, the most convenient approach that has gained much attention is knowledge based question answering that allows end users to pose natural language questions over a knowledge base without knowledge of the underlying schema, and in return receive entities as the answers to the question.

In the past, a number of question answering systems, such as AquaLog [62],

NLP-Reduce [54], PowerAqua [63], and FREyA [25], have been proposed. Many of them map a natural language question to a triple-based representation. For example, a simple question like "who wrote the Neverending story?", PowerArqua [63] would map this question to the triple ([person,organisation], wrote, Neverending story) and then, similarity measures are applied to retrieve the matching sub-graphs from the RDF repository. This approach, however, has a number of drawbacks such as failure to capture the original semantic structure of the question using triples.

In addition, several studies in deep learning, have seen a surge of end-to-end complex neural network approaches that have performed well on a variety of natural language processing tasks like opinion extraction [47], sentence classification [55], entity linking [102], and question answering [41, 65]. While these methods achieve good results, this performance comes at a high memory and computational cost because data or tokens are represented as vectors or embeddings of high dimension to train the neural network. These high dimension representations require more computational resources and take long to execute when training the neural network. Further more, such end-to-end approaches, makes it even more difficult to conduct a detailed performance analysis.

In this dissertation, we propose a modular and efficient knowledge based question answering approach. The proposed approach consists of two parts; In the first part, the question answering task is decomposed into three different components of entity detection, entity linking, and relation prediction, and we solve each of the components separate to come up with the correct answer candidate to the question [18]. The second part applies dimension reduction technique to generate low dimensional vectors that are used to train a similarity matching function between the question and the candidate answers from which the closest candidate answer is selected to be the correct answer to the question [17].

Although the question answering task only involves retrieving facts in the KB, it is quite challenging in real life because (1). a synonymy or variant of the name may be used in the knowledge base (KB), (2). knowledge bases contain millions of entities represented by entity machine identifiers (MID's) whereby different MID's posses the same name and when a natural language query is posed, it retrieves many candidate entities that are hard to distinguish. For example, it becomes difficult to answer a question such as 'what country was the film the debt from?' because there are 4 entity MID's (04j0t75, 0bj3wz4j, 0bjwlk1l, 0dy60p) with the name 'the debt' in the Freebase KB. A sim-

ilar problem is faced during relation prediction, where multiple relations are identified and it is not clear which one is being referenced. For example, there are different relation types referenced for the above question i.e., (`film/film/country`, `film/film/written_by`, `music/album/release_type`). These ambiguity problems, exacerbate as the scale of the knowledge base grows. For these reasons, entity linking becomes a bottleneck in knowledge base question answering and can be conjectured that a method to identify such ambiguities in the data would be crucial for the enhancement of KBQA.

1.1 Contributions

In this study, we propose a simple yet efficient approach to address the knowledge base question answering task. It consists of two parts; part one is referred to as pipeline-based question answering, and this approach decomposes the question answering task into three different components that is to say **entity detection**, **entity linking**, **relation prediction** and each of these components are separately solved to come up with the correct answer candidate for the given question. Part two which we refer to as the dimension reduction for question answering applies dimension reduction technique to generate low dimensional vectors for both the natural language question and the candidate answers to train the similarity matching model between the question and candidate answer representation. This proposed approach is faster, efficient, and performs reasonably well compared to previous complex approaches. In this section, we summarize the main ideas and contributions of this study.

1.1.1 Pipeline-Based Approach for Question Answering

The knowledge based question answering task aims at identifying a triple from the underlying knowledge base to answer a given natural language question. We employ what we call the pipeline-based approach to solve the task where by the task is decomposed into sub-tasks of entity detection, entity linking, and relation prediction.

- **Entity detection:** Given a natural language question, we start by identifying the entity mentioned in the question. The entity mention identified from the question text is used as the query in the entity linking phase to search for all

candidate entity nodes in the knowledge base associated to the mention. We formulate this sub-task as a sequence labeling problem that assigns a label to each token.

- **Entity linking:** The entity detection step outputs a sequence of tokens that represent an entity mention which should be linked or queried to the knowledge base. This is formulated as a search problem with the aim of retrieving top matching candidates using string matching.
- **Relation prediction:** The goal of this sub-task is to identify the relation being queried in the given natural language question. This is done by classifying the natural language question as one of the knowledge base relation types. It outputs the relation type associated to the given question. This relation type is used for pruning the candidate list generated by the entity linking step and all the entity nodes with a relation type different from the one associated to the question are removed from the list. The candidate triple with the highest score in remaining list is chosen to have the object entity which is the answer to the question.

The experiment performed using the SimpleQuestions benchmark dataset shows that a combination of basic LSTMs, GRUs, and non-neural network techniques achieve reasonable performance while providing an opportunity to understand the question answering problem structure.

1.1.2 Dimension Reduction for Question Answering

Dimensional reduction techniques have proved to be computationally efficient while at the same time achieving good results on different NLP tasks [35]. However, they have not been extensively explored for the question answering task.

In this study, we extend the dimension reduction approach to the question answering task. To this end, we presents a novel approach to reconstruct embedding dimensions to a low dimension and use the reconstructed low dimension embeddings to design a similarity matching function to measure the semantic similarity between the input natural language question and the candidate answers where the closest candidate becomes the answer to the question.

To be specific, we apply the Long Short Term Memory auto-encoder (LSTM-AE) [61] to generate low dimensional vectors. The LSTM-AE, captures both the

semantic information in word embeddings and the syntactic information in word order. The LSTM-AE focuses on the most relevant features of the input embeddings and uses these most relevant features to train a question answering model.

The Experiments on a real world insuaranceQA benchmark, show that the proposed approach can obtain performance comparable to standard baselines while remaining cost efficient on both time and memory.

1.2 Overview of the Thesis

In the previous sections we have presented the background, motivations, and the main contributions of this thesis. In this section, we summarize the dissertation structure:

Chapter 2: Preliminary. This chapter makes a thorough review on the trends in KBQA. We discuss the background, advances, and the applications of question answering research.

Chapter 3: Related work. In this chapter, we review existing studies on question answering, to be specific, we review the pros and cons of the existing methods.

Chapter 4: Pipeline-Based approach. In Chapter 4, we formally define the question answering problem and propose our work of decomposing the knowledge base question answering task into sub-tasks of entity detection, entity linking, and relation prediction. We also present a detailed analysis of the experimental results that confirm the effectiveness of our approach.

Chapter 5: Dimension Reduction approach. Chapter 5, covers our work on the application of dimension reduction techniques to solve the question answering task. In this work, we apply an auto-encoder to generate low dimensional embeddings from the original input dimensions of the question and the candidate answers, and we design a similarity matching function to measure the semantic similarity between the input question and the candidate answers.

Chapter 6: Conclusion. Finally in Chapter 6, we conclude the thesis and talk about future directions.

Chapter 2

Preliminaries

As mentioned earlier, this thesis focuses on improving the performance of question answering over knowledge base also known as KBQA. In this chapter, we thoroughly review the trends in KBQA, we discuss the background, advances and applications of question answering research.

2.1 Question Answering Research

Question answering is one of the oldest tasks in natural language processing which aims at addressing the human need to information access. It can be traced back in the 1960's [85] when research on practical question answering systems and language processing started to take off. At the time, English questions on how to conduct natural language research such as; how to characterize a sentence?, how to handle ambiguity and partial answers in the language were being asked [94]. Thanks to this, high quality question answering systems that answered questions about baseball [38] and human dialogue systems [11, 96] were designed as early as 1960. These and other question answering systems not mentioned here, relied on two major QA paradigms to answer natural language questions posed by humans that is to say information retrieval-based QA and knowledge base-based QA.

While there are different forms of questions that can be asked by humans, most of the existing question answering systems focus on factoid questions which is also the primary focus of our thesis. Factoid questions can be answered by simple facts

that can be expressed inform of short texts. Examples of such questions include:

- `Where was` **`Barack Obama`** `born?`
- `Who was the first prime minister of` **`Ethiopia`**`?`

Such questions are referred to as factoid questions because they consist of a single entity mention and as earlier mentioned, the major paradigms for addressing the factoid question answering are information retrieval-based question answering and knowledge base-based question answering. In the following subsections, we discuss in details each of the question answering paradigms.

2.1.1 Information Retrieval-based Question Answering (IR-based)

Information retrieval-based question answering which is sometimes called open domain question answering, relies on the vast amount of web text or collections of scientific papers like DBLP to retrieve relevant passages used to answer users question [53].

Usually, the user will pose a natural language query to the information retrieval system, the system will then return a set of documents retrieved from a vast amount of documents that is to say, news articles, paragraphs, web texts, and scientific papers as illustrated in Figure 2.1 a similarity matching function such as a cosine similarity [82] is applied to rank potential documents. To rank documents in information retrieval **(IR)**, the term weight for each document word referred to as term frequency inverse document frequency **(tf-idf)** is computed.

Term Frequency Inverse Document Frequency (tf-idf): It is the product of two terms, the term frequency **(tf)** and the inverse document frequency **idf**. Term frequency **tf** is the measure of the word frequency in the document. The words that occur more often are likely to inform the document contents. Quite often, the term frequency is obtained using log of word frequency Equation 2.1. This is due to the fact that the more frequency the word is in the document does not necessarily translate into direct relevance to the document.

$$\mathbf{tf} = \mathbf{log_{10}}(\mathbf{count}(\mathbf{t}, \mathbf{d}) + \mathbf{1}) \tag{2.1}$$

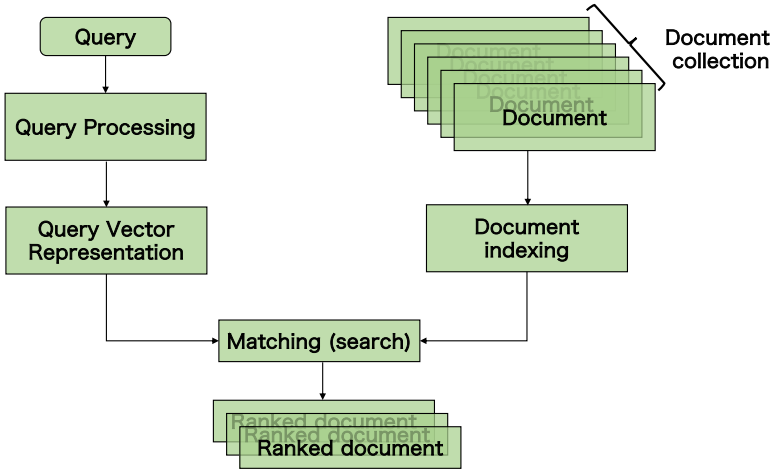


Figure 2.1: Illustration of the basic information retrieval or open domain question answering architecture. The query and documents are represented as vectors in the vector space, and a similarity matching function is applied to rank potential documents.

where **t** and **d** represent the $term$ and $document$ respectively. One is added in Equation 2.1 to void taking log on 0.

On the other hand, document frequency **df** is the number of documents containing the term. The terms that occur in only a few documents are helpful in discriminating those documents from the rest while the terms that occur across the entire collection of documents are less helpful. Inverse document frequency **(idf)** [51] is obtained from;

$$\mathbf{idf} = \log_{10} \frac{\mathbf{N}}{\mathbf{df}} \tag{2.2}$$

where **N** is the total number of documents, and **df** the number of documents in which **t** occurs.

Therefore, term frequency inverse document frequency **(tf-idf)** is obtained as a product of term frequency **tf** and the inverse document frequency **idf**.

$$\textbf{tf-idf} = \mathrm{tf} \cdot \mathrm{idf} \tag{2.3}$$

The similarity matching function is computed using the cosine similarity between the document $\overrightarrow{\mathbf{d}}$ and query $\overrightarrow{\mathbf{q}}$ vector representations as;

$$\mathbf{cos}(\overrightarrow{\mathbf{q}}, \overrightarrow{\mathbf{d}}) = \frac{\overrightarrow{\mathbf{q}} \cdot \overrightarrow{\mathbf{d}}}{||\overrightarrow{\mathbf{q}}||||\overrightarrow{\mathbf{d}}||} \tag{2.4}$$

It is also possible to compute the cosine similarity as the dot product of unit vectors as;

$$\mathbf{cos}(\overrightarrow{\mathbf{q}}, \overrightarrow{\mathbf{d}}) = \frac{\overrightarrow{\mathbf{q}}}{|\overrightarrow{\mathbf{q}}|} \cdot \frac{\overrightarrow{\mathbf{d}}}{|\overrightarrow{\mathbf{d}}|} \tag{2.5}$$

on replacing the query and document representation with term frequency inverse document frequency, the Equation 2.5 can be written to measure the similarity between the query and the document as a sum of products of:

$$\mathbf{score}(\mathbf{q}, \mathbf{d}) = \sum_{\mathbf{t} \in \mathbf{q}} \frac{\textbf{tf-idf}}{|\mathbf{d}|} \tag{2.6}$$

Evaluation: The above similarity score, outputs a set of ranked documents and we evaluate the performance of ranked documents as either relevant to our purpose or not relevant using precision, recall, and F_1-measure as the metrics.

The precision measures the fraction of the returned documents that are relevant as;

$$Precision = \frac{|Relevant \cap Retrieved|}{|Retrieved|} \tag{2.7}$$

While the recall measures the fraction of all relevant documents that are returned as;

$$Recall = \frac{|Relevant \cap Retrieved|}{|Relevant|} \tag{2.8}$$

It is important to note that for the question answering task, the corresponding terms i.e., relevant and retrieved would mean correctly answered and attempted to answer respectively.

The F-measure is the weighted average between the precision and recall and it is computed as follows:

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{2.9}$$

We also evaluate the ranked retrieval using mean average precision **(MAP)**. This metric, quantifies how good the model performs at a given question. It is given as the average over ranks;

$$MAP = \frac{1}{Q} \sum_{q=1}^{Q} aveP(q) \tag{2.10}$$

where Q is the number of questions and $aveP$ is the average precision for a given question q. Given a question, its corresponding $aveP$ is calculated and the mean of all these $aveP$ scores gives us the mean average precision.

Datasets for Information Retrieval-based QA Most of the existing datasets used in the **IR**-based question answering contain tuples of `(passage, question, answer)` an example can be seen in Figure 2.2.

Table 2.1: Existing Information Retrieval Based Question Answering Datasets.

Dataset Name Reference	Size Total Questions	Source Questions Source	Formulation Dataset formulation
SQuAD V1.0 [77]	questions 100,000	Crowdsourced Wikipedia	Passage text spans
SQuAD V2.0 [76]	questions 150,000	Crowdsourced Wikipedia	Passage text spans
HotPotQA [99]	questions 113,000	Crowdsourced Wikipedia	Passage sentence
TriviaQA [52]	questions 650,000	Crowdsourced Wikipedia & web	passage sentence
Natural Questions [58]	questions 323,045	Crowdsourced Wikipedia	Passage paragraph
TyDiQA [23]	questions 204,000	Multilingual Wikipedia	Passage paragraph
MS MARCO [71]	questions 100,000	User logs Bing	Generated Human
NewsQA [88]	questions 100,000	Crowdsourced CNN	passage text-spans
WikiQA [98]	questions 3,047	Crowdsourced Bing	passage paragraph
TRECQA [92]	questions 1,362	Crowdsourced user	passage text-spans

Several datasets for Information Retrieval-based QA have been introduced and Some of the existing **IR**-based QA datasets are shown in table 2.1.

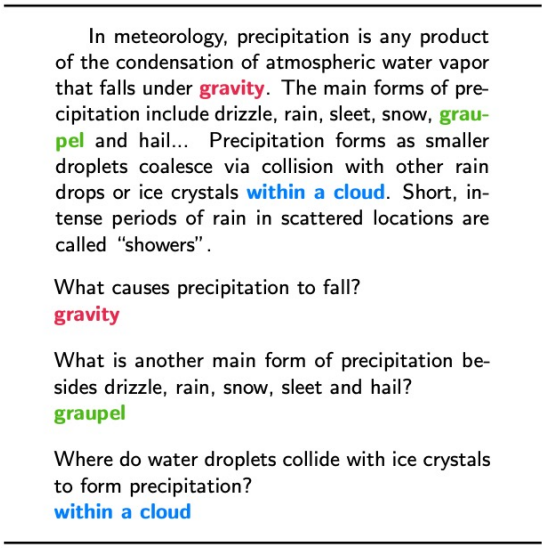


Figure 2.2: A wikipedia sample passage in the SQuAD [76] dataset. Each of the answers is a segment of text from the passage.

2.1.2 Knowledge Base-based Question Answering (KBQA-based)

The second paradigm for factoid question answering is the knowledge-based. Different from the **IR-based** in Section 2.1.1, the **KBQA-based** relies on a structured knowledge base to answer the posed natural language question. The system maps the natural language question to a structured query which is then used to query the knowledge base.

Knowledge bases consists of a huge amount of valuable information encoded in form of resource description framework (RDF) triples. The RDF triple, consists of a tuple that is to say `entity, relation, entity` which often represents real world entities in form of `subject, predicate, object`. Using existing knowledge bases such as DBpedia [60] or Wikidata [93], the Knowledge-based QA task is usually solved by either modeling the knowledge base as a graph consisting of real world entities as nodes and connected to other nodes by relations as edges between nodes.

Given a natural language question, the task of knowledge-based question answering is to identify a triple or fact from the knowledge base that is the intended

answer to the question. A triple can also be interpreted as a directed edge. Figure 2.3 shows the the process of question answering using a graph as the knowledge base. As can be seen from the figure, for a natural language to be answered, the entity mention in the question is linked to the knowledge base through entity linking.

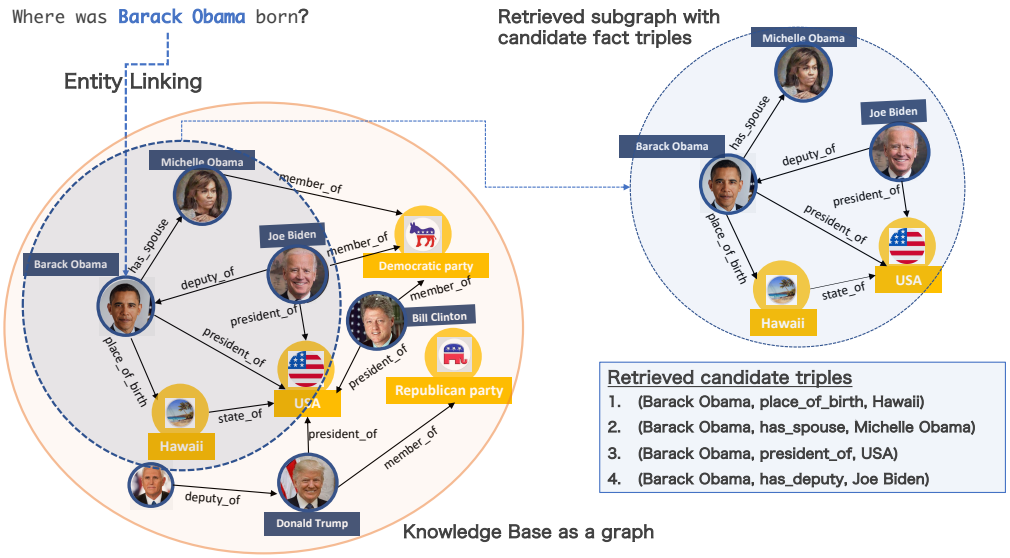


Figure 2.3: Illustration of the question answering over a knowledge base. From the figure, given a natural language question the mention in the question is first linked to the knowledge base to retrieve a sub-graph with ranked candidate fact triples.

Entity linking Entity linking is at the core of knowledge-based question answering. It plays a central role of linking the entity mention in the natural language question to the corresponding entity nodes in the knowledge base [15, 48]. Entity linking in the knowledge-based question answering task is performed through mention detection and mention disambiguation [40, 67, 70]. The entity linking task is a well studied problem in literature [16, 84] and **TAGME** linker [31] is one of the common entity linking algorithms used in literature which first detects the mention in the question and then links the mention to the corresponding entity nodes in the knowledge base to output a set of candidate fact triples from the underlying knowledge base as shown in Figure 2.3. The knowledge-based question answering task has increasingly attracted many researchers attention and some of the existing

datasets for this problem are shown in Table 2.2.

Table 2.2: Existing Knowledge Base-Based Question Answering Datasets.

Dataset Name Reference		Size Total Questions		Knowledge & train set Knowledge-base : train		Dev & test sets Dev: test	
SimpleQuestions	[13]	questions	108,442	Freebase	: 75,910	10,845	: 21,687
FreebaseQA	[50]	questions	28,348	Freebase	: 20,358	3,994	: 3,996
WEBQUESTIONS	[6]	questions	5,810	Freebase	: 3,778	-	: 2,032
WEBQUESTIONSSP	[101]	questions	4,737	Freebase	: 3,098	-	: 1,639
COMPLEXWEBQUESTIONS	[86]	questions	34,689	Freebase	: -	-	-

Modern systems also combine the knowledge-based and information retrieval based knowledge sources also known as hybrid-base question answering to achieve better performance on the question answering task [32].

2.1.3 Application of Question Answering

Question answering systems have seen a wide range of applications including human computer interfaces, where they seem to be more natural and convenient compared to keyword search for non-technical users [34, 83, 91].

Traditionally, information access calls for manual reading of large amounts of documents to identify the relevant piece of information. However, question answering systems have proven potential to directly obtain the correct answer to the posed questions which accelerates the information search process in practice [80].

One of the prominent question answering systems is the IBM Watson that worn the Jeopardy challenge [33]. IBM watson has since been extended to other components such as healthcare decision support and business intelligence. Besides, question answering research have led to designing of domain specific systems in the medical field [21], and question answering for education [20].

Furthermore, question answering systems are increasingly being applied in electronic edge devices such as wearables, google home, Apple siri to mention but a few. Although question answering systems are increasingly being applied to edge devices, they have not been fully adopted in actual real world application. One of the main reasons for this limitation is the computational complexity associated with existing state-of-art question answering systems.

Chapter 3

Related Work

Research on Question answering has continued to gain much attention in recent years, and several studies that solve this task are increasingly being proposed. This section reviews some of the existing works in the literature that solve this task.

Today, the existing approaches for solving the QA task maybe labeled as, free text-based or open domain question answering and structured knowledge-base based question answering. In free text-based question answering, answer candidates are first retrieved from text documents, and the answer to the question is obtained by identifying the most similar answer out of the candidate answers. On the other hand, the structured knowledge-based question answering approach relies on large scale knowledge bases like DBpedia [60] to find the answer to the user's question. In the related works, we will focus on structured knowledge based question answering.

The existing methods for solving the above question answering approaches can be categorized into two main categories of semantic parsing, and information extraction methods. The semantic parsing methods aim at mapping the natural language question into a logical form for it to be queried on the knowledge base. The existing methods that directly parse the natural language question into a structured query using semantic parsing include syntactic parsing [10, 78, 97], semantic role labeling [9], and semantic parsing [100]. In addition, other semantic parsing methods such as [6, 7] use hand-crafted rules to manually translate the natural language question to a structured query.

However, the application of manual annotation is often expensive when training

the parser and for this reason, a quite different but semantic approach was proposed in [8]. In this approach, Berant et al start by representing the document entities and relations in form of a knowledge base, and then they represent the question as a structured query for easy matching with the knowledge base contents. It is important to note that these semantic based methods rely on linguistic heuristics hence domain specific, and as previously mentioned, they largely depend on manually annotated hand-crafted rules which limits them from being scaled and transferred.

In contrast, the information extraction methods focuses on retrieving a set of candidate answers from text documents or candidate triples from the underlying structured knowledge base and measure semantic similarity between the question and candidate answers to obtain the correct answer. Information extraction methods have increasingly become popular and some of the earlier studies that applied the method to the question answering task include [27, 57, 104, 106]. In both works, a natural language sentence paired with its logical form is taken as input to output the classification. Although these methods showed improvement on the given task, they required expensive manual annotation.

In addition, embedding models that take as input a pair of questions and their corresponding answers to learns the vector representation of the question words and the constituents of the knowledge base [14, 79] were proposed. The learned vector representations are then used to score the natural language question against the candidate answers. The proposed approach achieve promising performance although, simply summing vectors falls short when it comes to word order information.

Furthermore, in 2015, Bodes et al [13] proposed a study that introduces a **SimpleQuestions** benchmark which consist of 108,442 simple questions annotated with the correct Freebase knowledge base triples. In this study, a memory network is proposed as the original solution to solve the task. This benchmark prompted a line of work that saw several researchers like Golub and He [41] applying character level encoder-decoder with attention mechanism to solve the simple question answering over knowledge base task. Although this approach improve the state-of-the-art, the character level encoding method results in longer sequences, and also introduces the complexity of the attention mechanism which would be of lesser importance in the simple question answering task.

In [102], authors use character-level convolutional neural network for entity linking and a separate word-level convolutional neural network with attentive max-pooling that models the relationship between the predicate and question pattern to improve the model. This approach employs attention mechanism to obtain better

matches for the relations. In this thesis, we only focus on word-level encoding and do not apply attention mechanisms.

Another related study is Dai et al's work [24] which investigated a word-level recurrent neural network (RNN). In this work, they proposed a conditional probabilistic framework using bidirectional gated recurrent units (BiGRUs) to infer the target relation first and then the target subject associated with the candidate relations. This work is trained on Freebase-specifc predicate and entity representation, and cannot be easily transferred to other KB's.

In addition, Lukovnikov et al [65], applied a hierarchical word-level and character-level question encoder to train a neural network. This model learns to rank subject–predicate pairs in an end-to-end manner to enable the retrieval of relevant facts given a question. The above techniques exploit increasingly complex end-to-end deep learning techniques that are computationally expensive and limit the opportunity to fully understand the problem structure.

However, our study takes a different approach of decomposing the question answering task into different components and solve each component separate. Moreover, our approach performs reasonably well compared to complex neural network methods and provides the opportunity to understand the problem structure. This approach is inspired by existing works [75, 89] that a similar approach of decomposing the KBQA into sub-tasks. Although they take a similar approach, our proposed method differs from these approaches in the following ways;

- Both [75] and [89] decompose the knowledge base question answering task into two components of entity detection and relation classification. Our approach decomposes the knowledge base question answering task into three components of entity detection, entity linking, and relation prediction.
- In [75], they apply only the BiLSTM on the entity detection task to find the topic entity together with attentive recurrent neural network with similarity matrix-based convolutional neural network (AR-SMCNN) for relation classification to compute the correlation between the question and candidate relation. Where as [89] applies only recurrent neural network (RNN) for both entity detection and relation prediction. Our proposed approach goes a step further to evaluate different methods on each of these tasks including BiLSTM, BiGRU, Conditional Random Fields (CRF), BiLSTM-CRF, and BiGRU-CRF on the entity detection task and BiLSTM, BiGRU, CNN and Logistic Regression (LG) for relation prediction task.

- In addition to the above differences, our proposed method treats the entity linking task as a separate task and applies string matching to solve the task. We also make an analysis to understand some of the limitations in our proposed model and how to overcome them.

Besides, the limited application of question answering systems to simple electronic edge devices brought about by the high dimension data used to train complex systems, has brought interest in the exploration of dimension reduction techniques for computation efficiency. In this regard, several studies have been proposed to model word embeddings as low dimensional linear sub-spaces extracted via principal component analysis (PCA) [35]. Although linear sub-spaces are efficient dimensional reduction techniques, they can only learn linear relationships but fall short when it comes to complex non-linear functions [2].

On the other hand, autoencoders have the ability to learn non-linear complex relationships of the input representation and generate more similar representations as the input but with a lesser dimension [42]. Despite the fact that dimensional reduction techniques have proved to be computationally efficient while at the same time achieve good results on different NLP tasks, they have not been extensively explored for the task of question answering. We take a step in exploring the dimension reduction technique for solving the question answering task.

Chapter 4

Pipeline-Based Approach for Question Answering

In this chapter, we propose an approach for efficient question answering (QA) of simple queries over a knowledge base (KB) whereby a single triple consisting of `(subject, predicate, object)` is retrieved from the KB when a natural language question is given. This approach is referred to as pipeline-based approach. Different from existing knowledge based question answering systems that make it difficult to examine the process of query processing because they exploit complex end-to-end neural network approaches that are computationally expensive and take long to execute when training the neural network. The pipeline-based approach that we propose, decomposes the question answering task into three sub-tasks of entity detection, entity linking, and relation prediction and solves each of the components separately. More precisely, our proposed approach is quite simple, it explores basic neural network and non-neural network methods for entity detection and relation prediction plus a few heuristics for entity linking. Substantial experiments performed using the `SimpleQuestions` benchmark dataset shows that our proposed approach performs reasonably well compared to previous state-of-the-art approaches. All the work introduced in this chapter are published in both [18] and [19].

4.1 Introduction

Question answering over knowledge base has been conducted using large scale knowledge bases (KB), such as Freebase [12], DBpedia [60], Wikidata [93], and YAGO [44]. These knowledge bases consist of a large pool of information with real-world entities as nodes and relations as edges. Each directed edge, along with its head entity and tail entity, constitute a triple, i.e., `(head entity, relation, tail entity)`, which is also known as a fact. Knowledge bases have in recent years become pivotal in question answering because of their ability to provide precise answers to users questions. However, because of the large volume and complex data structure, of these large scale KB's, it is difficult for non-technical users to access the substantial and valuable knowledge in them. To bridge this gap, the most convenient approach that has gained much attention is knowledge based question answering that allows end users to pose natural language questions over a knowledge base without knowledge of the underlying schema, and in return receive entities as the answers to the question. Several existing question answering studies that map a natural language question to a triple-based representation and retrieve a matching sub-graph from the knowledge base have been proposed [54, 62]. This approach, however, has a number of drawbacks such as failure to capture the original semantic structure of the question using triples. Besides, recent studies in deep learning, have seen a surge of end-to-end complex neural network approaches that perform well on a variety of natural language processing tasks like opinion extraction [47], sentence classification [55], entity linking [102], question answering [45] to mention but a few. Such end-to-end architectures suffer from long execution time when training the networks and more so, difficult to conduct detailed performance analysis in the end-to-end setting.

To overcome this challenge brought about by end-to-end complex methods, this approach focuses on simple factoid questions based on the `SimpleQuestions` benchmark [13], in which answering the question requires the extraction of a single fact from the knowledge base. The pipeline-based approach reduces the question answering task to finding a single fact i.e., `(subject, predicate, object)` in the KB that answers the question. For example, given a question `"where was Barack Obama born?"`, this approach identifies a triple, `(Barack Obama, people/person/place_of_birth, Honolulu)` from the knowledge base that contains an object entity which is the answer to the question. This study focuses on simple questions mainly because they are the most common types of questions observed in various question answering sites [102]. Although, this task is referred

to as "simple", in reality it is difficult and far from being solved. We, therefore, propose an efficient method that identifies a fact from the knowledge base that best matches the question. The proposed method bears a resemblance to answer selection [103] in which the question answering system chooses the answer from a candidate list given a question.

Furthermore, knowledge based question answering faces a challenge when identifying the matching entities and relations in the knowledge base. This is due to the fact that knowledge base entities are represented by machine identifiers (MID's), and its possible to have a single name represented by multiple MID's, for example in the freebase knowledge base, a name "the debt" is represented by four MID's (04j0t75, 0bj3wz4j, 0bjw1k11, 0dy60p) which makes it difficult to answer the question what country was the film the debt from?. Similar challenges are faced when matching relations, an example here is that three different relation types are referenced for the above question i.e., (film/film/country, film/film/written_by, music/album/release_type). These ambiguity problems, exacerbate as the scale of the knowledge base grows. For these reasons, entity linking becomes a bottleneck in knowledge base question answering and can be conjectured that a method to identify such ambiguities in the data would be crucial for the enhancement of KBQA.

In summary, this research makes the following key contributions:

- We present a simple yet effective approach to address the knowledge base question answering task. Our approach is faster, efficient and performs reasonably well compared to previous complex approaches that apply end-to-end neural network on a similar task of simple question answering.
- We establish a strong non-neural-network baseline on this task to compare with neural networks. The baseline includes CRF's (Conditional Random Fields) for entity detection and LR (Logistic Regression) for relation classification.
- We show that ambiguity in the data limits the performance. There are often multiple answers that are not easy to disambiguate and our approach identifies such ambiguities in the data which improves the performance.
- We present an empirical error analysis to gain insights into the mistakes produced and the reasons that bring about the mistakes in an attempt to improve the performance in the future work.

The rest of this chapter is organized as follows: We formulate the problem and introduce key concepts in Sect. 4.2, we provide the details of our proposed approach in Sect. 4.3. We examine experiments and provide a detailed analysis and discussion in Sect. 4.4, and conclude the Chapter in Sect. 4.5 with an outlook on future work.

4.2 Problem Definition

The goal of this study is to design a question answering system that can map a simple natural language question *q* to a matching query *Q* consisting of the subject and the relation referred to in the question that can be executed against the knowledge base *G* to retrieve the answer to the question. We restrict ourselves to simple questions, which only require the retrieval of a single fact from the knowledge base to be answered for the purpose of experiment.

Let $G = \{(Si, Pi, Oi)\}$ be the knowledge base representing a set of triples where Si represents a subject entity, Pi a predicate or relation, and Oi an Object entity. Given a natural language question *q* represented as a sequence of words, $q = \{w_1, w_2, ..., w_T\}$, the task of question answering is to identify a triple or fact from the knowledge base $(\hat{\mathtt{s}}, \hat{\mathtt{p}}, \hat{\mathtt{o}}) \in G$ such that $\hat{\mathtt{o}}$ is the intended answer to the question. A triple can also be interpreted as a directed edge, from $\mathtt{s}$ to $\mathtt{o}$.

We therefore formulate this task to finding the right subject $\hat{\mathtt{s}}$ and predicate or relation $\hat{\mathtt{p}}$ referred to in the question *q* that characterizes a set of triples in the knowledge base *G* that contain the answer $\hat{\mathtt{o}}$ to the question. The initial goal is to generate a structured query *Q* consisting of (subject, relation) from the natural language question *q* that accurately interpret the question. To generate the structured query, our approach makes two assumptions; First, we assume first-order questions that can be answered by retrieving a single fact from the KB and second, we assume that the source or subject entity is mentioned in the question. The pipeline-based approach solves the task in the following steps:

- Given a natural language question *q*, we start by identifying the entity mention m in the question.
- Link the mention m to the knowledge base and generate the entity candidate list C_e from the knowledge base comprising of all the entities associated with m and the relation candidate list C_r comprising of all the relations

connected to the candidate entities.

- We then score the semantic similarity between the question and the generated candidate sets c_e and c_r and the triple consisting of the subject entity s and predicate p with the highest score will be considered to have an object entity o as the final answer.

4.3 Proposed Approach

In this section, we provide a detailed explanation of our proposed approach. In Figure 4.1, we present an illustration of the proposed workflow which solves the question answering task in the following steps:

- Given a natural language question, we start by identifying / detecting the entity mentioned in the question.
- Then, the question is classified as one of the relation types in the knowledge base. From these two steps a structured query in the form of entity and relation is generated.
- The next step is to link the entity mention to the knowledge base and generate the entity candidate list from the knowledge base comprising of all the entities associated with the entity mentioned in the question. These entities are connected to other entities by a relation which generates a list of candidate triples from which a candidate triple with high score is identified as the answer to the question.

Different from existing models that solve the question answering task in an end-to-end setting that makes it more difficult to conduct a detailed performance analysis, this pipeline approach of decomposing the question answering task into different components provides us with an opportunity to solve each component separate and understand the problem structure. In the following paragraphs we provide details of how each component in our proposed approach is solved and the methods used.

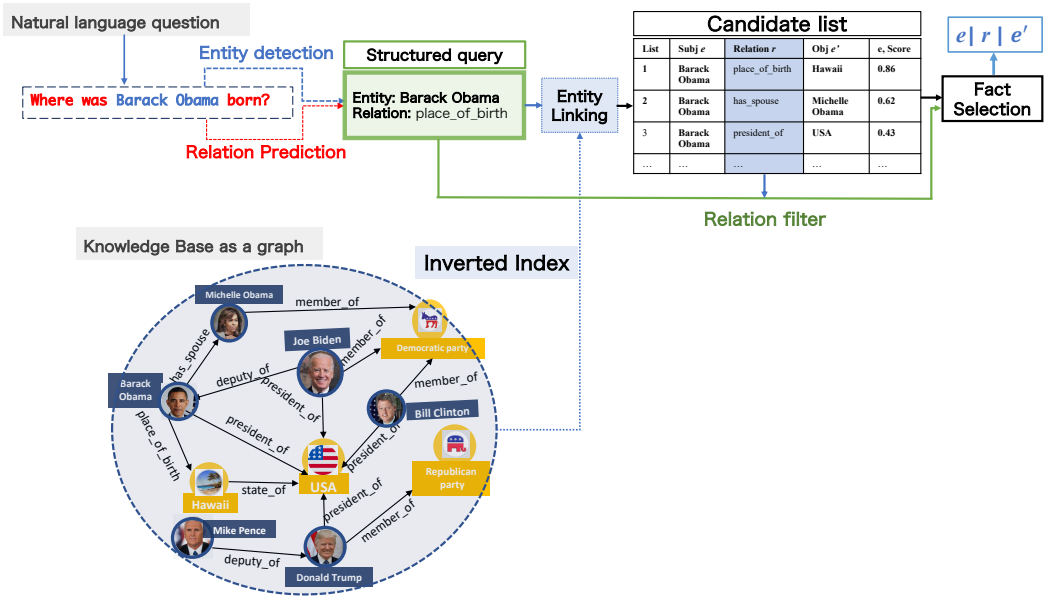


Figure 4.1: Illustration of the proposed approach, the two main components of entity detection and relation prediction compose Q the structured query from which a list of candidates is generated by an inverted index from the knowledge base G, and the list is filtered using the query relation type to select the best candidate.

4.3.1 Step 1: Entity Detection

The goal of Step 1 is to identify the entity m mentioned in the question. We formulate this as a sequence labeling problem that assigns a label (**I**: entity and **0**: non-entity) to each token in the given input question. To this end, we apply both recurrent neural networks RNN's and conditional random field (**CRF**) a non-neural network model.

Recurrent Neural Network (RNN): Recurrent neural networks [29] are capable of processing arbitrary sequential inputs by applying an activation function to the hidden vector recursively. The general idea of recurrent neural network is to come up with the output y given the input x. This is done by computing the hidden state at the current time step h_t, which is given by the input at the current time step x_t,

and the hidden state at previous time step h_{t-1}.

$$\begin{aligned} y_t &= (W_o \cdot h_t) \\ h_t &= (W_{ih} \cdot x_t + U \cdot h_{t-1} + b), \end{aligned} \tag{4.1}$$

where W and b are the weights and bias respectively.

From Figure 4.2, as we read the sentence from left to right, the RNN considers the hidden state at the previous time step to compute the hidden state at the current time step. As we go deep into the sentence, the equations for computing the hidden states at each time step becomes;

$$\begin{aligned} h_1 &= (U^\top \cdot h_0 + W^\top \cdot x_1) \\ h_2 &= (U^\top (U^\top \cdot h_0 + W^\top \cdot x_1) + W^\top \cdot x_2) \\ h_3 &= (U^\top (U^\top (U^\top \cdot h_0 + W^\top \cdot x_1) + W^\top \cdot x_2) + W^\top \cdot x_3) \end{aligned} \tag{4.2}$$

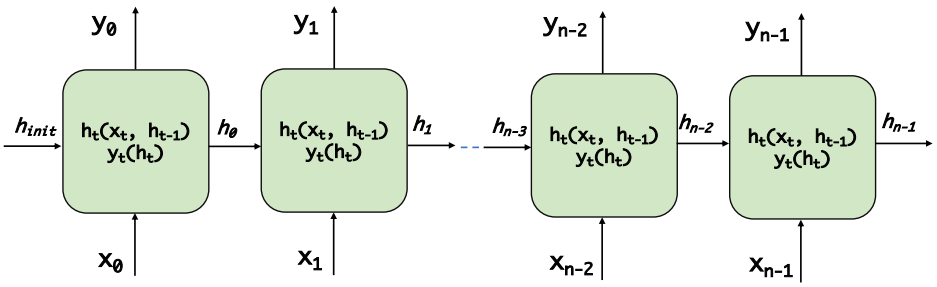


Figure 4.2: Vanilla RNN showing the processing of sequential inputs and outputs.

From Equation 4.2, we can see a long series of multiplication of small values which brings about the diminishing of gradients that causes the learning process in the

recurrent neural network to become degenerate. In other words, the recurrent neural network tends to be biased towards the most recent input [5]. To solve the Vanishing gradient tendency in RNN's, we use LSTM or long short term memory networks that are commonly used due to their ability of processing longer sequences.

In the sequence tagging task, each token in the input question is represented with a word embedding, the representation is then combined with the hidden layer representation from the previous time step using either **BiLSTMs** (Bidirection Long Short-Term Memory) [43] or **BiGRUs** (Bidirectional Gated Recurrent Units) [22] as illustrated in Figure 4.3. In our experiment, we go a step further to combine

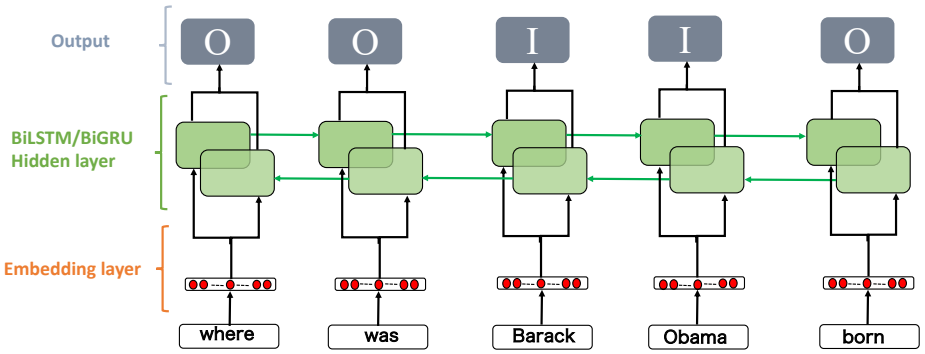


Figure 4.3: Showing sequence tagging for entity detection using either BiLSTM or BiGRU.

both neural network and non-neural network (**BiLSTM-CRF** and **BiGRU-CRF**) models to label the entity m mentioned in the question. For example, the question "`where was Barack Obama born?`" is assigned a gold label sequence "{**0, 0, I, I, 0**}".

The key idea of **LSTMs** is the cell state. As we read the sentence from left to right, the LSTM is going to have a new memory variable $\mathbf{C}^{<\mathbf{t}>}$ called the memory cell at time step $\mathbf{t}$ so that when the network gets further into the sentence it can still remember information seen earlier [37]. LSTMs have the ability to control the hidden state updates and outputs using gates. They consist of three gating functions;

the update gate Γ_u used to control the update at each time step, the forget gate Γ_f which decides the amount of information from the previous hidden state to keep or throw away, and the output gate Γ_o which regulates the flow of information in and out of the cell.

The current memory cell $\mathbf{C}^{<\mathbf{t}>}$ is computed by interpolating the previous hidden state $\mathbf{a}^{<\mathbf{t-1}>}$ and the candidate state $\tilde{\mathbf{C}}^{<\mathbf{t}>}$. The equations that govern the LSTM behavior are defined as;

$$\begin{aligned}
\hat{\mathbf{C}}^{<\mathbf{t}>} &= tanh(W_c[\mathbf{a}^{<t-1>}, X^{<t>}] + b_c), \\
\Gamma_f &= \sigma(W_f[\mathbf{a}^{<t-1>}, X^{<t>}] + b_f), \\
\Gamma_o &= \sigma(W_o[\mathbf{a}^{<t-1>}, X^{<t>}] + b_o), \\
\mathbf{C}^{<\mathbf{t}>} &= \Gamma_u \odot \hat{\mathbf{C}}^{<\mathbf{t}>} + \Gamma_f \odot \mathbf{C}^{<\mathbf{t-1}>}, \\
\mathbf{a}^{<t>} &= \Gamma_o \odot \mathbf{C}^{<\mathbf{t}>},
\end{aligned} \tag{4.3}$$

where $\hat{\mathbf{c}}^{<\mathbf{t}>}$ is the candidate cell state at time $\mathbf{t}$, $\mathbf{a}^{<\mathbf{t-1}>}$ the activation at previous time step, $X^{<t>}$ the current input, W and b are parameter and bias terms respectively. $\mathbf{c}^{<\mathbf{t}>}$ is the current internal cell state with $\odot$ as the element wise vector product. σ denotes sigmoid and *tanh* the hyperbolic tangent.

The **GRUs** on the other hand, consists of two gates; the update gate Γ_u, and the reset gate Γ_r. The update gate allows the model to learn by itself, how much of the previous information should be passed to the future. This limits the vanishing gradient problem since the model does not have to remember all the information seen previously. The reset gate tells how relevant is the previous cell state for computing the current candidate state. The GRU transition equations are defined as follows:

$$\begin{aligned}
\hat{\mathbf{C}}^{<\mathbf{t}>} &= tanh(W_c[\Gamma_r \odot \mathbf{C}^{<t-1>}, X^{<t>}] + b_c), \\
\Gamma_u &= \sigma(W_u[\mathbf{C}^{<t-1>}, X^{<t>}] + b_u), \\
\Gamma_r &= \sigma(W_r[\mathbf{C}^{<t-1>}, X^{<t>}] + b_r), \\
\mathbf{C}^{<t>} &= \Gamma_u \odot \hat{\mathbf{C}}^{<\mathbf{t}>} + (1 - \Gamma_u) \odot \mathbf{C}^{<t-1>},
\end{aligned} \tag{4.4}$$

where by $\mathbf{C}^{<t-1>}$ is the previous cell state and σ denotes the sigmoid activation function given by;

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.5}$$

σ is applied element wise to the vector.

The BiLSTMs and BiGRUs are able to capture the information flowing in both ways. They apply a non-linear transformation to compute the hidden layer representation at the current time step. The final hidden representation at the current time step is then projected to the output dimensional space and normalized into a probability distribution via a softmax layer Figure 4.3 shows the network architecture. In our sequence tagging setting, we concatenate the forward and backward hidden states $\mathbf{h}_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$. Finally the hidden states sequence obtained by either the **BiLSTM** or **BiGRU** is denoted by:

$$\begin{aligned} H_{lstm} &= BiLSTM(X_1, X_2, ..., X_n) = (\mathbf{h_1}, \mathbf{h_2}, ..., \mathbf{h_n}), \\ H_{gru} &= BiGRU(X_1, X_2, ..., X_n) = (\mathbf{h_1}, \mathbf{h_2}, ..., \mathbf{h_n}), \end{aligned} \tag{4.6}$$

here, H is the hidden states matrix, X the input sequence and $_n$ the length of the sequence.

Conditional Random Fields (CRF's): The other method that we use for entity detection is Conditional random fields [59]. **CRF** has been successfully used in many sequence labeling tasks, we therefore compare the performance between non-neural networks with neural networks on the entity detection task. They are used to represent the probability of a hidden state sequence given some observations.

For example, given the input sequences $x = (x_1, x_2, ..., x_m)$, and $s = (s_1, s_2, ..., s_m)$ the output states or label sequence, the conditional probability P_{cond} can be given by;

$$P_{cond} = p(s|x; W, b) \tag{4.7}$$

We define $\Phi(x, s) \in R^d$ a feature map that maps *x* paired with *s* to *d* a dimensional feature vector. The probability is therefore modeled as a log linear;

$$p(s|x, w) = \frac{exp(w.\phi(x, s))}{\sum_{s'} exp(w.\phi(x, s'))} \tag{4.8}$$

where $w \in R^d$ is a parameter vector with *s'* ranging over all possible outputs. The parameter vector *w* can be estimated by assuming that we have a set of *n* labeled samples $\{(x^i, s^i)\}^n$, with *i* = 1. The regularized log likelihood is given by;

$$\mathcal{L}(w) = \sum_{i=1}^{n} \log p(s^i|x^i, w) - \frac{\lambda 2}{2}||w||_2^2 - \lambda 1||w||_1 \tag{4.9}$$

where $\frac{\lambda 2}{2}||w||_2^2$ and $\lambda_1||w||_1$ forces w to be small in the respective norm. We estimate a parameter vector $w*$ as;

$$w^* = argmax(w \in R^d \mathcal{L}(w)), \tag{4.10}$$

if we are able to estimate $w*$ the parameter vector, we can then find the most likely tag of the sentence $s*$ for a sentence x by;

$$s^* = argmax_s p(s|x : w^*), \tag{4.11}$$

Combination of RNN's and CRF (BiLSTM-CRF and BiGRU-CRF): The combination of RNN's with CRF incorporates both the input sequence information as well as the information on the output sequence. As shown in Figure 4.4 the initial layer encodes the input sequence to a vector representation, the vector representations are then passed on to the BiLSTM or a BiGRU layer which captures the semantic information of the input sequence. The output is then passed to a CRF layer that calculates the probability distribution using the dependencies among labels of the entire sequence.

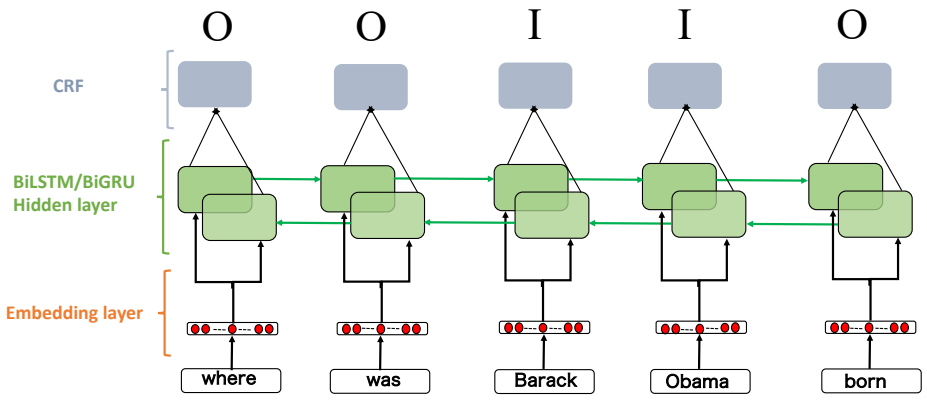


Figure 4.4: Showing sequence tagging for entity detection using either BiLSTM-CRF or BiGRU-CRF.

4.3.2 Step 2: Relation Prediction

The goal of relation prediction is to identify the relation being queried in the given natural language question. This is done by classifying the natural language question as one of the knowledge base relation types. Several methods including neural networks such as **BiLSTM**, **BiGRU**, **CNN** and non-neural networks methods like logistic regression **LR** are explored for question classification.

Recurrent Neural network(RNN): A model similar to the one used for entity detection is used and both BiLSTM and BiGRU are applied. However, relation prediction is not a tagging task since it is over the entire question. The classification decision is based on the output of the hidden layer of the last token as shown in Figure 4.5.

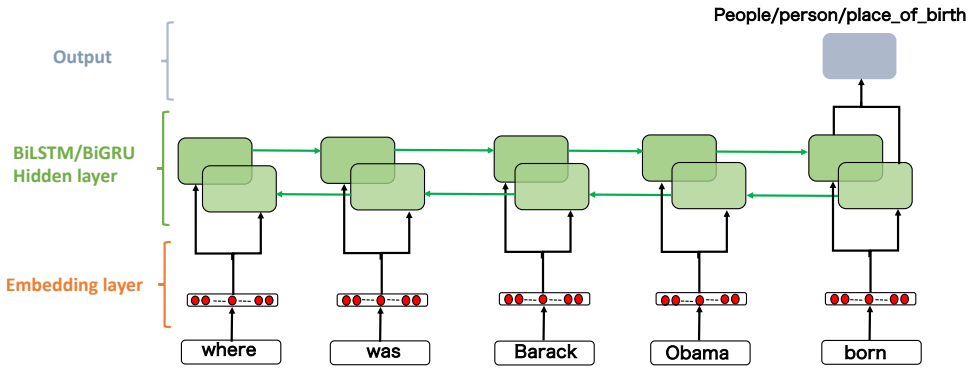


Figure 4.5: RNN architecture for relation prediction

Convolutional Neural Networks (CNNs): Figure 4.6 illustrates the application of CNN for relation classification, we apply vanilla CNNs to extract local features by sliding filters over the word embeddings. the sentence is represented by concate-

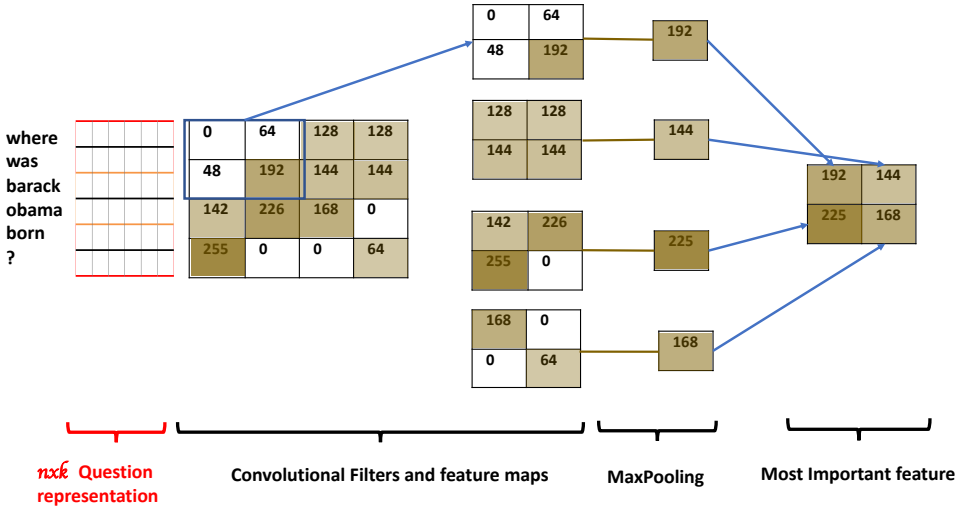


Figure 4.6: Showing how convolutional filters are applied to word window to produce a feature map to which Maxpooling is applied to reduce the size while maintaining the most important features for Relation prediction.

nating words and padding where necessary as;

$$x_{1:n} = x_1 \oplus x_2 \oplus .... \oplus x_n, \tag{4.12}$$

We use convolutional filters on the input matrix transformed into word embeddings to generate new features from a window of words represented by;

$$c_i = f(W.x_{i:i+h-1} + b), \tag{4.13}$$

The filter is applied to each of the possible window of words in the sentence to produce a feature map represented as;

$$c = [c_1, c_2, \ldots, c_{n-h+1}], \tag{4.14}$$

and finally we apply maxPooling over the filter to take the maximum value as a feature corresponding to this particular filter. The idea is to capture the most important feature, which is basically one with the highest value for each feature map. And finally these features are passed on to the fully connected softmax layer whose output is the probability distribution over labels. CNNs have shown to perform well on sentences classification [46], we adopt the architecture by [55], and modify it to a single static channel instead, and use it for relation classification.

Table 4.1: Table showing sample questions and their relation, the questions and relations are split into individual words to come up with their respective representations that is to say word embeddings from question words and bag of words from relation words.

Question	Relation
what is the place of birth of Barack Obama born?	people/person/place of birth
what country was the film the debt from?	film/film/country
which artist recorded most of us are sad?	music/recording/artist

Logistic Regression: In order to compare the performance of neural network with non-neural network methods on the relation prediction task, we apply logistic regression. In our experiment, we consider two types of features extracted from the question, that is to say; (1). `Term Frequency-Inverse Document Frequency (tfidf)`, (2). `Question Word embeddings and 1-Hot encoding of relation words`. The vector representations of the question, are obtained from both question and relation words. Table 4.1, shows sample questions and the respective relations. Question word embeddings are averaged and out-of-vocabulary words are assigned a vector of zero. Words in the relation class are split into individual tokens to come up with a vocabulary of relation words. The vector representations are concatenated as shown in Figure 4.7 to come up with the question features. This vector representation, combines word embeddings strength and one-hot encoding. Both the neural network and the non-neural network methods classify the entire question into one of the Freebase KB relation types. The relation prediction step outputs the relation type associated to the given question.

4.3.3 Step 3: Entity Linking

The entity detection and relation prediction steps outputs a structured query in form of the entity mention and the relation that accurately represent the natural language question. The identified entity sequence of tokens that represent the candidate entity in the structured query are linked to the actual entity node in the knowledge base. We apply the string matching technique to solve the entity linking task. We build an inverted index to link the extracted entity to the actual knowledge base. The inverted index maps all entity n-grams where $n \in \{1, 2, 3\}$ to the entity's name for every knowledge base entity. Borrowing from our previous example identified entity "`Barack Obama`", the corresponding n-grams are {`"Barack",`

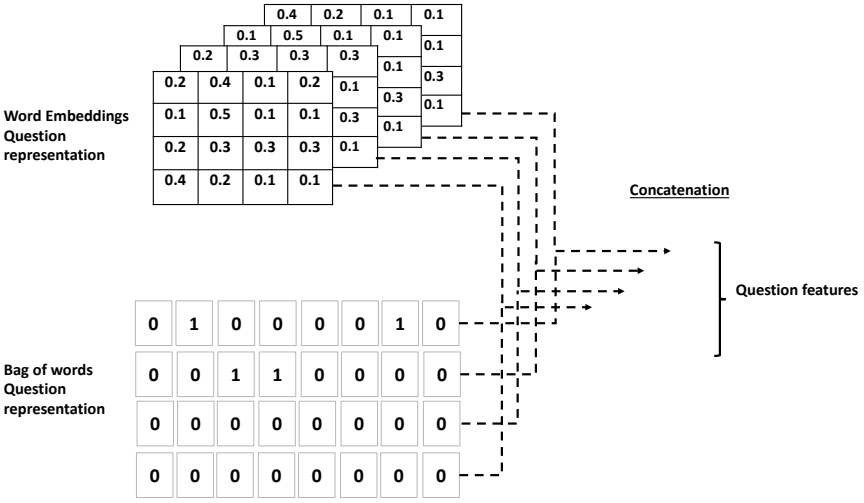


Figure 4.7: Shows the concatenation of the average word embeddings from the question words and the bag of words from the relation words to come up with the question features.

"Obama", "Barack Obama"}. All the corresponding n-grams from the entity mention are extracted, we iterate over entity n-grams in a decreasing order of n and if we find candidate values, early termination is applied to stop searching for smaller values of n. This helps in pruning entities that would have been retrieved for smaller values of n. After coming up with all possible entities from the knowledge base, candidate entity nodes are retrieved from the index and appended to the list. These candidate entities in the list are ranked using edit distance and the top candidates are kept. The process is illustrated in Algorithm 1.

Once we have a list of candidate entities, each candidate node is used as a starting point to reach candidate answers. We limit our search to a single hop and retrieve all nodes that are reachable from the candidate node where the relation path is consistent with the relation obtained from step 2. We scan the entity candidate list generated by the entity linking step and all the entity nodes with a relation type different from the one associated to the question are removed from the list, only those candidate entity nodes with a relation type leading to another node that is similar to the one generated in step 2 are kept from which the entity node with a high score in the remaining candidate list has an object entity node which is the answer to the question. This entity linking method however, introduces some ambiguity chal-

Algorithm 1 Step 3: Entity Linking

Input: Detected Entity i, Inverted Index I_{inv}
Output: Candidate List C_e
0: Initialize $C_e = \{\}$, i_n = $n - grams$ in i
0: $i_n \longleftarrow$ Sort i_n by $lengt$
0: $j \longleftarrow Max\_Lengt(i_n)$ // get_Max_Toks in i_n
for $tok \in$i$_n$ **do**
0: **if** C_e = $\{\}$ and $Tok\_Num(tok) < j$ **then**
$j \longleftarrow Tok\_Num(tok)$
0: **else if** $C_e \neq \{\}$ and $Tok\_Num(tok) < j$ **then**
break
0: **end if**
$C_e \longleftarrow C_e U I_{inv}[tok]$
end the for loop =0

lenges especially for large scale knowledge base where many entities may share the same name, making it difficult to find the correct entity.

4.3.4 Identification of Ambiguity in the Data

One of the contributions of this work is to show that there exists multiple answers to the question that are not easy to disambiguate which limits the performance of the question answering system. This is a common challenge in free open datasets and such ambiguities are most likely to arise from the annotation process. In the Simplequestions dataset, annotators are asked to write a natural language question for a corresponding triple [13]. In such a case, where one is only given a triple, it'd be difficult to anticipate possible ambiguities in the KB. We identify such ambiguities in the data.

Given a natural language question *q* with the corresponding triple (s, p, o), where s, p, and o are the subject, relation and object respectively, we aim at determining a set of all possible (s, p) pairs that accurately interpret the question q. The first step is to determine the string alias a by matching the the phrase in q (i.e., entity in the structured query) with the subject alias s in Freebase. Next, we find all other Freebase entity MID's that share this alias and add them to a set S. For example, given the question `"what country was the film the debt from?"`, from Table 4.2 we can see three examples of subject-relation pairs with equal lin-

Table 4.2: Showing examples of ambiguity in the data

Entity MID (S)	**Freebase alias (s)**	**Potential relations (P)**
fb:m.04j0t75	the debt (country)	film/film/country
fb:m.0bj3wz4j	the debt (film)	film/film/written_by
fb:m.0bjwlk11	the debt (music)	music/album/release_type

guistic evidence that can not be easily disambiguated.

After generating a set of entities S, the next step is to come up with a set of potential relations P as shown in Table 4.2. For this, we abstract a from the question i.e., "`what country was the film` $< e >$ `from?`" to determine an abstract relation p. An accurate semantic interpretation of the question q is defined if there exists a subject-relation pair $(s, p) \in KB$ where $p \in P$ and $s \in S$. In a case where multiple (s, p) pairs exist as shown in the table, the question is not answerable. To answer such a question, we predict the most likely relation $p_{max} \in P$ that makes the answer candidate to be $(s, p_{max}) \in KB$. If it happens that answer candidates are more i.e., p_{max} is more than one, then we pick s_{max} with the most facts of type p_{max}.

To summarize, the entity linking process uses the inverted index to map all the entity n-grams of the identified mention to all the corresponding nodes in the knowledge base in order to generate the candidate list of the corresponding nodes. All of the nodes in the generated list are connected to other nodes by a relation to form a triple. The candidate list is then filtered using the relation obtained in the relation prediction step. To identify the triple that contains an object entity which is the answer to the question, all the candidate triples with a relation type different from the one obtained are filtered out and the candidate triple with a highest score in the remaining candidate list becomes the answer to the question. Figure 4.8 gives a summary of the process.

4.4 Experiments

We compute precision, recall, and F1-score for every sequence tags against the ground truth for evaluation in entity detection. We evaluate recall for top results *(R@k)* for both entity linking and relation prediction to see if the correct answer appears in the top k results and the final prediction is marked as correct if both

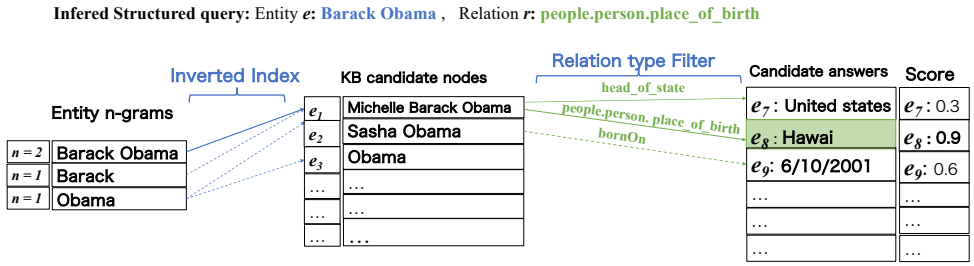


Figure 4.8: Illustration of the entity linking process. The inverted index maps entity n-grams to corresponding nodes in the knowledge base to come up with a candidate list. The relation type filter is used to filter out all candidate nodes with a different relation from that in the structured query.

entity and relation match the ground truth in end-to-end evaluation.

4.4.1 Experimental Setting

We evaluate our proposed method using the `SimpleQuestions` benchmark. This benchmark was first released by [13], and it is commonly used for studying the simple question answering task. It consists of 108,442 total questions with 75,910, 10,854, and 21,687 train, validation, and test sets respectively. Each question is associated with a triple from the freebase knowledge base that answers the question. In the experiment, we initialize word embeddings using glove vectors [73] of 300-dimension, and train on batches of 64 with a learning rate of 0.0001 and we use negative log likelihood to optimize parameters using Adam optimizer [56]. We use the FB2M freebase subset as the knowledge base in our experiments for purposes of comparison with previous work that applied a similar subset on the simple question answering task.

There are three main components in our proposed approach: entity detection, entity linking, and relation prediction. Each of these components is solved separately. In the experiment, entity detection and relation prediction experiments are conducted independently. However, the entity linking experiment depends on the output of the entity detection and therefore conducted after the entity detection.

Table 4.3: Entity detection results for both neural network, non-neural network and a combination of neural and
non-neural network methods.

	Neural Network				**Non-neural network**		**Neural net & Non-neural net**			
Metrics	BiGRU		BiLSTM		CRF		BiGRU-CRF		BiLSTM-CRF	
	Val	**Test**	**Val**	**Test**	**Val**	**Test**	**Val**	**Test**	**Val**	**Test**
Precision	92.69	92.12	92.73	92.04	90.85	90.72	97.30	**96.78**	97.32	**96.71**
Recall	93.28	93.12	93.46	92.23	89.92	89.8	96.47	**96.08**	96.49	**95.18**
F1	92.99	**92.62**	93.09	**92.63**	90.36	**90.2**	95.80	**95.63**	95.87	**95.68**

The experiment for each component (entity detection and relation prediction) was implemented using PyTorch v0.2.1 and we use fuzzywuzzy package to compute string matching scores on a PC with an Intel Core i5 (3.3 GHz) CPU with 16 GB RAM running on macOS sierra.

4.4.2 Entity Detection Evaluation

We evaluate the token level precision, recall, and F1 score. Token level represents an exact match between the predicted token and the ground truth. The results in Table 4.3, shows that neural network methods that is to say BiGRU and BiLSTM achieves a good performance with F1 score of 92.62 and 92.63 respectively, although the results from CRF a non-neural network method is comparable at 90.2.

Besides, we can observe from the Table 4.3 that a combination of BiGRU-CRF and BiLSTM-CRF achieves better results. This confirms that a combination of RNN's with CRF can boost the performance on the entity detection task.

4.4.3 Entity Linking Evaluation

Table 4.4, presents the entity liking results obtained for a given model. The table consists of two parts, the top part of the table represents results before ambiguity is resolved and the bottom part represents results after resolving the ambiguity is-

Table 4.4: Entity linking results using neural network, non-neural and a combination of neural and non-neural network methods
before and after ambiguity resolution.

	Neural Network				**Non-neural network**		**Neural net & Non-neural net**			
Top-*k*	BiGRU		BiLSTM		CRF		BiGRU-CRF		BiLSTM-CRF	
	Val	**Test**	**Val**	**Test**	**Val**	**Test**	**Val**	**Test**	**Val**	**Test**
1	0.675	**0.662**	0.679	**0.662**	0.663	**0.649**	0.680	**0.671**	0.682	**0.672**
5	0.823	0.810	0.827	0.811	0.809	0.796	0.854	**0.814**	0.857	**0.816**
10	0.860	0.849	0.889	0.876	0.871	0.861	0.892	**0.879**	0.895	**0.880**
20	0.885	0.876	0.912	0.903	0.895	0.889	0.914	**0.904**	0.917	**0.906**
Result after ambiguity is resolved										
1	0.806	**0.761**	0.810	**0.780**	0.726	**0.713**	0.812	**0.804**	0.813	**0.804**
5	0.876	0.870	0.881	0.871	0.852	0.850	0.910	**0.883**	0.911	**0.889**
10	0.905	0.904	0.908	0.904	0.874	0.871	0.916	**0.910**	0.921	**0.912**
20	0.913	0.910	0.919	0.914	0.899	0.893	0.921	**0.915**	0.924	**0.917**

sue. We evaluate the recall of the top-k entity candidates which is obtained as a percentage of questions whose top-k candidates includes the ground truth.

It is observed that before and after ambiguity is taken care of, the result obtained by CRF a non-neural network method is comparable to the neural network on the entity linking task. Although CRF may have performed slightly lower than BiLSTM and BiGRU on the entity detection task, the bottleneck is being able to link the detected entity mention to the correct entity node in the knowledge base. Furthermore, we can observe that when CRF is combined with the neural network, the performance is boosted in both cases before and after ambiguity is resolved. On comparing the entity detection results in Table 4.3 and entity linking results in Table 4.4, we can observe that before resolving the ambiguity issue, all the models perform far below on the entity linking task when Top-k=1 compared to the entity detection and this is due to the existence of ambiguous entities in the knowledge base that made it difficult to extract the correct entity. However, we can see an im-

Table 4.5: Relation Prediction results for both Neural and non-neural network methods.

	Neural Network						**Non-neural network**			
Accuracy@-*k*	BiGRU		BiLSTM		CNN		LR(TF-IDF)		LR(W2Vec+1-Hot)	
	Val	**Test**	**Val**	**Test**	**Val**	**Test**	**Val**	**Test**	**Val**	**Test**
1	82.22	**81.59**	81.61	**81.10**	82.88	**81.92**	73.36	**73.64**	71.64	**72.31**
3	97.75	93.68	93.59	93.35	93.75	93.68	85.67	**86.39**	86.70	**87.11**
5	95.93	95.76	96.10	95.52	95.86	95.64	88.58	**89.16**	90.12	**91.63**

provement on the entity linking result obtained by all models after the ambiguity is resolved.

4.4.4 Relation Prediction Evaluation

In Table 4.5, we observe that, at k=1, CNN outperforms both BiGRU and BiLSTM. We also observed that neural network methods (i.e., BiGRU, BiLSTM, and CNN) perform much better than non neural network (i.e., Logistic Regression) on the relation prediction task.

From the Table 4.5, we can also observe that logistic regression performs better on test set than the validation set, and this is mainly due to the fact that we used logistic regression with default parameters. Another important observation to note is that training each of the models component, i.e., entity detection and relation prediction for 50 epochs using our PC, it takes approximately 8 hours which is quicker and efficient compared to training Lukovinikov et al. [65] which takes close to 6 days for the same number of epochs.

4.4.5 Answering the Question Correctly

Figure 4.9 shows test set accuracy results for obtaining the correct answer to the question. Our best model which combines the BiGRU-CRF for entity detection and

BiGRU for relation prediction achieves 80.27% when the ambiguity in the data is taken care of. From the figure we can observe that a combination of non-neural network methods i.e., CRF for entity detection and logistic regression (W2Vec+1-Hot) for relation prediction also, achieve a good performance of up to 75.26% when ambiguity is taken care of. The performance of our proposed non-neural network methods is way higher compared to the original paper Bodes et al., [13] which applied complex memory networks.

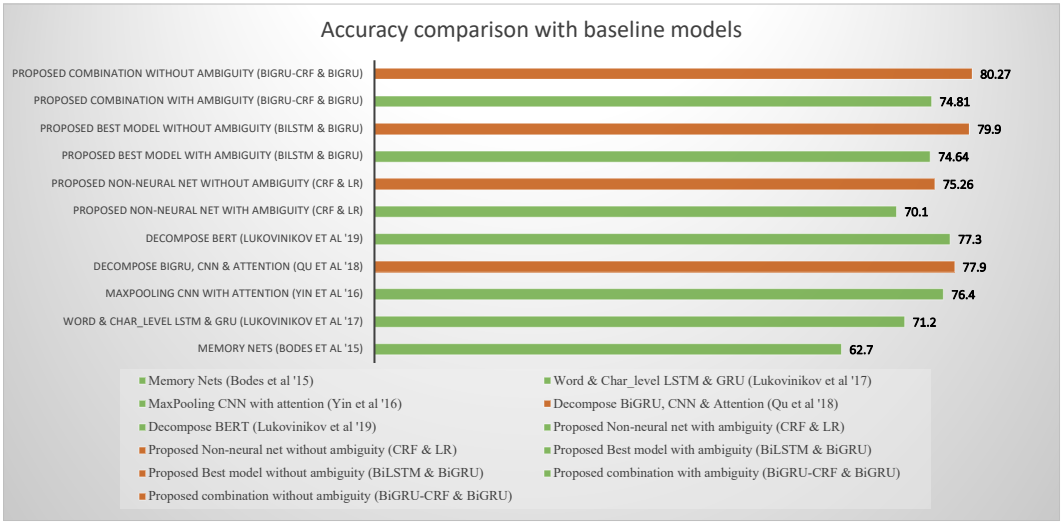


Figure 4.9: Performance comparison of our approach with existing methods on the task of question answering. The green color represents models when there is ambiguity in the data.

More results from different experimental settings conducted are presented in Table 4.6 with an extensive comparison to other state-of-art methods. We compare our results with other existing state-of-art complex models in Table 4.6 to examine the necessity of model complexity on this task. It can be observed that our best results outperforms the existing models that apply complex neural network. With this result, we are convinced that the improvement directly attributed to complex neural networks on the simple question answering task is modest.

Table 4.6: Accuracy results for selecting the correct fact from the knowledge base that has the object entity which answers the question.

Entity Detection (Model)	Relation Prediction (Model)	Accuracy (Before Ambiguity resolution)	Accuracy (After Ambiguity resolution)
BiGRU-CRF	BiGRU	74.81	80.27
BiLSTM-CRF	BiLSTM	74.79	80.3
BiLSTM	BiGRU	74.64	79.9
BiLSTM	CNN	74.63	79.3
CRF	CNN	73.42	78.63
CRF	BiGRU	73.39	78.61
CRF	BiLSTM	73.34	78.6
CRF	LR(W2Vec+1-Hot)	70.1	75.26
CRF	LR(TF-IDF)	68.4	70.66

Existing Baselines							
Model	Year	Description		Accuracy	Before Ambiguity resolution	Accuracy	After Ambiguity resolution
Zhou et al. [107]	2021	Deep fused end-to-end model	BiGRU & CNN		79.0		...
Lukovnikov et al. [64]	2019	Decompose into sub-tasks	BERT		77.3		...
Xiao et al. [45]	2019	Embedding based	BiLSTM with attention		75.4		...
Qu et al. [75]	2018	Decompose into sub-tasks	BiGRU & CNN with attention		...		77.9
Lukovnikov et al. [65]	2017	Word and character encoding	LSTMs & GRUs		71.2		...
Yin et al. [102]	2016	Max pooling	CNN with attention		76.4		...
Dai et al. [24]	2016	Probabilistic	BiGRU		75.7		...
Golub & He. [41]	2016	Character level encoding	LSTM with attention		70.9		...
Bodes et al. [13]	2015	Memory networks			62.7		...

4.4.6 Error Analysis

We make more analysis to understand some of the errors in our model and the main cause of these errors in an attempt to improve the performance.

- We considered the questions that were retrieved in the second position (hits = 2) but not in the top position and we found that 733 questions were retrieved in the second position.
- We also considered questions that were retrieved in the third position (hits = 3) but not in the first and second position and we found 203 questions.

We performed the analysis using RNN for entity detection and RNN for relation prediction using the top 50 entities and the top 5 relations for answering the question on the validation set. We found out that errors mainly occurred during entity linking and Figure 4.10 shows the causes of the error.

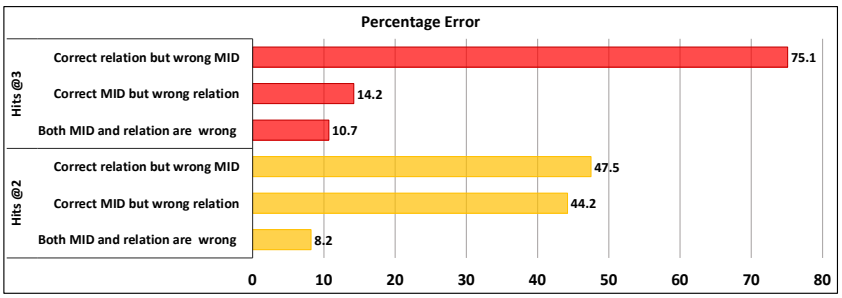


Figure 4.10: Shows the main causes of error that limit the performance of our model.

Generally, the two main reasons for the error were; incorrect query which was due to errors propagated during the entity detection phase and incorrect linking where it was noticed that in almost all cases, different MID's had the same entity name and the model was unable to disambiguate the correct entity from the incorrect one. We observed 33.9% of such examples (3675 of 10846) in the simplequestions dataset. Such examples that were found to have multiple MID's with the same name include the name `"holywood"` which has 270 entity MID's, the subject name `"chalie`

chaplin", which has 20 entity MID's to mention but a few. Such cases make it difficult to answer the questions. To overcome this, the relation frequency for each subject in the KB was taken into account which improves the performance of our model to a new accuracy as shown in Figure 4.9.

Another observation was a number of questions that did not reference a subject. For example a question "which book is written about?" does not reference the corresponding subject "01n7q: California". Furthermore, it was observed that many questions where a correct MID was seen but with a wrong relation were hard to disambiguate even for humans. For example, the relation referred to by the question "which release was reading on?" is relation "music/release_track/recording", the classifier however, predicted "music/release_track/release".

4.5 Summary of this Chapter

In this chapter we address the simple question answering task using the SimpleQuestions dataset. In this task, questions can be answered by retrieving a single fact from the freebase knowledge base. We address this problem by decomposing the question answering task into sub-tasks namely: entity detection, entity linking, and relation prediction.

We conduct extensive experiments using different models for each sub-task of entity detection and relation prediction. The aim is to try and bring the task down to the simplest possible models that perform really well. The results show that neural network methods perform well on the relation prediction task but not so much on the entity linking task.

We also present evidence that ambiguities in the data limits performance on the simplequestions dataset and we present a method to deal with it. Our main contribution in this work is establishing strong baselines one that uses simple neural-net and one without neural-net. We are convinced that our baselines will encourage future researchers to adequately examine and take advantage of simple baselines to fully understand the simplequestions problem structure.

Our approach applies simple baseline methods that achieve results comparable to state of the art methods that apply complex neural networks and attention

mechanisms. This is surprising given the proven strength of attention mechanisms. Although we did not apply attention mechanisms on the simple questions task, we believe that they might be of help in future work when we extend our approach to more complex questions.

Chapter 5

Dimension Reduction for Question Answering

In this chapter, we present a novel approach to reconstruct embedding dimensions to a low dimension for solving the question answering task. As earlier mentioned, it has increasingly become a common practice for QA deep learning models to represent words as vectors or embeddings in the vector space in order to compute the similarity measure between questions and the answer candidate because the vector representations encode semantic information. These vector representations are usually in dimensions of 100, 300 or even more which makes the question answering models costly in terms of memory and training time. For this reason, we explore the dimension reduction approach to efficiently solve the question answering task. In the following sections, we describe the background of the research, problem definition, model descriptions, and the experiments. The work described in this chapter is also published in [17].

5.1 Introduction

In recent years, question answering (QA) system have successfully achieved promising results, these results are largely attributed to the sheer size of available knowledge and the application of deep learning models [69]. The existing major approaches for QA systems are; text-based, and knowledge-based.

As mentioned in Section 2.1.1, for the text-based QA, a most similar candidate answer is identified as the answer to the question from a list of candidate answers [1], whereas the knowledge-based QA approach, relies on knowledge bases like DBpedia [60] to be able to answer the user's question. The existing methods for solving QA systems are categorised into two; 1). Methods based on semantic parsing, and 2). Information retrieval methods. Semantic parsing methods usually convert the natural language question to a logical form, while information retrieval methods retrieve a set of answer candidates or facts from the underlying knowledge base. The question and the retrieved facts are then embedded into vector representations for measuring the similarity between the given natural language question and candidate facts from which the closest fact is selected to be the answer.

It has increasingly become a common practice in recent years for QA deep learning models to represent words as vectors or embeddings in the vector space in order to compute similarity measure between questions and the answer candidate because the vector representations encode semantic information [68]. While the application of deep learning models for question answering has been successful, the high dimensional embedding representations comes at a high memory and computational cost. This hinders the potential of applying question answering systems to simple electronic edge devices such as smart watches, mobile phones, and also internet of things (IOT).

For this reason, some studies have proposed to model word embeddings as low dimensional linear sub-spaces extracted via principal component analysis (PCA) [35]. Although linear sub-spaces are efficient dimensional reduction techniques, they can only learn linear relationships but fall short when it comes to complex non-linear functions [2].

However, auto-encoders have been introduced as network architectures for compressing the input into a latent representation and reconstruct the latent representation such that it is so close in similarity to the original input [4]. Auto-encoders have the ability to learn non-linear complex relationships of the input representation and generate representations that are more similar to the input but with a lesser dimension [42]. Despite the fact that dimensional reduction techniques have proved to be computationally efficient while at the same time achieve good results on different NLP tasks, they have not been extensively explored for the task of question answering.

In this study, we explore the dimension reduction technique for solving the question answering task. To be specific, we apply the Long Short Term Memory auto-

encoder (LSTM-AE) [61] to generate low dimensional vectors. The LSTM-AE, captures both the semantic information in word embeddings and the syntactic information in word order. The auto-encoder architecture consists of an encoder whose role is to encode the original input data into a compressed representation in the latent space, and the decoder which reconstructs the data from the encoded representation so that the deference between the outputs and the original inputs is minimal. The reconstructed low dimensional vectors become the inputs to train the matching function that learns to measure the similarity between the natural language question and the candidate answers. We make the following contributions:

- We propose and analyze an auto-encoder framework that learn low dimension representations of the original input embeddings in the latent space, and then reconstructs latent representation such that the error between the input and the low dimension output is minimized.
- We show that our proposed framework retains as much information as possible and minimize the reconstruction error between the original and the reconstructed data when training the similarity matching function.
- Our experiment and the analysis of results indicate that we can achieve reasonable performance on the question answering task while reducing computational time and memory requirements.

The remaining sections in this chapter are organized as follows: In Sect. 4.2, we formulate the problem and propose our approach, Sect. 4.3, examines experiments, and provides a detailed analysis and discussion of our findings. and we conclude the chapter in Sect. 4.4 with an outlook on the future directions.

5.2 Proposed Method

We propose to apply an LSTM-AE to focus on the most relevant features of the input embeddings and use these most relevant features to train a question answering model. If we consider natural language questions given as a set Q, and the underlying knowledge base given as a set of candidate answers $\mathcal{F}$, the aim is to train a question answering model that can correctly answer the question given a set of questions and the corresponding candidate answers. For every natural language

question, answer candidates are first retrieved from the underlying knowledge base to generate pairs of questions and their corresponding answer candidates.

5.2.1 Formal Definition

Given $q \in Q$ a natural language question, and its candidate answers $A = \{a_1, a_2, ..., a_n\}$, where by $A \in \mathcal{F}$ comprises of a ground truth answer and the sampled negative answers, each $q \in Q$ and each answer candidate $a \in A$ comprises of a sequence of tokens.

We start by learning the representations (embeddings) of the given question and its corresponding candidates answers, we then apply the LSTM-AE to generate low dimensional embedding representation from the the original embeddings, and then compute the similarity to measure the matching degree between the question and the candidate answers using the generated representations. The overall flow of the proposed QA approach is illustrated in Figure 5.1

Let $\mathbf{T}$ be a set of questions and their corresponding candidate answers;

$$\mathbf{T} = \{(q_i, A_i), ..., \}, \tag{5.1}$$

where $i = 1, 2, ..., |T|$, We define the embedding matrix;

$$\mathcal{W} \in \mathbb{R}^{k * \mathbf{N}}, \tag{5.2}$$

where, k is the embedding dimension,

$$\mathbf{N} = \mathbf{N_Q} + \mathbf{N_A}, \tag{5.3}$$

$\mathbf{N}$ is the embedding dictionary with $\mathbf{N_Q}$ and $\mathbf{N_A}$ the total number of questions and answers respectively.

The vector representation of the question and the answer are obtained using the embedding matrix $\mathcal{W}$ as shown in Figure 5.2.

where $f(q)$ and $g(a)$ are the functions that map the question and the candidate answer to the embedding space respectively. We are able to learn the question and candidate answer embeddings by learning a scoring function $S(q, a)$ as;

$$S = f(q)^\top \cdot g(a) = W\phi(q)^\top \cdot W\varphi(a) \tag{5.4}$$

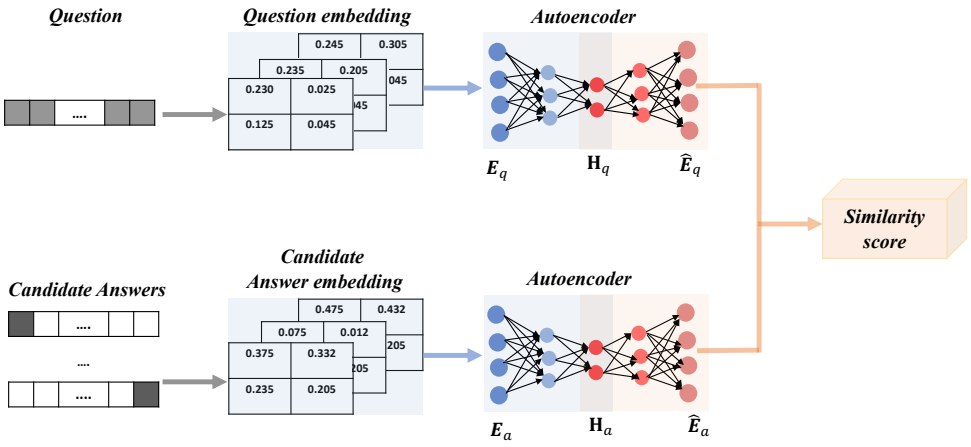


Figure 5.1: Illustration of the proposed approach. The question (E_q) and answer (E_a) embedding modules generate the dimensional vector representation of words that are input to the auto encoder for dimension reduction and the reconstructed question ($\hat{E}_q$) and candidate answer ($\hat{E}_q$) representations are used to measure the similarity score.

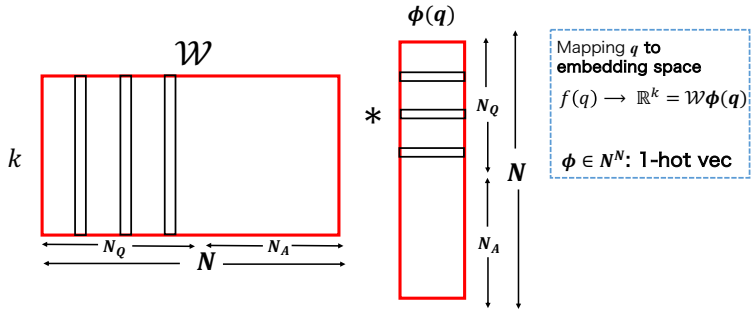
The question and candidate answer embedding sequences $E_q \in R^k$ and $E_a \in R^k$ are denoted as;

$$\begin{aligned} E_q \in R^k &= \{e_{q_1}, e_{q_2} ... e_{q_w}\}, \\ E_a \in R^k &= \{e_{a_1}, e_{a_2} ... e_{a_m}\}, \end{aligned} \tag{5.5}$$

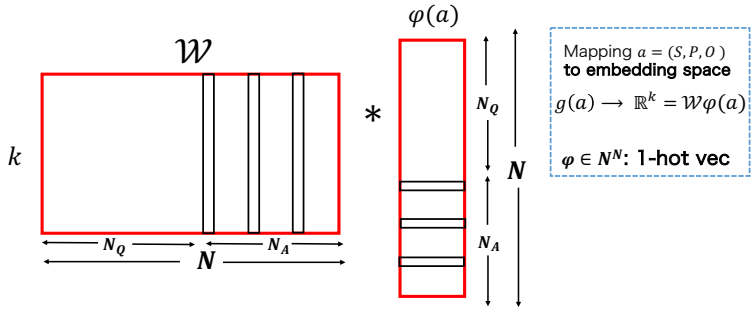
Given the learned question and candidate answer embeding sequences as in Equation 5.5, we train our question answering model using a loss function that ranks the answer to the question based on the margin.

As previously seen in equation 5.1, $\mathbf{T}$ is the training set of questions paired with their candidate answers. We minimize the loss function as;

$$\sum_{i=1}^{|T|} \sum_{\hat{a} \in \hat{A}(a_i)} \Big(max\{0, \vartheta - S(q_i, a_i^-) + S(q_i, a_i)\} \Big), \tag{5.6}$$



(a) Mapping the question to the embedding space using embedding matrix.



(b) Mapping the candidate triples to the embedding space using embedding matrix.

Figure 5.2: Shows how vector representation are obtained. Each figure correspond to mapping the question and the answer to the embedding space.

where ϑ is the margin. Minimizing the loss above, learns the embedding matrix in Equation 5.2 such that the score of the correct answer a is greater than the score with any incorrect answer a_i^- by at least the margin ϑ.

As earlier proposed, we apply an auto-encoder to focus on the most relevant features of the input embeddings and use these most relevant features to train a question answering model. The questions and candidate answers embedding sequences obtained in Equation 5.5 are the inputs to the auto-encoder.

5.2.2 Auto-Encoder for Question Answering

The auto-encoder [105] attempts to copy the inputs of the network to its output. Auto-encoders are a special type of feed forward network applied in various tasks such as compression, denoising, sparse representation, and data generation. In case of compression, the auto-encoder takes as input a large dimension vector and maps it to a vector representation of lesser dimension which retains as much information as possible such that the original input can be reconstructed from the vector representation of lesser dimension. It is made up of the encoder that maps the input to a latent representation, and a decoder that reconstructs the latent representation back to the original input Figure 5.3.

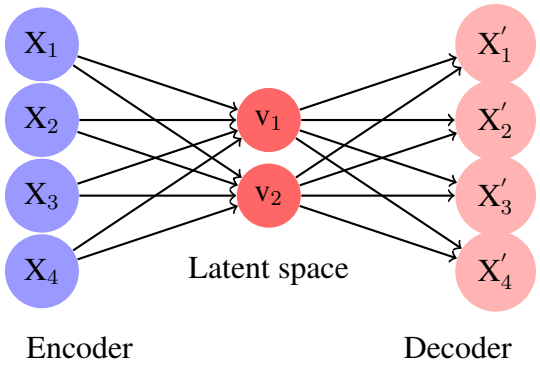


Figure 5.3: The conceptual illustration of a vanilla auto-encoder, consisting of the encoder which encodes the input embeddings to a low dimension, the latent-space which is a low dimensional representation of the original input that focus on the most important attributes, and the decoder which decompresses the representation back to the original domain.

Concretely, for a simple auto-encoder, the encoder part takes as input $\mathbf{x} \in R^k$ and maps it to $\mathbf{h} \in R^m$ as.

$$\mathbf{h} = \sigma(\mathbf{W}_f\mathbf{x} + \mathbf{b}) \tag{5.7}$$

where $\mathbf{h}$ is the latent representation, σ the element-wise activation function, $\mathbf{W}_f$ the weight matrix that maps the input to the latent representation, and $\mathbf{b}$ the bias vector. The decoder part, reconstructs the original input but with lesser dimension $\tilde{\mathbf{x}} \in R^m$ from the latent representation $\mathbf{h}$ as.

$$\tilde{\mathbf{x}} = \tilde{\sigma}(\mathbf{W}_g\mathbf{h} + \tilde{\mathbf{b}}) \tag{5.8}$$

where $\mathbf{W}_g$ is the weight matrix that reconstructs the latent representation to $\tilde{\mathbf{x}} \in R^m$ the approximated original input. The auto-encoder is then trained to find the wights $\mathbf{W}_f$ and $\mathbf{W}_g$ that minimize the reconstruction error using the objective function;

$$\mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|^2 = \|\mathbf{x} - \tilde{\sigma}(\mathbf{W}_g(\sigma(\mathbf{W}_f\mathbf{x} + \mathbf{b})) + \tilde{\mathbf{b}})\|^2 \tag{5.9}$$

where $\mathbf{x} \in R^k$ and $\tilde{\mathbf{x}} \in R^m$ are the original input and the approximated or the reconstructed respectively and $\mathbf{x}$ is always averaged over the training set. For the case of a linear auto-encoder, the objective in Equation 5.9 can also be written as;

$$\min_W \frac{1}{2} \sum_n \|\mathbf{W}_g\mathbf{W}_f\mathbf{x}_n - \mathbf{x}_n\|_2^2 \tag{5.10}$$

In such a case, where we use an euclidean norm i.e., the squared loss, it is equivalent to the principal component analysis (PCA). In other words, the solution for the linear auto encoder is the same as PCA.

For a non-linear auto-encoder where the functions f and g are non-linear, our objective function becomes;

$$\min_W \frac{1}{2} \sum_n \|g(f(\mathbf{x}_n; \mathbf{W}_f); \mathbf{W}_g) - \mathbf{x}_n\|_2^2 + c\|f(\mathbf{x}_n; \mathbf{W}_f)\|_1 \tag{5.11}$$

we add a regularization term $c\|f(\mathbf{x}_n; \mathbf{W}_f)\|_1$ to constrain the auto-encoder. The auto-encoder architecture used in our proposed approach is illustrated in Figure 5.4.

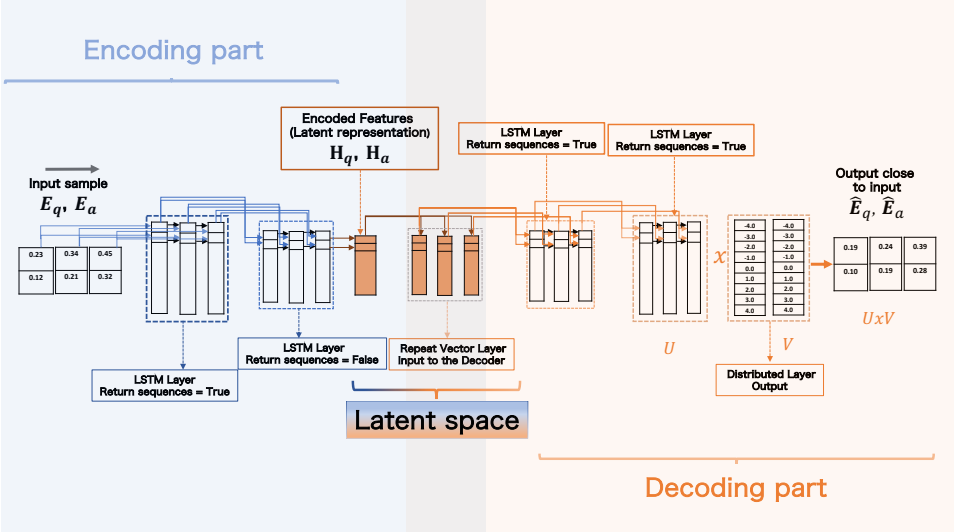


Figure 5.4: Illustrating the auto-encoder architecture, consisting of the encoder which encodes the input embeddings (E_q for question and E_a for the answer) to a low dimension, the latent-space which is a low dimensional representation ($\mathbf{H_q}$ for question and $\mathbf{H}_a$ for the answer) of the original input that focus on the most important attributes, and the decoder which decompresses the encoded representations ($\hat{E}_q$ for question and $\hat{E}_a$ for the answer).

It consists of the encoder network, the latent space, and the decoder network. The encoder and decoder consists of multiple layers.

The question embedding $\boldsymbol{E}_q$ and candidate answer embeddings $\boldsymbol{E}_a$ obtained in Equation 5.5, are the inputs to the encoder network. When the question embeddings $\boldsymbol{E}_q$ and the candidate answer embedding $\boldsymbol{E}_a$ are fed to the encoder network, they are compressed to the latent representations $\mathbf{H}_q$ of the question, and $\mathbf{H}_a$ of the answer candidate. The decoder network then decompresses the encoded representations to $\hat{\boldsymbol{E}}_q$ of question, and $\hat{\boldsymbol{E}}_a$ of the candidate answer with dimension less than that of the input. The decoder outputs are compared to the initial inputs to update the network weights. We adopt the LSTM-AE by [61] which uses the encoder to compress the inputs into a representation and uses this representation to reconstruct it back by the decoder.

Given the inputs question and candidates answer sequence embeddings 5.5, the encoder compresses them to obtain latent representations of the question and answer

candidates as;

$$\begin{aligned}\mathbf{H_{q_{enc}}} &= f(\boldsymbol{E}_q) \in R^m = Bi\_LSTM_{Enc_q}(e_{q_t}, h_{t-1}),\\ \mathbf{H_{a_{enc}}} &= f(\boldsymbol{E}_a) \in R^m = Bi\_LSTM_{Enc_a}(e_{a_t}, h_{t-1}),\end{aligned} \tag{5.12}$$

where the dimension m of the latent representation is less than dimension k of the input embeddings $(m < k)$. e_{q_t}, e_{a_t} represent input word embeddings at a given time step t, and h_{t-1} the time step of the previous state.

Given $\mathbf{H_{q_{enc}}}$ and $\mathbf{H_{a_{enc}}}$ the encoded question and answer representation, the decoder which is connected to the encoder via a latent space also known as the undercomplete [36], is tasked with generating the input sequence. Learning the under complete representation, allows the decoder to capture or focus on the salient features of the data in the course of decoding. The decoder reconstructs the latent representation back to the output sequence by sequentially predicting the next token y_t at every time step using the current input and the state at the previous time step y_{t-1} until the last token by conditional probability. From the definition of conditional probability;

$$p(y) = \prod_{t=1}^{T} p(y_t|\{y_1, ..., y_{t-1}\}, c) \tag{5.13}$$

$$\begin{aligned}y_q &= f_D(\mathbf{H_{q_{enc}}}) = F_D(W_d\mathbf{H_{q_{enc}}} + b),\\ y_a &= f_D(\mathbf{H_{a_{enc}}}) = F_D(W_d\mathbf{H_{a_{enc}}} + b),\end{aligned} \tag{5.14}$$

where f_D is the mapping function, F_D the decoder activation function, W_d the weighted matrix and b is the bias. From the above definition, the conditional probability is modeled as;

$$\begin{aligned}p(\hat{e}_{q_t}|\{\hat{e}_{q_1}, ..., \hat{e}_{q_{t-1}}\}, c) &= g(\hat{e}_{q_{t-1}}, s_t, c),\\ p(\hat{e}_{a_t}|\{\hat{e}_{a_1}, ..., \hat{e}_{a_{t-1}}\}, c) &= g(\hat{e}_{a_{t-1}}, s_t, c),\end{aligned} \tag{5.15}$$

where $\hat{e}_{q_t}$ and $\hat{e}_{a_t}$ are the question and answer candidates input embeddings at time step t, g is the non-linear activation function that out puts the probability, s_t is the decoder hidden state, and c the parameters. The auto-encoder learns how to reconstruct the original inputs by optimizing the objective function to minimize the reconstruction error as;

$$J_R = \frac{1}{N}\sum_{i<N} E(e_i, \hat{e}_i), \tag{5.16}$$

where e and $\hat{e}$ are the original and the reconstructed data point respectively while E is the mean squared error. Therefore, the final reconstructed question and candidates answer representation generated by the decoder are denoted by;

$$\begin{aligned} \hat{q} &= Bi\_LSTM_{Dec_q}(\hat{e}_{q_t}, h_{t-1}), \\ \hat{a} &= Bi\_LSTM_{Dec_a}(\hat{e}_{a_t}, h_{t-1}), \end{aligned} \tag{5.17}$$

For each reconstructed question $\hat{q}$, there is a reconstructed positive or ground truth and the reconstructed negative answers $\hat{a}$ sampled from the whole answer space for training. The goal is to design a similarity matching model using the reconstructed representations for question answering. To do this, we adopt the cosine similarity;

$$S = \cos(\theta) = \frac{\hat{q} \cdot \hat{a}}{||\hat{q}||||\hat{a}||}, \tag{5.18}$$

cosine similarity is a common measure for similarity matching and we use it for computing the similarity matching between the question and answer pairs. We train the question answering model such that the score of the function S is high if $\hat{a}$ is the correct answer of the question $\hat{q}$ and low if the answer is incorrect. During training, we minimize the following loss function;

$$J_H = \sum_{i=1}^{|T|} \sum_{\hat{a} \in A} \Big(max\{0, \vartheta - S(\hat{q}_i, \hat{a}^-) + S(\hat{q}_i, \hat{a})\} \Big), \tag{5.19}$$

where ϑ is the margin, $\mathcal{S}$ the cosine similarity score, and $\hat{a}^-$ is sampled from the negative candidate answers. The similarity between the question, positive answer, and the question, negative answer are compared to the margin ϑ during the process of training the loss function.

- If $\vartheta \geq \mathcal{S}(q_i, a_i) - \mathcal{S}(q_i, a_i^-)$, then either the positive answer is ranked below the negative answer or positive answer ranking is not sufficiently above the negative one.
- If $\vartheta \leq \mathcal{S}(q_i, a_i) - \mathcal{S}(q_i, a_i^-)$, another candidate answer is sampled from the candidate list (no update is made to the parameters) until a margin value less than ϑ is obtained.

We formally define our overall loss function as:

$$J = J_R + J_H, \tag{5.20}$$

where J_R is the reconstruction error loss and the mean squared error loss (MSE) is used here. J_H is the triplet or hinge loss for comparing the distance between the two similarities of question ground truth answer and question incorrect answer to the margin. We train an end to end question answering architecture as shown in Figure 5.1.

5.3 Experimental Evaluation

In this section, we evaluate our proposed question answering approach on the InsuaranceQA benchmark dataset and compare our result with existing baseline approaches.

5.3.1 Data

The proposed approach was evaluated on the InsuranceQA dataset [30], the dataset contains real data collected from the insurance library website [1]. we chose to evaluate our proposed method on this benchmark for purposes of comparison with the existing state-of-art methods evaluated on the same benchmark. In our experiment set up, we use the version v1 dataset which has a total of 12,887 train questions, 1,000 validation questions, and 3,600 test questions together with 18,540 train answers, 1,454 validation answers, and 2,616 test answers respectively. Also, a question may have more than one answer. In our experiment, we set the candidate answer size to 500 for each question, this includes the ground truth, and the randomly selected candidates from the entire dataset.

5.3.2 Implementation Details

We implement our proposed method using tensorflow [2] version 2.5.0 on a google Tesla T4 GPU and we conduct two sets of experiment;

- In the first setting, word embeddings were initialized with pre-trained word2vec

[1] https://www.insurancelibrary.com/
[2] https://www.tensorflow.org/

embeddings of dimension 100 [68]. The tokens that are absent in the pre-trained embeddings are replaced by $< UNK >$ representing an unknown token and every sentence (question/answer) is marked by an end of sentence token $< eos >$. Our autoencoder consists of two encoder and decoder layers one of 64 and the other of 32 dimensions. We use the Adam optimizer [56] and train in an end-to-end setting our question answering model for 100 epochs on batches of size 32, with a 0.001 learning rate, a margin value of 0.7 and a dropout regularizer of 0.00001. We test different hyperparameters and report the one with best result.

- In the second setting, we conduct the same experiment using bidirectional encoder representations BERT [26]. BERT is a model trained for language representation, it has a wide range of application in solving natural language processing tasks. It can be used for extracting quality language features from text as well as for fine tuning on a given task. For this task, we opt for the former, and generate word embeddings for our own vocabulary using BERT. In our experiment, we use base BERT of size 768 to extract embeddings and the two hidden layers in the encoder and decoder are set to 384 and 192 each.

5.3.3 Evaluation

We evaluate our model by comparing the convergence of the loss function during training. For this, we compare three sets of experiments;

- Training question answering model using the initial input embedding dimension,
- Training the question answering model using low dimension representation reconstructed by the auto-encoder network with a non-linear hyperbolic tangent activation function applied on the network layer (non-linear auto-encoder),
- Training a question answering model using low dimension representation reconstructed using auto-encoder network with a linear activation function applied to the network layer (Linear auto-encoder). This is equivalent to principal component analysis (PCA) [74].

In the inference, we evaluate our proposed method using Mean Average Precision (MAP) in order to compare the performance of our proposed approach, with existing

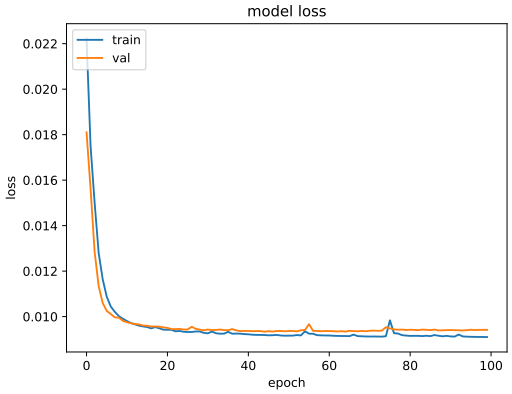
methods. MAP is a commonly used metric for inference, it quantifies how good the model performs at a given question.

$$MAP = \frac{1}{Q}\sum_{q=1}^{Q} aveP(q) \tag{5.21}$$

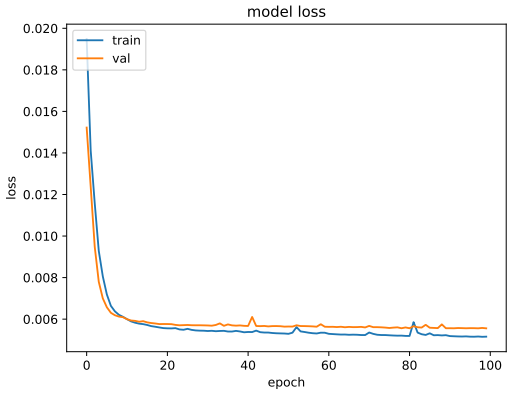
where Q is the number of questions and $aveP$ is the average precision for a given question q. Given a question, its corresponding $aveP$ is calculated and the mean of all these $aveP$ scores gives us the mean average precision.

5.3.4 Results and Discussion

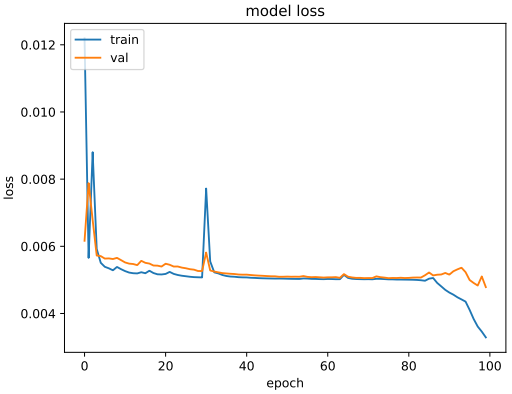
Figure 5.5, shows the performance comparison of the three experimental settings conducted using the word2vec embeddings. From left to right; a) the initial embedding dimension, b) reconstructed low embedding dimension using non-linear auto-encoder , and c) reconstructed low embedding dimension using linear auto-encoder. We observe a faster convergence when using the reconstructed lower embedding dimension. The initial word2vec embedding dimension had a better accuracy of 0.932 ± 0.076. We however, did not see a statistical significance in the result. Which further confirms that a time efficient approach in this case the low dimension embedding reconstruction method is worth considering.



(a) Using initial word2vec embedding dimension.



(b) Using low embedding dimension generated by Non-linear AE.



(c) Using low embedding dimension generated by linear AE.

Figure 5.5: Shows the train and validation loss for each of the models used; (a). The initial word2vec embedding dimension, (b). Low embedding dimension using non-linear auto-encoder, and (c). Low embedding dimension using linear auto-encoder.

We report in Table 5.1, the train and validation mean accuracy ($\pm$ standard deviation) of each of the QA model and the time it takes to train each of the model for 100 Epochs. The highlighted values represent a statistical significance.

Table 5.1: Showing mean accuracy values for the training and validation, and the time taken to train each model setting for 100 epochs using word2vec embeddings.

QA Model set up	**Accuracy**		**Time (minutes)** for 100 Epochs
	Train	**Val**	
Word2Vec	0.932 ± 0.077	0.923 ± 0.076	122.42 ± 0.38
Non-Linear AE	0.902 ± 0.066	$\mathbf{0.895 \pm 0.071}$	$\mathbf{97.35 \pm 0.17}$
Linear AE	0.797 ± 0.079	$\mathbf{0.690 \pm 0.077}$	$\mathbf{97.02 \pm 0.15}$

We observed that the QA model using the initial embedding dimension takes 30 minutes more to train with respect to the same model using reconstructed embedding dimension in both the linear and non-linear case.

Training the QA model on the initial word2vec embedding dimension seems to have a better accuracy value of 0.932, a 0.030 higher compared to 0.902 obtained using low dimensional reconstructed embeddings from non-linear auto-encoder. However, considering the $\pm$ standard deviation, this difference is not statistically significant. With this result, we may conclude that a time efficient approach in this case the low dimension embedding reconstruction method is worth considering.

In contrast, we observe a statistical significance in the validation accuracy when training the QA model using low embedding dimension from non-linear AE and those from linear AE, although the non-linear AE takes 0.33 ± 0.03 more minutes to train. We believe that a non-linear AE maybe more suitable for this specific task since it can capture the non-linear transformations. It is important to note that all the results reported in Table 5.1 were obtained using word2vec embeddings.

Inference The inference results of both the existing methods, and our proposed method are reported in Table 5.2. Results from the existing methods are reported in the first group of the table. These are the results reported by the authors in the paper.

From the Table 5.2 of results, our proposed approach performs reasonably well

Table 5.2: performance comparisons between our experiment set up and existing works, the time taken per question and memory cost in different experiment settings. In our experiment, the candidate answer pool is set to 500 candidates for each question.

Model (Set up)		Test set (one)	Test set (two)	Time (in seconds))	Memory (in GB))
Setting 1. word2vec embeddings					
Word2Vec	Initial dimension	66.9 $\pm$ 0.26	60.75 $\pm$ 0.13	2.28 $\pm$ 0.03	7.29 $\pm$ 0.03
Non-linear AE	Lesser dimension	64.52 $\pm$ 0.21	60.3 $\pm$ 0.04	0.13 $\pm$ 0.02	5.45 $\pm$ 0.02
Linear AE	Lesser dimension	53.5 $\pm$ 0.14	51.0 $\pm$ 0.09	0.08 $\pm$ 0.01	5.24 $\pm$ 0.03
Setting 2. BERT embeddings					
base BERT	Initial dimension	74.35 $\pm$ 0.11	70.83 $\pm$ 0.33	3.19 $\pm$ 0.04	11.14 $\pm$ 0.24
Non-linear AE	Lesser dimension	73.1 $\pm$ 0.10	70.28 $\pm$ 0.05	0.26 $\pm$ 0.03	6.63 $\pm$ 0.07
Linear AE	Lesser dimension	67.43 $\pm$ 0.15	67.21 $\pm$ 0.06	0.14 $\pm$ 0.04	6.40 $\pm$ 0.1
Existing works					
Feng et al [30]	2015	65.3	61.0	...	...
Wang et al [95]	2016	67.0	61.5	...	...
Ming et al [87]	2016	69.0	64.8	...	...
Gurevych at al [81]	2017	70.0	...	...	...

compared to the existing baselines on both test sets. It can be observed from the table that the result obtained using the low embedding dimension generated by a non-linear auto-encoder are better than the results obtained using the low embedding dimension generated by a linear auto-encoder for both word2vec and base BERT experiment settings. Furthermore, we compare the time taken and the memory usage in our experimental settings.

Word2vec setting: The time taken per question in the initial word2vec embedding dimension setting is 2.28 $\pm$ 0.03 seconds which is almost 18 times slower than the result from the non-linear auto-encoder setting where the time taken per question is 0.13 $\pm$ 0.02 seconds. The time cost per question observed in the linear auto-encoder setting is 28 times less compared to the initial word2vec embedding settings.

In the table we also compare the memory cost in all settings of our experiment, the Word2vec initial embedding dimension, non-linear auto-encoder, and the linear auto-encoder. The memory cost is 7.27 $\pm$ 0.03 GB, 5.45 $\pm$ 0.02 GB, and 5.24 $\pm$ 0.03 GB respectively. From the table, the Word2vec initial embedding dimension setting consumes approximately 2GB more of memory compared to the auto-encoder settings which further confirms that our proposed approach can be effective.

base BERT setting: In our experiment, we also train a question answering model using BERT word emebddings to compare the result with the word2vec because unlike word2vec which does not consider word context during representation, BERT trains word representations that are informed by the words around them. In the experiment we use BERT with a 768 dimension. As expected, BERT outperforms the previous works and other experimental settings in the table on the question answering task. This performance is mainly due to the context-informed word embeddings that result in more accurate feature representations. Although BERT performs way better in terms of accuracy, it suffers from long execution time and memory issues compared to word2vec i.e base BERT takes 0.19 $\pm$ 0.01 sec more and 3.85 GB more compared word2vec as seen from the table. However, we observed that applying BERT embeddings to both linear and non-linear auto-encoders, is both time and memory efficient and achieve good accuracy. The time and memory cost comparison between our method and existing methods could not be possible because previous work does not make this kind of analysis. We also would like to mention that the current proposed LSTM-AE used in our experiment does not apply attention mechanisms to solve the QA task. For this reason, the performance of our method on word2vec is less compared to the existing methods that apply attention mechanisms.

5.4 Summary of this Chapter

In this study, we proposed a new question answering approach using low embedding dimension generated by the long short term memory auto-encoder (LSTM-AE). We also apply a linear activation function on each of the auto-encoder layer to model low embedding dimension which is equivalent to the principal component analysis PCA.

Although many existing question answering embedding based methods achieve good performance, they use the initial embedding dimesnion and less attention has been paid to the long execution time and memory cost. To measure how effective our proposed approach can be, we conduct experiment using word2vec and BERT embedding representation. The experiments showed that solving the question answering task using the low embedding dimension generated by the LSTM-AE is time and memory efficient and achieves reasonable performance.

In the future work, we wish to extend our approach on the question answering task in the following directions: First, we seek to consider other dataset and other pre-trained vector representations with varying dimensions. Second, we wish to apply our proposed model to other question answering architectures with attention mechanism like transformers and compare the performance in terms of accuracy, time, and memory.

Chapter 6

Conclusions and Future Work

In this dissertation, we propose two approaches to efficiently solve the question answering task.

In chapter 4, we proposed a pipeline-based approach where the knowledge base question answering task is decomposed into 4 sub-tasks of entity detection, entity linking, and relation prediction and we solve each of the components separately. We explore different models for the entity detection and relation prediction tasks with the aim of establishing strong baselines; one that uses simple neural networks and one without neural-networks. Experimental results show that baseline methods achieves results comparable to state-of-the-art methods that apply complex neural networks and attention mechanisms. We also present a detailed analysis of the experimental results to understand some of the errors in our model and the main cause of these errors in an attempt to improve the performance.

Chapter 5, explores the application of dimension reduction techniques to solve the question answering task. Specifically, we apply an auto-encoder to generate low dimensional embeddings from the original input embedding dimension of the question and the candidate answers. We design a similarity matching function to measure the semantic similarity between the input question and the candidate answers. Experiments using real world insuaranceQA benchmark, shows that the proposed approach can obtain performance comparable to the standard baselines while remaining cost efficient on both time and memory.

6.1 Summary of the Contributions

In this dissertation, we make the following contributions:

- We present a simple yet effective approach to address the knowledge base question answering task, our approach is faster, efficient, and performs reasonably well compared to previous complex approaches that apply end-to-end neural network on a similar simple question answering (Chapter 4).
- We establish a strong non-neural-network baseline on this task to compare with neural networks. The baseline includes Conditional Random Field (CRF) for entity detection and Logistic Regression (LR) for relation prediction (Chapter 4).
- We show that there exist ambiguities in the data which limits the performance, i.e., there are often multiple answers that are not easy to disambiguate and our approach identifies such ambiguities in the data which improves the performance (Chapter 4).
- We also present an empirical error analysis to gain insights into the errors and the reasons that bring about these error in an attempt to improve the performance in the future work (Chapter 4).
- We propose and analyze an auto-encoder framework that learn low dimension representations of the original input embeddings in the latent space, and then reconstructs latent representation such that the error between the input and the low dimension output is minimized (Chapter 5).
- We show that our proposed framework retains as much information as possible and minimize the reconstruction error between the original and the reconstructed data when training the similarity matching function (Chapter 5).
- We conduct experiments and analysis using real world insuaranceQA benchmark, and the results indicate that we can achieve reasonable performance on the question answering task while reducing computational time and memory requirements (Chapter 5).

6.2 Future Work

In the future, we wish to extend our work on the question answering task in the following directions:

First, in chapter 4 we seek to extend our approach to more complex questions such as Luo et al., [66] and ExAQT [49], we also plan to consider other datasets like COMPLEXWEBQUESTIONS [86], LC-QuAD [28] with varying embeding dimensions. In addition, our approach did not apply attention mechanisms [3] on the simple questions task. However, we believe that they might be of help in future work when we extend our approach to more complex questions.

Second, in chapter 5 we are considering to explore other datasets and other pre-trained vector representations such as glove [73] with varying dimensions. Besides, we believe that it is equally important to apply the proposed approach to other question answering architectures with attention mechanism like transformers [90] and compare the performance in terms of accuracy, time, and memory.

In long term goal, we plan to to extend our research of question answering to ever changing knowledge bases. In the real world knowledge changes over time, but most of the existing KBQA approaches often assume a static knowledge base making it difficult for a typical KBQA system to answer questions related to the new knowledge when the KB evolves since it lacks the ability to detect knowledge unseen in the training. With the current progress in life-long machine learning, where a model is trained to adapt to new tasks while preserving its capability on previous tasks [39], we hope to propose a novel life-long KBQA learning framework that can progressively expand learning capacity as humans do.

In their work, Pan et al., [72] shows that it is possible to identify the most relevant information in past data and use it to regularize the training of the same model on new tasks, avoiding catastrophic forgetting. This idea has a natural extension to our problem, where past data represents the old version of the knowledge base, and the new task is learning to answer question relevant to new facts added to the knowledge base.

Bibliography

[1] Zahra Abbasiantaeb and Saeedeh Momtazi. Text-based question answering from information retrieval and deep neural network perspectives: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, page e1412, 2021.

[2] Jasem Almotiri, Khaled Elleithy, and Abdelrahman Elleithy. Comparison of autoencoder and principal component analysis followed by neural network for e-learning using handwritten recognition. In *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, pages 1–5. IEEE, 2017.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[4] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49. JMLR Workshop and Conference Proceedings, 2012.

[5] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[6] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[7] Jonathan Berant and Percy Liang. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, 2014.

[8] Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510, Doha, Qatar, October 2014. Association for Computational Linguistics.

[9] Matthew W Bilotti, Jonathan Elsas, Jaime Carbonell, and Eric Nyberg. Rank learning for factoid question answering with linguistic and semantic constraints. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 459–468, 2010.

[10] Matthew W Bilotti, Paul Ogilvie, Jamie Callan, and Eric Nyberg. Structured retrieval for question answering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 351–358, 2007.

[11] Daniel G Bobrow, Ronald M Kaplan, Martin Kay, Donald A Norman, Henry Thompson, and Terry Winograd. Gus, a frame-driven dialog system. *Artificial intelligence*, 8(2):155–173, 1977.

[12] Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In Jason Tsong-Li Wang, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250. ACM, 2008.

[13] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075, 2015.

[14] Antoine Bordes, Jason Weston, and Nicolas Usunier. Open question answering with weakly supervised embedding models. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 165–180. Springer, 2014.

[15] Jan A. Botha, Zifei Shan, and Daniel Gillick. Entity Linking in 100 Languages. In *Proceedings of the 2020 Conference on Empirical Methods in*

Natural Language Processing (EMNLP), pages 7833–7845, Online, November 2020. Association for Computational Linguistics.

[16] Jan A Botha, Zifei Shan, and Daniel Gillick. Entity linking in 100 languages. *arXiv preprint arXiv:2011.02690*, 2020.

[17] Happy Buzaaba and Toshiyuki Amagasa. A scheme for efficient question answering with low dimension reconstructed embeddings. In *In Proceedings of the 23nd International Conference on Information Integration and Web-based Applications & Services, iiWAS2021, linz austria November 29 - December 1, 2021*.

[18] Happy Buzaaba and Toshiyuki Amagasa. A modular approach for efficient simple question answering over knowledge base. In Sven Hartmann, Josef Küng, Sharma Chakravarthy, Gabriele Anderst-Kotsis, A Min Tjoa, and Ismail Khalil, editors, *Database and Expert Systems Applications - 30th International Conference, DEXA 2019, Linz, Austria, August 26-29, 2019, Proceedings, Part II*, volume 11707 of *Lecture Notes in Computer Science*, pages 237–246. Springer, 2019.

[19] Happy Buzaaba and Toshiyuki Amagasa. Question answering over knowledge base: A scheme for integrating subject and the identified relation to answer simple questions. *SN Comput. Sci.*, 2(1):25, 2021.

[20] Jinwei Cao and Jay F Nunamaker. Question answering on lecture videos: a multifaceted approach. In *Proceedings of the 2004 Joint ACM/IEEE Conference on Digital Libraries, 2004.*, pages 214–215. IEEE, 2004.

[21] Yonggang Cao, Feifan Liu, Pippa Simpson, Lamont Antieau, Andrew Bennett, James J Cimino, John Ely, and Hong Yu. Askhermes: An online question answering system for complex clinical questions. *Journal of biomedical informatics*, 44(2):277–288, 2011.

[22] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[23] Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. TyDi QA: A

benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470, 2020.

[24] Zihang Dai, Lei Li, and Wei Xu. CFO: Conditional focused neural question answering with large-scale knowledge bases. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 800–810, Berlin, Germany, August 2016. Association for Computational Linguistics.

[25] Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In *Extended Semantic Web Conference*, pages 106–120. Springer, 2010.

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[27] Li Dong, Furu Wei, Ming Zhou, and Ke Xu. Question answering over Freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 260–269, Beijing, China, July 2015. Association for Computational Linguistics.

[28] Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon, editors, *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*, volume 11779 of *Lecture Notes in Computer Science*, pages 69–78. Springer, 2019.

[29] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[30] Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. Applying deep learning to answer selection: A study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 813–820. IEEE, 2015.

[31] Paolo Ferragina and Ugo Scaiella. Fast and accurate annotation of short texts with wikipedia pages. *IEEE Softw.*, 29(1):70–75, 2012.

[32] David Ferrucci. Build watson: An overview of deepqa for the jeopardy! challenge. In *2010 19th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 1–1, 2010.

[33] David A Ferrucci. Introduction to "this is watson". *IBM Journal of Research and Development*, 56(3.4):1–1, 2012.

[34] Justin Scott Giboney, Susan A Brown, Paul Benjamin Lowry, and Jay F Nunamaker Jr. User acceptance of knowledge-based system recommendations: Explanations, arguments, and fit. *Decision Support Systems*, 72:1–10, 2015.

[35] Hongyu Gong, Tarek Sakakini, Suma Bhat, and JinJun Xiong. Document similarity for texts of varying lengths via hidden topics. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2341–2351, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[36] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[37] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[38] Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224, 1961.

[39] Haibo He, Sheng Chen, Kang Li, and Xin Xu. Incremental learning from stream data. *IEEE Transactions on Neural Networks*, 22(12):1901–1914, 2011.

[40] Jiyin He and Maarten de Rijke. An exploration of learning to link with wikipedia: Features, methods and training collection. In Shlomo Geva, Jaap Kamps, and Andrew Trotman, editors, *Focused Retrieval and Evaluation, 8th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2009, Brisbane, Australia, December 7-9, 2009, Revised and Selected Papers*, volume 6203 of *Lecture Notes in Computer Science*, pages 324–330. Springer, 2009.

[41] Xiaodong He and David Golub. Character-level question answering with attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1598–1607, Austin, Texas, November 2016. Association for Computational Linguistics.

[42] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[43] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[44] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61, 2013.

[45] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 105–113, 2019.

[46] Mark Hughes, I Li, Spyros Kotoulas, and Toyotaro Suzumura. Medical text classification using convolutional neural networks. *Stud Health Technol Inform*, 235:246–250, 2017.

[47] Ozan İrsoy and Claire Cardie. Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 720–728, Doha, Qatar, October 2014. Association for Computational Linguistics.

[48] Heng Ji and Ralph Grishman. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1148–1158, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[49] Zhen Jia, Soumajit Pramanik, Rishiraj Saha Roy, and Gerhard Weikum. Complex temporal question answering on knowledge graphs. In Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong, editors, *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pages 792–802. ACM, 2021.

[50] Kelvin Jiang, Dekun Wu, and Hui Jiang. FreebaseQA: A new factoid QA data set matching trivia-style question-answer pairs with Freebase. 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 318–323, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[51] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *J. Documentation*, 60(5):493–502, 2004.

[52] Mandar Joshi, Eunsol Choi, Daniel 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, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[53] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, USA, 1st edition, 2000.

[54] Esther Kaufmann, Abraham Bernstein, and Lorenz Fischer. Nlp-reduce: A naive but domainindependent natural language interface for querying ontologies. In *4th European Semantic Web Conference ESWC*, pages 1–2, 2007.

[55] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[56] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[57] Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1223–1233, 2010.

[58] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov.

Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

[59] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[60] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[61] Jiwei Li, Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1106–1115, Beijing, China, July 2015. Association for Computational Linguistics.

[62] Vanessa Lopez, Victoria Uren, Enrico Motta, and Michele Pasin. Aqualog: An ontology-driven question answering system for organizational semantic intranets. *Web semantics: science, services and agents on the world wide web*, 5(2):72–105, 2007.

[63] Vanessa Lopez, Victoria Uren, Marta Reka Sabou, and Enrico Motta. Cross ontology query answering on the semantic web: an initial evaluation. In *Proceedings of the fifth international conference on Knowledge capture*, pages 17–24. ACM, 2009.

[64] Denis Lukovnikov, Asja Fischer, and Jens Lehmann. Pretrained transformers for simple question answering over knowledge graphs. In Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon, editors, *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I*, volume 11778 of *Lecture Notes in Computer Science*, pages 470–486. Springer, 2019.

[65] Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. Neural network-based question answering over knowledge graphs on word and character level. In Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy

Gabrilovich, editors, *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 1211–1220. ACM, 2017.

[66] Kangqi Luo, Fengli Lin, Xusheng Luo, and Kenny Zhu. Knowledge base question answering via encoding of complex query graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2185–2194, 2018.

[67] Rada Mihalcea and Andras Csomai. Wikify!: linking documents to encyclopedic knowledge. In Mário J. Silva, Alberto H. F. Laender, Ricardo A. Baeza-Yates, Deborah L. McGuinness, Bjørn Olstad, Øystein Haug Olsen, and André O. Falcão, editors, *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*, pages 233–242. ACM, 2007.

[68] 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*, volume 26. Curran Associates, Inc., 2013.

[69] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas, November 2016. Association for Computational Linguistics.

[70] David N. Milne and Ian H. Witten. Learning to link with wikipedia. In James G. Shanahan, Sihem Amer-Yahia, Ioana Manolescu, Yi Zhang, David A. Evans, Aleksander Kolcz, Key-Sun Choi, and Abdur Chowdhury, editors, *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*, pages 509–518. ACM, 2008.

[71] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. November 2016.

[72] Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard E. Turner, and Mohammad Emtiyaz Khan. Continual deep learning by functional regularisation of memorable past. In Hugo Larochelle,

Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[73] 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.

[74] Elad Plaut. From principal subspaces to principal components with linear autoencoders. *CoRR*, abs/1804.10253, 2018.

[75] Yingqi Qu, Jie Liu, Liangyi Kang, Qinfeng Shi, and Dan Ye. Question answering over freebase via attentive rnn with similarity matrix based cnn. *arXiv preprint arXiv:1804.03317*, 38, 2018.

[76] Pranav Rajpurkar, Robin Jia, and Percy 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)*, pages 784–789, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[77] 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, Austin, Texas, November 2016. Association for Computational Linguistics.

[78] Siva Reddy, Mirella Lapata, and Mark Steedman. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392, 2014.

[79] Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 5418–5426. Association for Computational Linguistics, 2020.

[80] Dmitri Roussinov and José A Robles-Flores. Applying question answering technology to locating malevolent online content. *Decision Support Systems*, 43(4):1404–1418, 2007.

[81] Andreas Rücklé and Iryna Gurevych. Representation learning for answer selection with lstm-based importance weighting. In Claire Gardent and Christian Retoré, editors, *IWCS 2017 - 12th International Conference on Computational Semantics - Short papers, Montpellier, France, September 19 - 22, 2017*. The Association for Computer Linguistics, 2017.

[82] Gerard Salton. *The SMART retrieval system—experiments in automatic document processing*. Prentice-Hall, Inc., 1971.

[83] Robert P Schumaker and Hsinchun Chen. Leveraging question answer technology to address terrorism inquiry. *Decision Support Systems*, 43(4):1419–1430, 2007.

[84] Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460, 2014.

[85] Robert F. Simmons. Answering english questions by computer: a survey. *Commun. ACM*, 8(1):53–70, 1965.

[86] Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. 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 641–651, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[87] Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 464–473, Berlin, Germany, August 2016. Association for Computational Linguistics.

[88] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[89] Ferhan Ture and Oliver Jojic. No need to pay attention: Simple recurrent neural networks work! In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2866–2872, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[90] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[91] Shahper Vodanovich, David Sundaram, and Michael Myers. Research commentary—digital natives and ubiquitous information systems. *Information Systems Research*, 21(4):711–723, 2010.

[92] Ellen M. Voorhees and Dawn M. Tice. The TREC-8 question answering track. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)*, Athens, Greece, May 2000. European Language Resources Association (ELRA).

[93] Denny Vrandecic. Wikidata: a new platform for collaborative data collection. In Alain Mille, Fabien Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab, editors, *Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume)*, pages 1063–1064. ACM, 2012.

[94] David L Waltz. An english language question answering system for a large relational database. *Communications of the ACM*, 21(7):526–539, 1978.

[95] Bingning Wang, Kang Liu, and Jun Zhao. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1288–1297, Berlin, Germany, August 2016. Association for Computational Linguistics.

[96] Terry Winograd. Procedures as a representation for data in a computer program for understanding natural language. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC, 1971.

[97] Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. Exploiting rich syntactic information for semantic parsing with graph-to-sequence model. In Ellen Riloff, David Chiang, Julia Hockenmaier, and

Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 918–924. Association for Computational Linguistics, 2018.

[98] Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[99] 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-November 2018. Association for Computational Linguistics.

[100] Wen-tau Yih, Xiaodong He, and Christopher Meek. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 643–648, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[101] Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany, August 2016. Association for Computational Linguistics.

[102] Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. Simple question answering by attentive convolutional neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1746–1756, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

[103] Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*, 2014.

[104] Luke S. Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *CoRR*, abs/1207.1420, 2012.

[105] Junhai Zhai, Sufang Zhang, Junfen Chen, and Qiang He. Autoencoder and its various variants. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 415–419. IEEE, 2018.

[106] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. Variational reasoning for question answering with knowledge graph. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6069–6076. AAAI Press, 2018.

[107] Guangyou Zhou, Zhiwen Xie, Zongfu Yu, and Jimmy Xiangji Huang. DFM: A parameter-shared deep fused model for knowledge base question answering. *Inf. Sci.*, 547:103–118, 2021.

Reference Papers

Journal Paper

- Happy Buzaaba, Toshiyuki Amagasa, "Question Answering Over Knowledge Base: A scheme for Integrating Subject and the Identified Relation to Answer Simple Questions" *SN Computer Science, Vol. 2, No. 1, Article 25, 2021*.

Conference Paper

- Happy Buzaaba, Toshiyuki Amagasa, "A Scheme for Efficient Question Answering with Low Dimension Reconstructed Embeddings" *The 23rd International Conference on Information Integration and Web Intelligence (iiWAS2021)*, pp=303–310, Linz, Austria, November 29 - December 1, 2021.

- Happy Buzaaba , Toshiyuki Amagasa, "A modular approach for efficient simple question answering over knowledge base" *In Proceedings of the 30th International Conference on Database and Expert Systems Applications (DEXA2019)*, pp. 237-246, Linz, Austria, August 26-29, 2019.

Other Papers

Journal Paper

- David Ifeoluwa Adelani, Jade Abbott, Graham Neubig, Daniel D'souza, Julia Kreutzer, Constantine Lignos, Chester Palen-Michel, Happy Buzaaba, Shruti Rijhwani, Sebastian Ruder, Stephen Mayhew, Israel Abebe Azime, Shamsuddeen H. Muhammad, Chris Chinenye Emezue, Joyce Nakatumba-Nabende, Perez Ogayo, Aremu Anuoluwapo, Catherine Gitau, Derguene Mbaye, Jesujoba Alabi, Seid Muhie Yimam, Tajuddeen Rabiu Gwadabe, Ignatius Ezeani, Rubungo Andre Niyongabo, Jonathan Mukiibi, Verrah Otiende, Iroro Orife, Davis David, Samba Ngom, Tosin Adewumi, Paul Rayson, Mofetoluwa Adeyemi, Gerald Muriuki, Emmanuel Anebi, Chiamaka Chukwuneke, Nkiruka Odu, Eric Peter Wairagala, Samuel Oyerinde, Clemencia Siro, Tobius Saul Bateesa, Temilola Oloyede, Yvonne Wambui, Victor Akinode, Deborah Nabagereka, Maurice Katusiime, Ayodele Awokoya, Mouhamadane MBOUP, Dibora Gebreyohannes, Henok Tilaye, Kelechi Nwaike, Degaga Wolde, Abdoulaye Faye, Blessing Sibanda, Orevaoghene Ahia, Bonaventure F. P. Dossou, Kelechi Ogueji, Thierno Ibrahima DIOP, Abdoulaye Diallo, Adewale Akinfaderin, Tendai Marengereke, Salomey Osei "MasakhaNER: Named Entity Recognition for African Languages" *In Transactions of the Association for Computational Linguistics (TACL2021)* 9: 1116-1131 (MIT Press Direct)

Conference Paper

- David Ifeoluwa Adelani, Jade Abbott, Graham Neubig, Daniel D'souza, Julia Kreutzer, Constantine Lignos, Chester Palen-Michel, Happy Buzaaba, Shruti Rijhwani, Sebastian Ruder, Stephen Mayhew, Israel Abebe Azime, Shamsuddeen H. Muhammad, Chris Chinenye Emezue, Joyce Nakatumba-

Nabende, Perez Ogayo, Aremu Anuoluwapo, Catherine Gitau, Derguene Mbaye, Jesujoba Alabi, Seid Muhie Yimam, Tajuddeen Rabiu Gwadabe, Ignatius Ezeani, Rubungo Andre Niyongabo, Jonathan Mukiibi, Verrah Otiende, Iroro Orife, Davis David, Samba Ngom, Tosin Adewumi, Paul Rayson, Mofetoluwa Adeyemi, Gerald Muriuki, Emmanuel Anebi, Chiamaka Chukwuneke, Nkiruka Odu, Eric Peter Wairagala, Samuel Oyerinde, Clemencia Siro, Tobius Saul Bateesa, Temilola Oloyede, Yvonne Wambui, Victor Akinode, Deborah Nabagereka, Maurice Katusiime, Ayodele Awokoya, Mouhamadane MBOUP, Dibora Gebreyohannes, Henok Tilaye, Kelechi Nwaike, Degaga Wolde, Abdoulaye Faye, Blessing Sibanda, Orevaoghene Ahia, Bonaventure F. P. Dossou, Kelechi Ogueji, Thierno Ibrahima DIOP, Abdoulaye Diallo, Adewale Akinfaderin, Tendai Marengereke, Salomey Osei "MasakhaNER: Named Entity Recognition for African Languages" *The 2021 Conference on Empirical Methods in Natural language Processing (EMNLP2021)* Dominican Republic, November 7-11, 2021.

- Happy Buzaaba, Toshiyuki Amagasa, "A Scheme for Factoid Question Answering Over Knowledge Base," *The 11th Forum on Data Engineering and Information Management (DEIM2019)*, Nagasaki, Japan, March 4-6, 2019.