NEURAL QUESTION ANSWERING SYSTEMS: THE ROLES OF ATTENTION AND RECURRENT NEURAL NETWORKS

A THESIS SUBMITTED TO AUCKLAND UNIVERSITY OF TECHNOLOGY IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Supervisors

Prof. Edmund M-K Lai

Dr Mahsa Mohaghegh

18 March 2022

By

Yuanyuan Shen

School of Engineering, Computer and Mathematical Sciences

Abstract

The roles of attention and recurrent neural networks (RNN) in RNN-based neural question answering (QA) systems are investigated. As an important component of neural QA systems, attention provides a way for the most relevant words in the passage text that are relevant to the question to be identified so that a subsequent module can make use of this information to infer the answer. There are two main steps involved. The first one computes similarity scores between the words in the question and those in the passage. The second step generates the information that is relevant to the question for subsequent layers in the QA model. Many different attention similarity functions and relevant information generation methods have been used by various neural QA systems. It is important to understand the characteristics of the similarity functions and the relevant information generation approaches that perform well.

In order to make fair comparisons among similarity functions and among relevant information generation methods, a novel baseline QA model is designed. It captures all the major common characteristics of the leading RNN-based neural QA models. It is made up of four parts – the embedding layer, context encoder, attention mechanism, and answer predictor. In this way, the various similarity functions and relevant information generation methods could be easily plugged in.

Using this baseline model, eleven existing similarity score functions are

compared. Experimental results show that the group additive functions perform better than the multiplicative functions. Based on this insight, a new similarity function, called T-trilinear function, is proposed. It combines the strengths of both the additive and multiplicative functions, and it generally outperforms all the other existing functions.

Regarding relevant information generation, five existing methods are compared. Experimental results show that incorporating element-wise products into the information concatenation helps to achieve better results. A new method is proposed, which is able to produce better results than these five methods. Further investigation reveals that using an FNN over the concatenation can further improve the performance. This finding results in the second new method. Results show that it achieves better performances than the other methods.

The role of RNNs in the neural QA systems is investigated using a representative of such systems known as DMN+. Although DMN+ performs well on most of the 20 tasks in the bAbI dataset, it is not able to tackle those tasks that involve multi-step inductive reasoning effectively. Research results show that the RNNs in the attention mechanism memorize the order of facts in the training data. As a result, the trained model does not generalize well to the test samples with different orders of facts. This problem is overcome by developing a new QA model called MoDMN+ which has an RNN-free attention mechanism. Experimental results demonstrate that MoDMN+ has better generalization ability than DMN+. Considering the adverse effect of the RNNs in the attention mechanism on the multi-step induction tasks, a new QA model called ff-DMN is proposed by discarding the RNNs from the input model in the MoDMN+ model. Experiments show that ff-DMN can successfully solve the inductive reasoning tasks with a significantly higher predictive accuracy than DMN+ and the other existing RNN-based QA models. Furthermore, an ensemble model is proposed and can tackle all the 20 reasoning tasks in the bAbI dataset.

Contents

List of Tables

List of Figures

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.

_____

Signature of candidate

Publications

Shen, Y. & Lai, M-K E. & Mohaghegh, M. (2020), "The Role of RNNs for Contextual Representations: A Case Study Using DMN-plus," in *Proceedings of the 2020 4th International Symposium on Computer Science and Intelligent Control* (ISCSIC 2020), Association for Computing Machinery, New York, NY, USA, Article 14, pp.1–5. DOI: https://doi.org/10.1145/3440084.3441190

Shen, Y & Lai, M-K E. & Mohaghegh, M. (2020), "Role of RNNs for Non-sequential Tasks in The Question Answering Context," in *Proceedings of the 2020 4th International Symposium on Computer Science and Intelligent Control* (ISCSIC 2020), Association for Computing Machinery, New York, NY, USA, Article 40, pp.1–6. DOI: https://doi.org/10.1145/3440084.3441216

Shen, Y. & Lai, E.MK. & Mohaghegh, M. Effects of Similarity Score Functions in Attention Mechanisms on the Performance of Neural Question Answering Systems. Neural Processing Letters (2022). https://doi.org/10.1007/s11063-021-10730-4

Shen, Y. & Lai, M-K E. & Mohaghegh, M. (2021), "Relevant Information Generation for Attention Mechanisms in Neural Network based Question Answering Systems," submitted to *Applied Soft Computing* (under review)

Acknowledgements

There are so many people throughout my PhD journey that I owe my gratitude to. I'm very fortunate to have them along the way. Without them this journey would have been less colourful, less meaningful, and even too tough to go through. The first person my sincere appreciation goes to is my supervisor Prof. Edmund Lai. Studying PhD with Prof. Lai is one of the best things in my life. I thank him for giving me this precious opportunity. His continuous patience and dedicated guidance and support pave the way for me to grow and improve. I have learned not only how to review the literature, but also to draw patterns and insights from it. I have learned not only how to develop a method or a system and but also to perform critical analysis and evaluation on it. I've learned how to break a big project into small manageable ones and complete them. I've learned so many valuable things from Prof. Lai which cannot be fully expressed here. Whenever I feel stuck and turn to Prof. Lai for advice, his insightful words are always like a light shining into my mind and the tangle is cleared up.

I am also very grateful of my co-supervisor Dr Mahsa Mohaghegh. I thank Mahsa for connecting me with industry, which resulted in the funding for supporting my research. Her encouragement and advice along the way are very helpful. Her active role in the artificial intelligence (AI) community in Aotearoa New Zealand is my learning example. During the research, implementing and training deep learning models is an important part. I thank Bumjun Kim for being so supportive to help me solve the GPU problems whenever I needed him. I thank Dr Junior Nomani for being very considerate and caring when I worked as a Peer Mentor with him. I thank Jessica Yamamoto for the support from graduate research school. I thank all the staff not only in the School of Engineering, Computer and Mathematical Sciences, but also in the Faculty of Design & Creative Technology.

I sincerely appreciate Callaghan Innovation for the R&D Fellowship Grant

and Ambit for sponsoring this project. It not only helps me a lot financially as an international student, but also gives me a great opportunity to connect with industry, so that I can learn to view, communicate, analyse and carry out things from the business perspective. These skills can hardly be obtained by studying solely on a university campus.

There is a very special group of people that I would like to acknowledge and express my warmest gratitude to, who are my fellow PhD friends in MH206 at AUT South campus. Without their accompany along the way, I cannot imagine how different and less colourful this journey would be. My thanks go to those who have already completed the PhD and obtained the degrees: Shabnam, Ekta, Reem, Marjs and Isaac. For those who are still at this journey with me, thank you my sisters: Preet, Katharine, Soana, Jean, Litiuingi and Christianah. If I wrote down all our lovely stories in this journey, it would need another thesis-length document.

Behind the scenes my husband Shixin (Tom) and our two lovely children Zixuan (Angela) and Enxi (Esther) always keep me fueled up. Thank my husband for being so supportive and look after the kids well whenever I need extra time to do research. Without his support along the way, this PhD would not be possible to come to an end. Our two kids always play the important role of refreshing my mind from a long-day's research work. Seeing them grow up day by day makes me feel so proud of these years' efforts.

Finally, I thank my parents and my brother back in my home country who are always with me. We are deep in each other's heart and distance does not make any difference. Thank my Mum and Dad who brought me up and provided everything I needed so that I can become who I am today. Thank my brother who was my best buddy when we were young kids and is still my best friend who supports me constantly.

Chapter 1

Introduction

1.1 Background

In recent years, artificial intelligence has been experiencing explosive growth in its applications to various types of digital services that support human users [1]–[4]. One of these services relates to providing customer services by conversing with customers in the form of replying to their queries and responding to their comments. This can be carried out through conversational agents [5], [6] such as chatbots [7], [8] and digital (virtual) assistants [9], [10] using natural language [11], [12].

Currently the mainstream conversational agents rely on rule-based conversation (dialogue) flows created by dialogue engineers manually [13]. At the same time, in order to provide answers to queries, the information need to be organised in a structured manner. With the expanding volume of information, the process of extracting and organizing such data is costly and time-consuming [14]. This task will be even more arduous if we are dealing with free-form unstructured texts, such as regulatory or policy

documents. An automated approach in the form of a Question Answering (QA) system is needed.

The history of QA systems can be traced back to six decades ago. An example is the Baseball program [15] which was developed to extract the answer to a question from a structured database. The format and expression of the questions have to be highly constrained. Another well-known example is Eliza [16], a computer chatbot designed to respond to the statements a human user entered by turning it into a question. More capable QA systems emerged with advancements in linguistic analysis techniques. A system called Lunar was developed by the NASA Manned Spacecraft Centre to answer questions raised by a lunar geologist regarding lunar rocks collected by the Apollo Moon Missions in 1970s [17]. At around the same time, a system called REQUEST was developed by the IBM Watson Centre [18]. These QA systems require the questions to be put in a particular format and answers are retrieved from well-formed databases.

After these initial efforts, the focus has shifted to the machine learning approach. This approach allows the answers to be inferred from a set of features extracted from relevant natural language documents [19]–[21]. One of these QA systems applied a Naïve Bayes classifier to select the correct answer from a group of candidates by feeding a set of exclusively designed attributes based on the lexical information of the question, passage, and the candidate answer fragments [22]. Another system utilised two unsupervised machine learning techniques to compute the correlation between a query and the document sentences depending on a feature set that was extracted carefully [23]. These QA systems make the use of feature engineering to select a set of inputs for the algorithms. Additionally, the types of questions they were designed to tackle are limited to factoids and definitions.

More advanced QA systems typically combine natural language processing

(NLP) technology with artificial intelligence techniques to form more complex system architectures. With the availability of semantically labelled sources for training, these systems are able to achieve higher levels of capabilities. An example is the Watson system developed by IBM Research based on the DeepQA architecture. It competed on the TV quiz show Jeopardy! against former human winners in 2010 and achieve excellent performance [24], [25]. The DeepQA architecture consists of components for question processing, answer source processing, evidence source proceeding, and answer confidence ranking, etc. The confidence scores of these components are combined by a hierarchical machine learning method that decide whether to buzz in to answer a question. Although the Watson system performed well on the Jeopardy Challenge, designing its highly sophisticated architecture took three years of intense research and development by a team of about 20 researchers. This is mainly because it relied heavily on feature engineering.

Although a lot of progress have been made, none of the aforementioned QA systems is capable of learning knowledge from unstructured texts directly without using pattern-based rules or feature engineering. Artificial neural networks, in the form of deep learning, offer a way to learn directly from data. It has enabled a big leap forward in natural language processing. Deep learning techniques have also been used to build end-to-end QA systems.

1.2 Current Issues

The inputs to neural network-based QA systems (henceforth referred to as neural QA systems) are typically a “passage” consisting of a number of sentences or statements, and a question. The output is the predicted answer which may be a single word or

a short phrase. These systems are trained in a supervised manner, meaning that the training dataset consists of inputs together with their corresponding correct outputs. Performance is then measured using previously unseen test data.

There are two main challenges for such neural QA systems. The first one relates to the fact that the passage from which the answer is to be obtained can be quite long. Therefore, looking for the right answer to the question can be difficult. A technique that has been used effectively in many existing neural QA systems is to make use of an attention mechanism. The purpose of the attention mechanism is to allow the words in the question to be associated with those in the passage to generate information that is relevant to the question. It can also be interpreted as a way by which the words in the question pay attention to different parts of the passage that are considered most relevant. This information could then be used in subsequent modules to infer the answer. This approach has been used successfully in many neural QA systems [26]–[29].

An attention mechanism consists of two steps. The first one computes an attention similarity score between various parts of the passage and the words in the question. Many different similarity functions have been used by different neural QA systems. It is not clear which similarity functions perform better than others.

The second step in an attention mechanism is the way by which information relevant to the question is generated for subsequent layers in the neural QA network. Two models, BiDAF and DCN, are among the first to propose methods for the relevant information generation [27], [28]. It involves generating a question summary and a passage summary. The relevant information is the integration of these summaries along with other information. Different QA systems utilise different methods to perform this integration. The relative effectiveness of these approaches is not currently known.

Although the recently emerged Transformer-based QA systems also contain

the attention mechanism called self-attention, they are not the focus in this thesis. Because in this type of QA systems self-attention is distributed and each layer contains this attention. There are a number of such layers in a Transformer-based model. This will result in a considerable amount of combinations of testing a method in self-attention. For example, which layers are to be tested and which remain the same. These are a lot more complex to investigate and require much longer time to undertake the investigation. It is not feasible to fit into the time frame of the thesis. Therefore, the RNN-based QA systems are chosen to be the focus.

The second challenge for neural QA systems is that obtaining the answer to a question may involve reasoning across multiple statements in the passage. This type of reasoning is usually referred to as multi-hop reasoning. There are three main types of reasoning – sequential, deductive, and inductive. All three types of reasoning are present in the bAbI dataset that is created by Facebook and is very extensively used by researchers in QA systems [30]. Current QA systems are able to achieve accuracies above 90% for the sequential and deductive reasoning tasks [20], [21], [31], [32]. However, for the inductive tasks, the accuracies of these same systems are only around 50%. QA systems that can perform all three types of reasoning equally well will be important for practical applications.

1.3 Research Aims and Objectives

The main aim of this research is to study the issues related to attention mechanisms and inductive reasoning for Recurrent Neural Network (RNN) based neural QA systems. The goal is to gain deeper understanding of these issues so that better systems could be designed. Five objectives of this research have been identified.

The study of the attention mechanism is broken down into two parts. The first part relates to the choice of attention similarity functions and the second is concerned with relevant information generation methods. Since such similarity functions and information generation methods are closely tied to the neural network architecture of the associated QA systems, it is difficult to make a fair comparison of their contributions to the performances of these systems. An approach to overcome this problem is to construct a model that captures the common characteristics of these QA models and allows the attention mechanism to be isolated as a separate module. Thus, the ***first objective*** is to develop a baseline model that captures the common characteristics of various RNN-based neural QA systems.

With an appropriate baseline model, comparisons between the performance of various similarity functions found in the literature can be made. This comparison will provide insight into the type of similarity functions that works well. The ***second objective*** is to make use of the insight obtained from this comparison to develop a new and better similarity function.

Different QA architectures aggregate different information to produce the relevant information. The effects of different approaches to generate the relevant information could be compared using the baseline model. The ***third objective*** is therefore to propose a better relevant information generation method using this new understanding.

The final two objectives relate to the unsatisfactory performances of existing RNN-based QA systems on the multi-hop inductive reasoning task. The architectures of such systems typically include both feedforward neural networks (FNNs) and recurrent neural networks (RNNs). RNNs are used for two different purposes. Firstly, they help to generate contextual representations in the attention mechanism. The ***fourth objective***

is to determine whether the RNNs in the attention mechanism adversely affect the performance of the model in the inductive task. If so, then a better alternative is to be proposed. The second purpose of RNNs is to enable the sequential dependency information of the words from the input be captured. The ***fifth objective*** is to determine how such RNNs may affect the performance of the model and to propose new alternatives.

1.4 Original Contributions

Corresponding to the five objectives listed above, the original research contributions of this thesis can be summarized as follows.

1. A novel baseline QA model is proposed that captures the common characteristics of the neural QA models with modular attention mechanisms. With this model a fair comparison among different methods in the second and third objectives can be conducted. This approach to investigating neural network architectures for QA systems has never been attempted before.

2. Different similarity functions are compared for the attention mechanism using the baseline model. Experimental results demonstrate that the additive similarity functions perform better than the multiplicative ones. A new similarity function T-trilinear is proposed by combining the strengths from both the additive and multiplicative groups, and it outperforms the other functions.

3. It has been found that incorporating element-wise products into the information concatenation helps to achieve better results. Furthermore, processing this concatenation through an FNN can further improve the performance. A new method for relevant information generation is proposed based on these findings. It is able

achieve better performances than all the existing methods.

4. The RNNs in the attention mechanism, known as attnGRU, have been found to memorize the sequence of facts in the training dataset. As a result, if the order of facts in the test data is not present in the training data, then the system performs poorly. A new model called MoDMN+ is proposed with an RNN-free attention mechanism. This model has better generalization capabilities in this respect than the DMN+ system.

5. A new QA model called ff-DMN is proposed with an RNN-free input module, building on MoDMN+. It is the first model that is able to solve the inductive reasoning task with a high predictive accuracy. An ensemble model that makes up of a conventional model and ff-DMN that can tackle all the 20 tasks in the bAbI dataset successfully is constructed.

1.5 Thesis Outline

The rest of the thesis is organised as follows.

Chapter 2 reviews the developments of QA systems since they were first introduced around 60 decades ago. After introducing the early rule-based and feature engineering based systems, an overview of neural network based systems, including RNN-based and Transformer-based models, is given. In addition, two important parts of a neural QA systems – word embeddings and attention mechanisms, are discussed in more detail. An extensive review of the characteristics of various datasets that are available for experimentation with QA systems is also conducted.

Chapter 3 describes the design of the baseline model that captures the common

characteristics of the neural QA models with modular attention mechanisms. It is instrumental to the study conducted in the Chapters 4 and 5.

The focus of Chapter 4 is on the effects of similarity score functions on the performance of the QA system. A new function is proposed which is shown to perform better than other functions used in the literature. Chapter 5 continues to study the effects of relevant information generation methods. A new method which is able to achieve the higher predictive scores than existing methods is proposed.

Chapter 6 investigates the role of RNNs in the memory-augmented QA networks for the non-sequential tasks. An RNN-free attention mechanism is proposed as a replacement of the RNN-based attention mechanism in the DMN+ model. The resulting model, MoDMN+, is able to resolve the fact order preserving phenomenon caused by the RNN-based attention mechanism and improve the generalization ability of the model. Furthermore, the RNN in the input module of MoDMN+ is replaced by an FNN which allows the resulting ff-DMN model to solve the inductive reasoning task with 97.2% predictive accuracy, far above what can be achieved by other QA systems. Finally, an ensemble model combining the powers of DMN+ and ff-DMN, allows all the 20 tasks in the bAbI dataset to be tackled with high accuracy.

Chapter 7 concludes this thesis with a summary of its contributions and discusses their significance. Several directions of further research are also suggested.

Chapter 2

Literature Review

Question answering (QA) systems have come a long way since its inception around 60 years ago. More recently, with advances in natural language processing (NLP) techniques, deep learning and the computational capabilities of computer systems, research in QA systems have experienced exponential growth. In this chapter, a review of QA systems and models is presented. After briefly describing the early approaches based on using rules and feature engineering, in Sections 2.1.1 and 2.1.2, the focus is turned to neural network based QA models. There are two main types of neural QA models. The first makes extensive use of recurrent neural networks. The second type makes use of feedforward networks only and has architectures that are variants of a deep network known as the Transformer. These characteristics and performances are reviewed in Sections 2.2.2 and 2.2.4 respectively. Furthermore, some of the main techniques used by neural NLP systems are also reviewed. They include word embeddings in Section 2.2.1, recurrent neural networks in Section 2.2.2, and attention mechanisms in Section 2.2.3. The main datasets used by researchers in QA system are described in Section 2.3.

2.1 Early QA Systems

2.1.1 Rule-based QA Systems

The history of QA systems traces back to 1960s. Surveys of these early QA systems were published in 1965 [33] and 1970 [34]. These systems were designed based on structured databases that were created with certain rules, such as lists and graphs with specified entities. Queries were restricted to certain formats or templates in order the answers could be found in the databases.

One of these rule-based QA systems is the Baseball program developed in 1961 [15]. The Baseball system was designed to answer English questions about the scores, teams, locations and dates of baseball games. It used list structures to organise the data, which summarized a Major League's season's experience. A snapshot of this database is provided in Figure 2.1. The questions are restricted to single clauses without logical connections and excluding a certain relations. Additionally, the Baseball program contained a dictionary that helped to convert questions into a specification list similar in format to the data structure [33]. The other list-structured database based QA programs, like SADSAM [35] and DEACON [36], [37], developed at this stage used similar techniques as Baseball by converting sentences into lists of entities or directly constructing a list-structured databases.

Due to the advent of computer network and the growth of computer facilities and linguistic techniques, the explorations in QA systems ventured into realistic applications from the experimental phase. It provided opportunities for researchers to access a variety of computer facilities and databases located remotely by providing answers to their questions. A representative of such practical QA systems is Lunar, which was developed by the NASA Manned Spacecraft Centre to answer questions

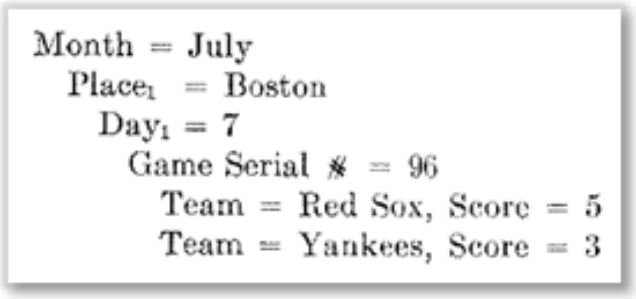


Figure 2.1: A snapshot of the Baseball program's list-structured database [33]

raised by a lunar geologist regarding lunar rocks collected by the Apollo Moon Missions in 1977 [17], [38]. The database is a table that contains 13,000 entries regarding the analyse information of the Apollo 11 samples. Queries were translated into a symbolic expression by a formal query language in order to be executed on the database. Another QA system is REQUEST developed by IBM in 1976 [18], which was the precursor of the famous IBM Watson system [24]. REQUEST required users to ask questions in a formal query language, so that it was able to obtain answers from a small formatted Fortune-500-type database.

1980s and 1990s experience the significant growth of linguistic techniques and lexical resources, as well as their empowerment in QA systems, such as the development of TELI [39], QUEST [40] and PARIS [41]. TELI was created to deal with comparatives by transforming natural language sentences to executable expressions. QUEST was designed to be a psychological model of QA, and produce answers that adults typically produce. It depends heavily on an organization of knowledge structures and a series of QA procedures that operated on the structure. PARIS extracts answers from a large knowledge base structured around WordNet [42] based on its inference rules implemented as chains of selected relations [41].

All these QA systems rely on highly structured databases build on a variety

of linguistic resources for retrieving the answer from. The prime limitation of these systems is that the knowledge is stored in structured databases, which allow them to be only capable of answering questions within the restricted scope of information. Another limitation is that the types of question expressions are strictly constrained in order to be executed by the query transformation rules.

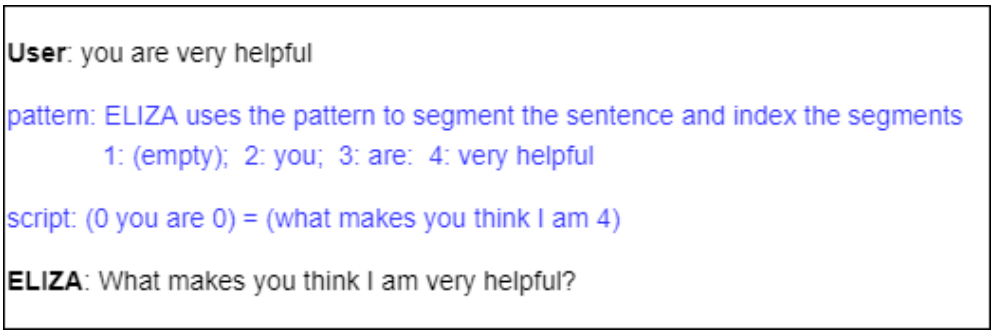


Figure 2.2: A snapshot of the conversation between ELIZA and a user

Another type of rule-based QA systems is based on pattern operation rules. One of these systems is ELIZA developed by Weizenbaum in 1966 [16]. ELIZA was created to simulate a Rogerian psychotherapist to converse with patients in natural language. The responses it produced to the statements entered by human users were based on a set of pattern transformations defined by a prepared script. The pattern consists of keywords and the operations are the substitution of an English text segment in conjunction with some portion of the user's input sentence. A snapshot of the conversion between a user and ELIZA is provided in Figure 2.2.

Following the advent of ELIZA, the other QA systems were developed, such as Parry and Alice. Parry was developed by Stanford University as a paranoid patient in 1970s [43]–[45], which was written in a high-level programming MLISP and was limited to linguistic communication by means of teletyped messages. Alice was developed in 2001 [46] using Artificial Intelligence Markup Language (AIML) to converse with human users with text-based rules.

These early pattern-oriented rule based QA programs, acting as conversational agents, are simply stimulus-response models, which recognise something in the input and then just respond to it without inferring [47]. Their rules created ways for them to converse with users, but also constrained them to a very limited scope of discourse.

2.1.2 Feature Engineering based QA Systems

Feature engineering based QA systems are characterized by the extraction of features from natural language questions and answer sources. They reply on a variety of linguistically-based NLP techniques, such as part-of-speech tagging [48], syntactic parsing [49], and name entity recognition [50]. These QA systems retrieve relevant passages to a question from document collections, extract candidate answers, and select the top-ranked one [51], [52]. In general, they are made up of three main components as shown in Figure 2.3, which are question processing, document processing, and answer processing [53].

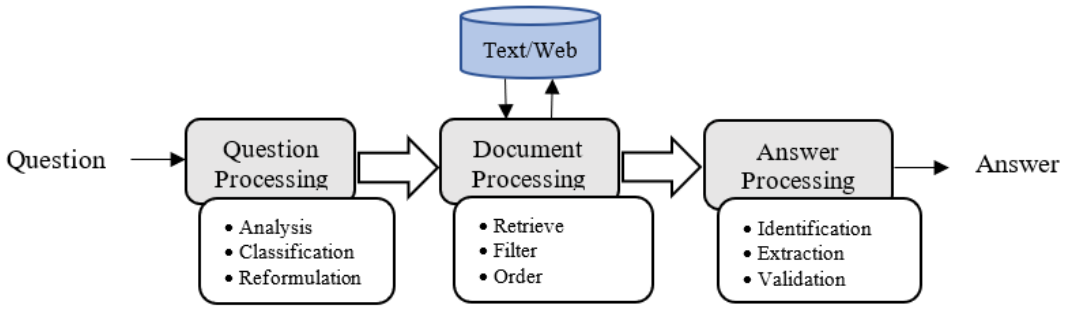


Figure 2.3: A general block diagram of feature engineering based QA systems

The goal of the question processing component is to classify a question into the taxonomy of answer types based on the features extracted from the question. Some early work undertook the classification through hand-craft rules or heuristics. This type

of work usually used a shallow level of NLP techniques to analyse the surface form of a question, and adopted the rules to assign the question to the corresponding type of answers [54]–[58].

The document processing component first retrieves relevant documents to the question, and then filters the retrieved documents and order the candidate paragraphs based on manually defined features, such as same word sequence score, distant score, and missing word score [59]. The answer processing component identifies the answer candidates within the filtered ordered paragraphs through parsing, extracts the answer by choosing the word or phrase and validates the answer by providing the confidence in its correctness [56]. The performance of the whole system is contingent on the features generated in each component.

With the advances in artificial intelligence, QA systems started to adopt machine learning techniques. One of the examples is in [60] which focused on improving the accuracy of question classification by applying a machine learning based parsing. Pasca and Harabagiu [61] fed a perceptron with seven manually created features to rank the answer candidates. Juárez-González, et al. [22] trained a Naïve Bayes classifier to select the answer from a set of candidates. Chali and et al [23] adopted two unsupervised machine learning techniques – K-means and Expectation Maximization, to compute the relative importance of sentences. In these QA systems, machine learning was used in one of the components of the whole system, and linguistic features are crucial for the performance.

Some advanced QA systems added more sophistication than that shown in Figure 2.3. A representative archetype is the IBM Watson [24], which competed on the TV quiz show Jeopardy! against former human winners in 2011 [24]. Its architecture is demonstrated in Figure 2.4. All its components post features of computation and the

associated confidence levels, and they are combined by a hierarchical machine learning method that decide whether to buzz in to answer questions in the open domain. Watson cognitive computing system has been serving as a platform for many applications in a variety of areas [62], [63]. However, it still basically relies on feature engineering and various external semantic resources. This requires considerable effort and time when adapting it to new domains, particularly for the languages with very few such resources available.

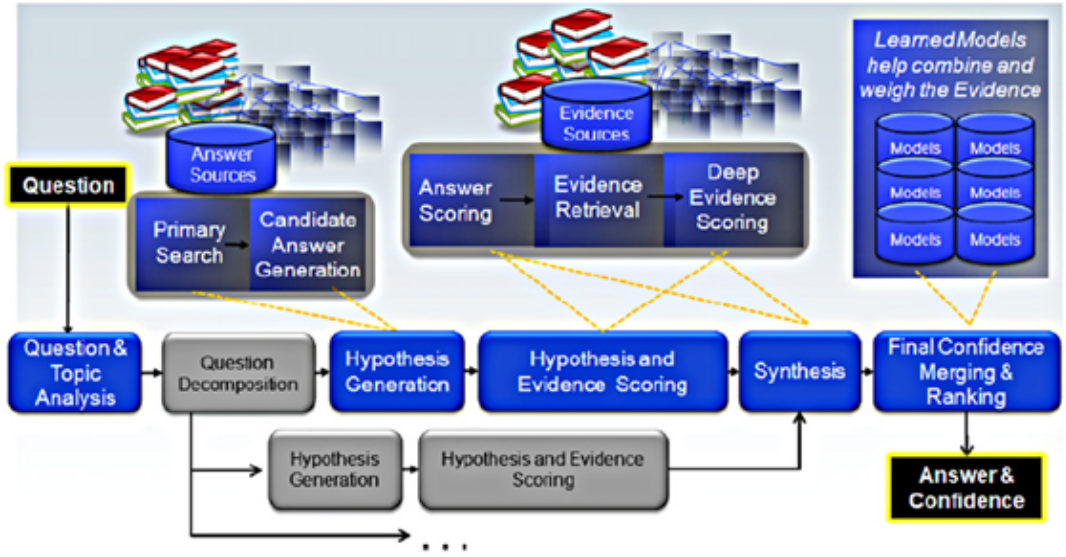


Figure 2.4: The DeepQA architecture [25]

The main limitation of these early QA systems is that they were designed to perform on specific datasets according to the different attributes of the datasets. This means that it is difficult to compare the performance of one system with another, as there is no common test data that applies to all of them. More importantly, they are not data-driven, which means that they cannot automatically learn from data.

2.2 Neural QA Systems

Deep learning is a recent development in artificial neural networks (ANN) [64]–[66]. It provides an alternative approach to build QA systems. Instead of having to manually define the features to be extracted, deep learning can learn useful representations at different scales automatically when presented with inputs and outputs [66]–[68]. Learning is conducted typically in a supervised fashion [69], [70]. Given labelled training data, deep learning can learn useful representations for predicting the downstream tasks. Some deep learning techniques are designed for unsupervised learning [71], [72]. The training algorithms enable these deep learning models to learn features and patterns that can provide better insights for understanding the data.

In the past decade, various types of neural network architectures in deep learning, such as deep feedforward neural networks (FNNs) [71], [73], convolutional neural networks (CNNs) [74], [75] and recurrent neural networks (RNNs) [76]–[78], have been shown to perform unexpectedly well in a number of traditionally considered difficult areas [79]–[83]. These areas include computer vision [84], [85], speech recognition [81] and natural language processing (NLP) [86], [87]. Figures 2.5, 2.6 and 2.7 illustrate the architectures of these three main types of neural networks.

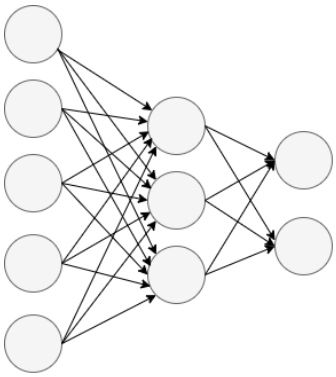


Figure 2.5: An example of a three layer FNN

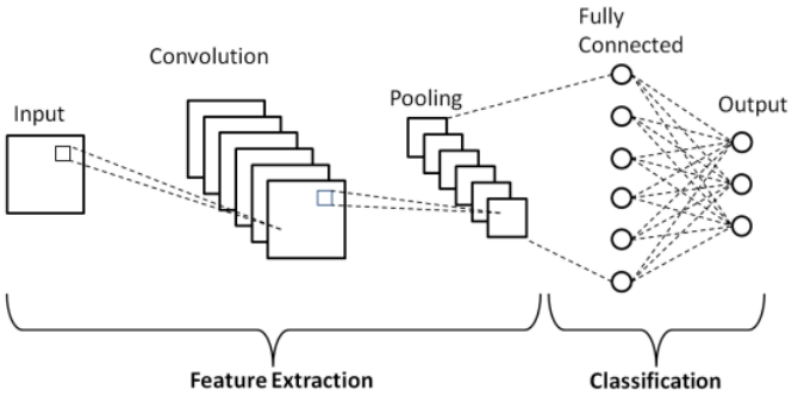


Figure 2.6: An example of a CNN [88]

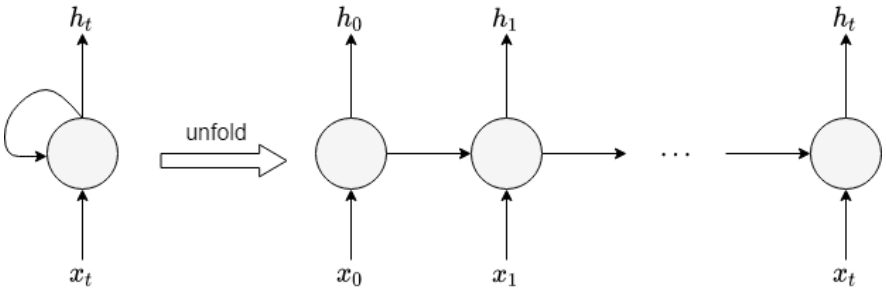


Figure 2.7: The illustration of an RNN

Deep neural networks have provided the most significant advances in NLP including the development of QA systems. These systems can be categorised into two main groups according to the type of neural network architectures that they used. The first group of systems makes use of RNNs to capture dependencies of words that are far apart. The second group is based on an architecture known as the Transformer that has a feedforward only architecture. RNN and Transformer based QA systems will be reviewed in Sections 2.2.2 and 2.2.4 respectively. Since the subject of this thesis is on RNN-based QA systems, they will be discussed in more detail.

In order to process natural languages which are symbols, they must be converted into numbers so that neural networks can process them. A basic technique is to represent each word as a numerical vectors known as word embedding. Techniques to

compute word embeddings will first be reviewed.

2.2.1 Word Embeddings

Word embeddings are the learned representations of words, in the form of numerical vectors. Ideally, these vectors should be close to each if the meanings of the words that they represent are similar. In order to provide sufficient space to encode a large number of words, these vectors are typically high-dimensional. They are obtained through embedding models that are trained on large corpora. The two main categories of models are described in the following subsections.

2.2.1.1 Matrix Factorization-based Methods

Word representations obtained by matrix factorization methods utilize low-rank approximation to decompose large matrices that capture statistical information on a corpus. The history of these methods can be traced back to 1990 when latent semantic analysis (LSA) was introduced [89]. With LSA, a "term-document" matrix which describes the occurrences of terms in documents. Each row of this matrix correspond to a term found in the documents and each column correspond to a document. The dimension of this sparse, high-dimensional matrix is then reduced using singular value decomposition (SVD) while preserving the similarity structure in the columns.

An alternative strategy was introduced in 1996. It is known as Hyperspace Analogue to Language (HAL) [90]. With this method, a word is represented as a vector of other words co-occurring with it. A co-occurrence matrix is formed by sliding a window consisting of several words over the source corpus and summing the co-occurrence counts for each word pair as the cells of the matrix. Dimension reduction

is achieved by retaining a relatively small number of principal components of the co-occurrence matrix. As a result, words co-occurring have similar row structures. HAL paved the way for the subsequent co-occurrence matrix based word vector approaches.

A main problem with HAL is that the most frequent but unimportant words, such as *the* and *a*, can cause high variance in their respective columns. This means that the information of these columns tend to be retained while those of other important but less frequently-occurring words tend to be eliminated through dimensionality reduction. Several techniques have been proposed to tackle this problem. One of them is COALS [91] where the co-occurrence matrix is first normalized by an entropy or correlation-based method. A merit of this approach is that the raw co-occurrence matrix is compressed so as to be distributed more evenly in a smaller interval. Other subsequent methods also pursue this direction of improvement. For example, Bullinaria and Levy [92] verify that positive pointwise mutual information (PPMI) is a good normalization method. It is worth noting that they also showed that the cosine distance measure is the best way to calculate word similarity along with the PPMI components. In 2014, Lebret and Collobert proposed Hellinger PCA (HPCA) [93] by making a square root type transformation called Hellinger distance, which was suggested as a relatively simple but still effective way among the word embedding methods.

More recently a GloVe vector model has been introduced [94], which not only makes use of the statistics of word occurrences in a corpus as the aforementioned methods do, but also captures the global corpus statistics. After constructing the matrix of word-word occurrence counts, the co-occurrence probabilities are extracted, which indicate how often one word appears in the context of another. Then the ratio of these obtained probabilities is calculated, which was verified to provide better discrimination between two relevant words. The pre-trained GloVe vectors with 50d, 100d, 200d, and 300d have been released publicly by the authors, which were trained on a large corpus

(Wikipedia 2014 + Gigaword 5). These pre-trained word vectors have become one of the most prevalent options for a number of subsequent NLP tasks to represent words [20], [95]–[98].

2.2.1.2 Neural Network-based Methods

Word representations learned by neural networks are usually called distributed word embeddings, wherein words are embedded into a vector space. In the beginning, words were represented as one-hot vectors, which have the same size of the vocabulary and were assigned 1 to the position of a word and remain all other elements as 0. The one-hot vectors require a lot of memory and cause the severe data sparsity problem. Converting them into a dense latent feature with neural networks was a main concern of the earlier NLP researchers.

One of the earliest work of distributed representation learning, which is later applied to word embeddings, dates back to 1986 published in Nature by Hinton et al. [99]. This idea was applied to a language model that learns word vector representations by Bengio in 2003 [100]. In 2008, Collobert & Weston brought the concept of *word embeddings* that each word is embedded into a fixed dimensional space [101], and since then *word embeddings* are commonly referred to word vectors obtained by any methods. In this word embeddings approach, the training of word representations is connected with the downstream multi-tasks learning in a unified deep neural network (DNN). By training this unified DNN with labeled data in a supervised way, word embeddings are obtained, which can optimize the performance of the system. This method not only establishes word embeddings as a useful tool for NLP, but also introduces neural networks as an important approach for many subsequent NLP techniques.

Despite the abovementioned Collobert & Weston's work plays a precursor

role in neural word representations, neural networks became eventually popular in word representations a few years later when Mikolov et al. presented two different neural network models for learning word embeddings in 2013 [102]. These two models are continuous bag-of-words (CBOW) and continuous skip-gram. The main difference between these two models and Collobert & Weston's work is that they do not require labeled data that are expensive to access. They can be trained on large corpora in an unsupervised fashion to obtain distributed word representations. The CBOW architecture predicts the centered word with the surrounding words, and the skip-gram architecture predicts the contextual words given the current word. The authors also made 300-dimensional pre-trained word2vec embeddings publicly available by training the CBOW architecture on a 100-billion-word corpus from Google News. The word2vec has been leveraged by many succeeding publications [103]–[106].

2.2.2 RNN-based QA Systems

In the area of NLP, recurrent neural networks (RNNs) have been widely used to model the dependencies of text which is essentially sequential in nature [86], [107]. RNNs have the capability of modelling semantic and syntactic relationships over words, sentences, or even larger chunks of texts.

A recurrent neural network (RNN) consist of layers of neurons that have connections not only with the succeeding layer but also with its own layer [108] as shown in Figure 2.7. This unique structure allows them to model sequences [109]. In principle a large enough vanilla RNN should be sufficient to learn sequences of arbitrary length. In practice, however, they are unable to learn the relationship between items in a sequence that are substantially far apart. This is because of a numerical problem known as the vanishing/exploding gradient problem during the training stage [110].

Some variations of RNNs have been developed to tackle this issue, such as Long Short-term Memory (LSTM) [111] and Gated Recurrent Unit (GRU) [112]. They have specially designed gates that makes them capable of having a longer memory without having the vanishing/exploding gradient problem. An LSTM has a forget gate, an input gate and an output gate to help decide what information in a sequence to keep and what to discard in each cell. A GRU has a reset gate and a update gate which play a similar role in regulating the flow of information. They have been successfully applied to a number of applications, such as language modeling [113], [114], neural machine translation [115], [116], sentiment analysis [87], [117], and question answering [20], [118].

The structure of a standard RNNs can only relate the current to the preceding context in a sequence; they are unable relate the current with the succeeding context. For instance, consider the following three sentences:

> "Yesterday Mary went to school in the morning. She walked to the bookstore after school. After that she took a bus to the cinema."

With standard RNNs, the representation of the middle sentence can only contain the information of the preceding sentence, but not the succeeding one. In order to let the representation capture the context in both directions, bi-directional RNNs was proposed [78]. They have been widely used in the forms of bidirectional LSTM and GRU [119]–[122].

Many question answering models also consist of both standard and bidirectional RNNs. Some utilise them to capture the sequential dependencies among words [27], [28] while others use them to capture dependencies among sentences [20], [32]. In this thesis, they are referred to as RNN-based QA systems.

RNN-based QA systems generally derive the answer to a question in the following way. Words in the passage and question are first converted into word embeddings as described in Section 2.2.1. Then RNNs are utilised to capture temporal dependencies between words, phrases or sentences. Next, the parts of the passage that are most relevant to the question is focused on through the use of an attention mechanism. Finally, the answer is inferred from the relevant information generated by this mechanism.

Different systems use different ways to implement these steps. Some models make use of pre-trained word embeddings like GloVe [20], [27], while [32] uses random embeddings. Yet others incorporate linguistic features [26], [123], [124]. For the second step, some QA models use standard RNNs to capture the contextual information in forward direction [20], [28], while some adopt bi-directional RNNs [27], [32].

As for the attention mechanism, a variety of different methods have been used in the literature as well. Some attention models contain only one hop of the computation which is applicable if the answer could be found directly from the passage [26]–[28]. Some others have multiple hops so that they are able to perform multi-step reasoning [20], [32]. Even for those with only one computational hop, different methods for calculating the attention similarity score are used. For instance, the BiDAF model uses a trilinear function, while the DCN model makes use of a dot-product function. Attention mechanisms are reviewed in more detail in Section 2.2.3. As for answers, some system produces single-word answers [20], [21], [32], [125]. Other systems are able to represent the answer as a span or range of word indices in the passage [26], [27], [29], [124].

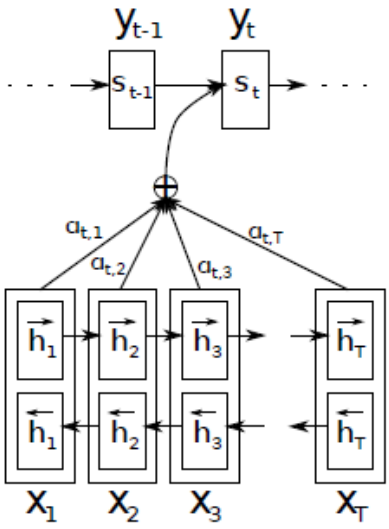


Figure 2.8: The attention mechanism in the RNNsearch model

2.2.3 Attention Mechanisms

Attention mechanisms have become an increasingly influential technique in deep learning, and they have been applied to many application areas in recent years [126]–[130]. In NLP, handling long sequences of words is a crucial and challenging task. The traditional way is to compress long sequences into short fix-length vectors [112], [131]. However, this can potentially cause a loss of information, especially in the earlier stages of the processing pipeline [115]. Attention mechanisms were proposed to circumvent this problem by focusing on the information that is most relevant to the target.

One of the earliest work in NLP that introduced an attention mechanism is the RNNsearch model which was developed for neural machine translation [115]. In this model, the attention mechanism allows the decoder to determine which parts of the source sentence to pay attention to at each time step. This is illustrated in Figure 2.8. Technically, it involves two step –

1. Calculate the matching scores between the target word in the previous time step and each word in the source sentence;

2. Generate the word in the current time step by taking the sum of all the source words weighted by their respective probabilities.

Another early work that makes use of attention is in [132]. Here, the authors used different functions for calculating matching scores.

Although both of these attention mechanisms are used in machine translation, their success has inspired its use in neural QA models [133]–[135]. The attention mechanisms used in QA models can be categorised into two types. The first type is known as modular attention. It is used in RNN-based QA systems. The second type is known as self-attention. It is an integral part of the Transformer model which will be reviewed in Section 2.2.4.

The differences in the way these two types of attention works is closely related to the architectures of RNN-based and Transformer-based QA models. The question and the passage text are separated in RNN-based systems. Therefore which part of the passage is paid attention to depends on the words found in the question. In other words, attention score is computed between these two entities. As a result, the attention mechanism forms a distinct module within the QA model architecture. On the other hand, in Transformer-based models, the question and the passage are concatenated as a single input sequence. Consequently, one cannot distinguish between the two and hence attention must be computed between all the words in this input sequence. Hence it is called self-attention.

There are a number of different ways modular attention is implemented. A simple but popular way is to use matrix multiplication operations. This can be found

in [26]–[29]. Another way is to use RNNs to produce relevant vectors [20], [32], [136], [137]. This is referred to as gated attention, as the attention weights are used to replace a certain gate in a gated RNN such as GRU. This method can be found in [20], [32]. A more detailed review of the operations involved in modular attention can be found in Sections 4.1 and 5.1.

2.2.4 Transformer-based QA Systems

The Transformer model was first proposed in [133] for natural language processing. Its structure consists of a stack of encoders and a stack of decoders as shown in Figure 2.9. These encoders and decoders are feedforward networks. Hence, a Transformer does not use RNNs to capture sequential dependencies in a sequence. Instead, it relies on a position encoder to preserve the information of the position of each word within the text. The encoders and decoders learn the contextualized representations of each word in the sequence and map the representation of the original sequence to that of the target sequence.

Figure 2.10 shows in more detail the structures of the sub-layers within each encoder and decoder layer. Self-attention is a sub-layer of each of the two different types of layers. The only difference between the decoder and the encoder is that the former predicts the current word based on the preceding words in the sequence (unidirectional) and the latter uses both the preceding and succeeding words (bi-directional).

Transformer has been shown to perform very well in natural language tasks. Many variants based on this architecture have appeared in the literature. GPT [138], developed by OpenAI, is among the first of these variants to utilise semi-supervised learning. It was trained with unlabelled data from over seven thousand unpublished

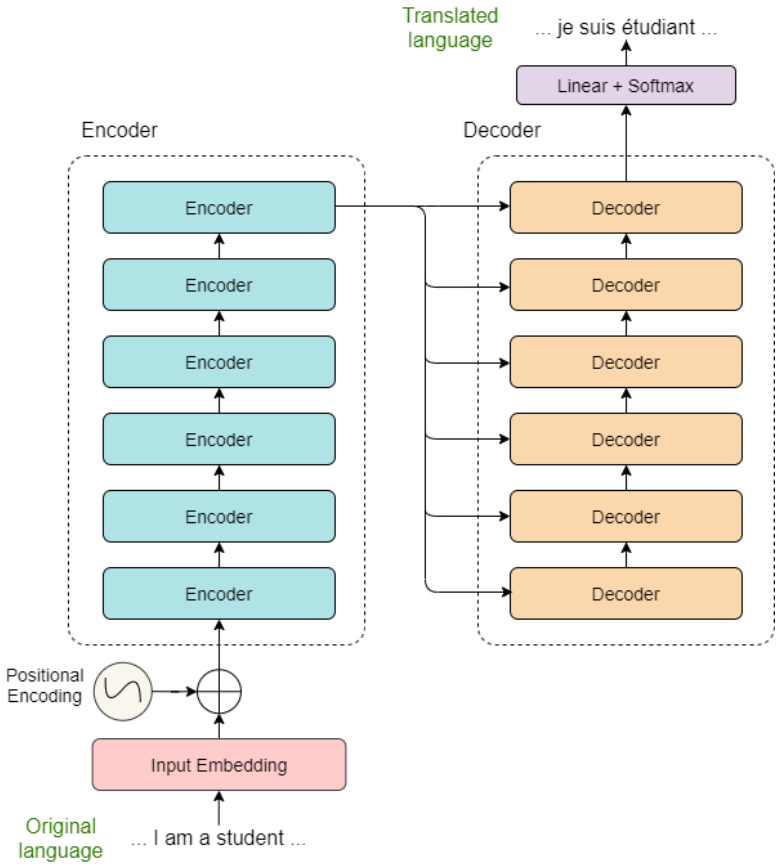


Figure 2.9: A block diagram of the Transformer model

books. Then the trained model was fine-tune with labelled NLP labelled data in a supervised fashion. GPT has 117M parameters and only consists of decoders. Thus it only uses a unidirectional context for the current word. Another variant known as BERT was proposed by Google. It has a full encoder-decoder structure to make use bi-directional contextual learning [139].

GPT-2 [140] has 1.5 billion parameters, ten times more than its predecessor GPT. It was trained with ten times more data than GPT as well. It exhibits the impressive ability of writing coherent essays better than any computer is able to produce. GPT-2 showed that having more parameters and trained on larger datasets help to improve the capability of such deep learning networks for many NLP tasks. Thus its successor, the

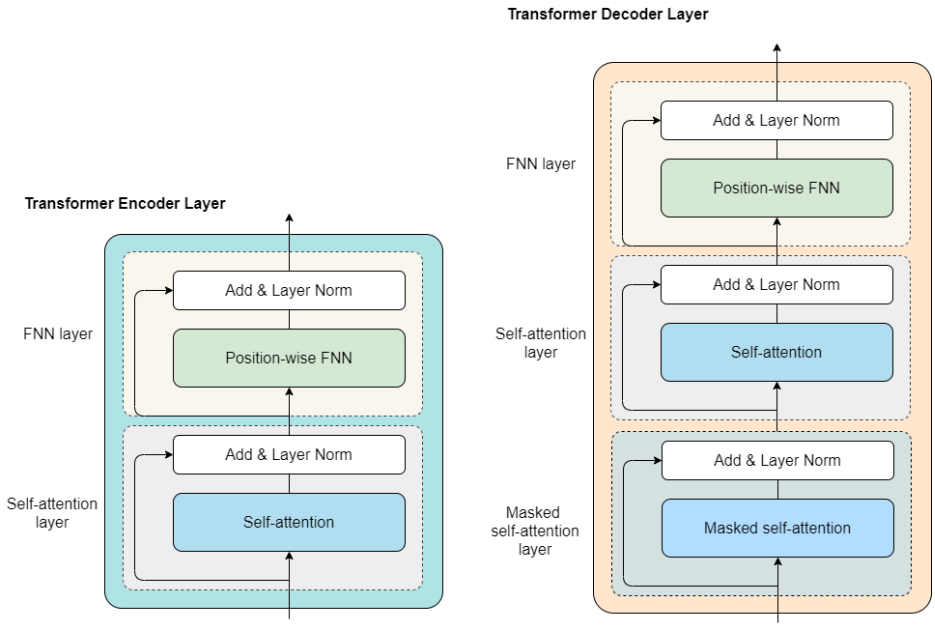


Figure 2.10: The sub-layers in the encoder and decoder of the Transformer model

GPT-3 [141], has ten times more parameters at 175 billion. It performs very well on downstream NLP tasks in zero-shot and few-shot settings. It is difficult to distinguish the articles written by GPT-3 from those written by humans. Other large-scale models have emerged. One of them is the Switch Transformer with a trillion parameters, but without increasing computational costs [142]. DALL-E [143] is a Text2Image system trained on text-image pairs. Wu Dao 2.0 has both Chinese and English language generation skills [144].

A drawback of the Transformer model is that the length of text dependency is fixed at design time. In order to enhance its ability to learn longer-term dependencies, recurrent structures are re-introduced in the Transformer-XL model [145].

Transformer-based models are first pre-trained on multiple large-scale language corpora, such as BooksCorpus dataset [146] containing 7,000 unpublished books,

Common Crawl dataset [1] and English Wikipedia [2]. Then these pre-trained models are deployed for specific downstream NLP tasks, including question answering. However, the architecture of these Transformer-based models is such that the input is a single sequence of text. Hence there is no distinction between the question and the passage. Therefore, the question and the passage are concatenated into a single sequence. This applies to BERT, GPT-3, XLNet [147] and T5 [148]. Thus the most significant difference with RNN-based QA models is that the passage and the question are not separated but are treated as a single concatenated sequence of words. Furthermore, self-attention is computed in each encoder layer rather than computed in one single part of the model. This makes self-attention a distributed and integral part of each layer which cannot be separated.

2.3 QA Datasets

Publicly available datasets are very important for the research in data-driven algorithms and models. They allow researchers to compare the capabilities of different systems. A number of datasets can be used for QA systems. They are categorised and reviewed below.

2.3.1 Datasets with Text-span Answers

The most important characteristic of this group of datasets is that the answers to questions may span multiple words in the passages. The answers are therefore represented by indices of the start and end (the span) words in the associated passages. The goal of

[1] https://commoncrawl.org/the-data/
[2] https://en.wikipedia.org/wiki/English_Wikipedia

a QA system is to extract these spans, referred as span-extraction [149]. An example is shown in Figure 2.11 where the answer is displayed in blue. The length of the spans for different answers will typically be different.

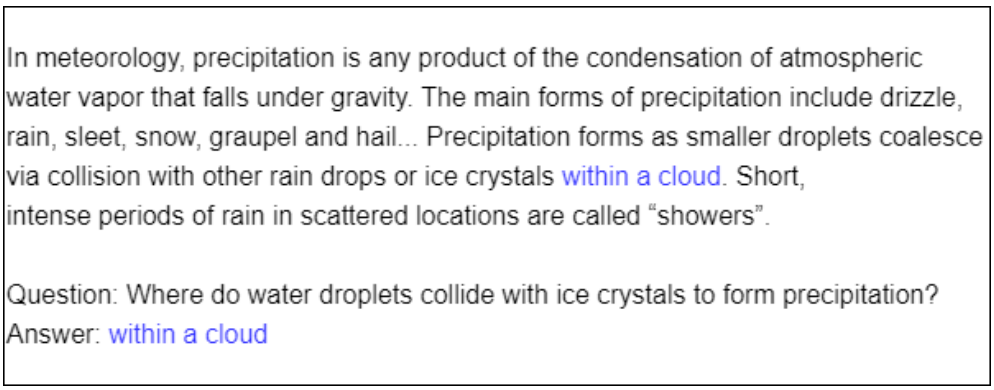


Figure 2.11: Example of answer span extraction task

One of the most popular datasets of this type is SQuAD [149]. It consists of 107,785 question-answer pairs on 536 articles selected from Wikipedia. It is the first large-scale dataset with span-based answers and natural (non-synthetic) passages. It has become a benchmark for neural QA systems [26]–[29]. A more recent version of this dataset is SQuAD 2.0. In addition to the data already in SQuAD, this version has over 50,000 unanswerable question. The main purpose of creating SQuAD 2.0 is to train QA models to handle questions which have no answer.

The other span-based datasets includes NewsQA, TriviaQA and MS MARCO. NewsQA [150] is a crowdsourced dataset with passages obtained from CNN news. It has similar characteristics with SQuAD but the articles (passages) in NewsQA are significantly longer. TriviaQA [151] contains over 650K question-answer pairs. The questions originate from trivia enthusiasts. The passages are obtained independently from various sources including encyclopedic entries, blog articles and news articles. Some questions require multi-sentence reasoning. MS MARCO [152] comprises of 1,010,916 anonymized questions sampled from the search engine logs of Bing. There

are 8,841,823 passages, extracted from 3,563,535 web documents retrieved by Bing.

2.3.2 Multi-step Reasoning Datasets

The first question answering dataset focusing on multi-step reasoning is a synthetic dataset known as bAbI [30]. It is created and released by Facebook. The answer to a question is a single word or entity, which is very different from the index-span datasets. The dataset contains 20 different types of tasks. Each type requires different reasoning abilities. Some of the tasks, such as sequential reasoning, deductive reasoning and inductive reasoning, require reasoning across multiple sentences. The bAbI dataset has aided the development of a number of neural QA systems that can conduct multi-hop reasoning. Such systems are usually made of memory networks [31], [153]. Some of the most successful ones include DMN [20] and DMN+ [32].

Another multi-hop reasoning dataset is HotpotQA [154], which contains 113k Wikipedia-based question-answer pairs. The answers require reasoning over multiple supporting paragraphs. The reasoning types are completely different from those in the bAbI dataset. They are defined according to the type of bridges between the entity in one hop and the following hop. They can be interpreted as knowledge graphs, and this dataset is used for building a Wikipedia hyper-link graph.

Other datasets include QAngaroo WikiHop [155] and QAngaroo MedHop [155] and COMPLEXWEBQUESTIONS [156]. The two QAngaroo datasets are constructed using existing knowledge bases – WikiData and DrugBank respectively. As a result, the answers are entities in these knowledge bases. The COMPLEXWEBQUESTIONS dataset makes use of the web-based knowledge base and focuses on decomposing a complex question into sequences of simple questions, that answer to each of which is

Passage: It was Jessie Bear's birthday. She ...
Question: Who was having a birthday?
Options: (A) Jessie Bear (B) no one (C) Lion (D) Tiger
Answer: A

Figure 2.12: A sample of the multi-choice QA task

extracted through a search engine. All these three datasets involve structured knowledge graphs to a more or less extent.

2.3.3 Multiple-choice Datasets

There are some datasets that are formulated as multiple-choice questions and answers. MCTest [157] is a representative in this group. A sample from MCTest is shown in Figure 2.12. This dataset contains 2000 questions and 500 fictional stories. The complexity level is limited to that of a 7 year-old child. Thus its size is too small for modern data-driven intensive deep learning models.

A larger dataset known as RACE consists of 27,933 passages and 97,687 questions [158]. The samples in RACE are collected from English reading comprehension tasks for Chinese students in middle and high schools. Thus the required understanding level is higher than that for MCTest. Another characteristic of RACE is that the candidate answers are not restricted to be text spans in the original passage and they can be words that do not exist in the passages.

A third multiple-choice QA dataset is QASC [159]. QASC contains 9,980 multiple-choice questions from elementary and middle school level science. In order to answer some questions, common-sense reasoning is required. Hence it is beyond the

Context: The BBC producer allegedly struck by Jeremy Clarkson will not press charges against the “Top Gear” host, his lawyer said Friday. Clarkson, who hosted one of the most-watched television shows in the world, was dropped by the BBC Wednesday after an internal investigation by the British broadcaster found he had subjected producer Oisin Tymon “to an unprovoked physical and verbal attack.” . . .

Question: Producer **X** will not press charges against Jeremy Clarkson, his lawyer says.
Answer: Oisin Tymon

Figure 2.13: A sample of the cloze-style QA task

scope of current QA systems.

2.3.4 Cloze-style Datasets

Cloze-style questions are statements with a missing segment. A sample of this type of questions is shown in Figure 2.13. It is obtained from the CNN/Daily Mail dataset [160]. This dataset contains 1.38M cloze-tyle questions, constructed by blanking out entities in abstractive summaries of CNN and Daily News articles. The answers can be found in the original articles.

Another cloze-style dataset is the Children’s Book Test (CBT) [161]. CBT is built from 108 freely available books. In each sample, twenty consecutive sentences from a chapter of a book are excerpted for the context, and the twenty first sentence serves as the question but with one entity or a common noun removed. The aim is to choose the right answer from the candidate answers. Thus, CBT is a cloze-style multiple-choice dataset. Another similar type of dataset is Who-did-What (WDW) [162]. One distinct characteristic of WDW is that for each sample two separate articles are used, with one serveing as the passage and the other as the basis for the question. Another characteristic is that the segment blanked out from the question is always the

name of a person.

2.3.5 Other Datasets

There exist other datasets. Some require boolean (Yes/No, True/False) answers. Examples are BoolQ [163], AmazonYesNo [164], PubMedQA [165], ShARC [166], and CoQA [167]. They are quite different from the datasets used in this thesis and serves different purposes.

In neural QA systems literature, performances using SQuAD and bAbI datasets are the most commonly cited. Thus, these are the datasets that are used in this research.

Chapter 3

The Baseline QA Model

This thesis aims to provide some understanding to the way each part of a neural QA system contributes to its performance. One important part is the attention mechanism. As the review in Chapter 2 shows, RNN-based QA systems are modular in nature. Therefore they are a good vehicle for our studies.

It is vital to study the relative effectiveness of different methods that are used. One of the obstacles to making fair comparisons is that the architectures of these systems are different. This creates problems in making comparisons. The approach taken in this research is novel. First, the common characteristics of the prevalent RNN-based QA systems are identified. Then a baseline model with all these common characteristics are designed for this study.

In this Chapter, the focus is on the design of the baseline model. It will be used in the studies described in Chapters 4 and 5. The architectural characteristics of the important RNN-based QA models are reviewed in Section 3.1. These characteristics lead to the proposal of the baseline model detailed in Section 3.2. In order to verify that

this baseline model is correct, the performances using this model are compared with those produced by the original models. These results are presented in Section 3.3.

3.1 Architectures of RNN-based QA Systems

A number of RNN-based QA systems can be found in the literature. A representative is BiDAF [27] which appeared in 2017. Its architecture is illustrated in Figure 3.1. It consists of five basic layers. The first layer maps each word to a high-dimensional vector by combining both word-level and character-level embeddings. The second layer is the contextual embedding layer which makes use of a bidirectional Long Short-Term Memory (biLSTM) to capture the temporal dependencies among words for both the passage and the question. This is followed by the attention flow layer that act as an attention mechanism to generate both context-to-query and query-to-context attention-based summaries. The subsequent modeling layer applies another biLSTM to the output of the attention mechanism, to produce query-aware context representations that capture the temporal interactions among them. Finally, the output layer outputs a prediction of the answer through two classifiers which produces the start and end indices of words in the passage.

These layers in the BiDAF model are typical for a number of such QA systems. However, in some models, two adjacent layers may be grouped into a single module. An example is the DCN model [28] which has three modules – a document and question encoder, a co-attention encoder, and a dynamic pointing decoder, as shown in Figure 3.2. The first encoder serves the same functions as the first and second layers of BiDAF but without the character-level embeddings. The co-attention encoder acts as an attention mechanism. But it uses a different attention similarity function and a different approach

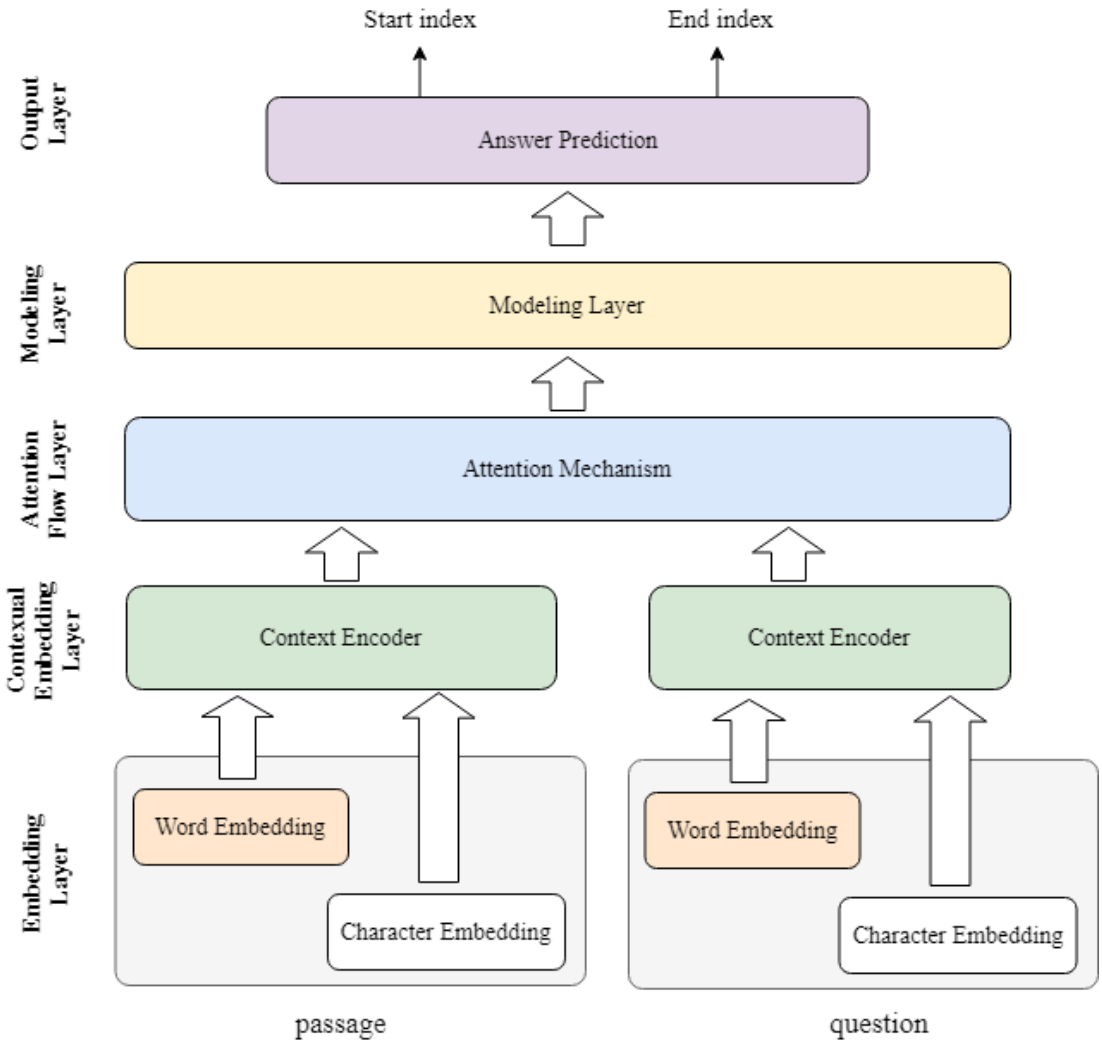


Figure 3.1: Block diagram of the BiDAF model

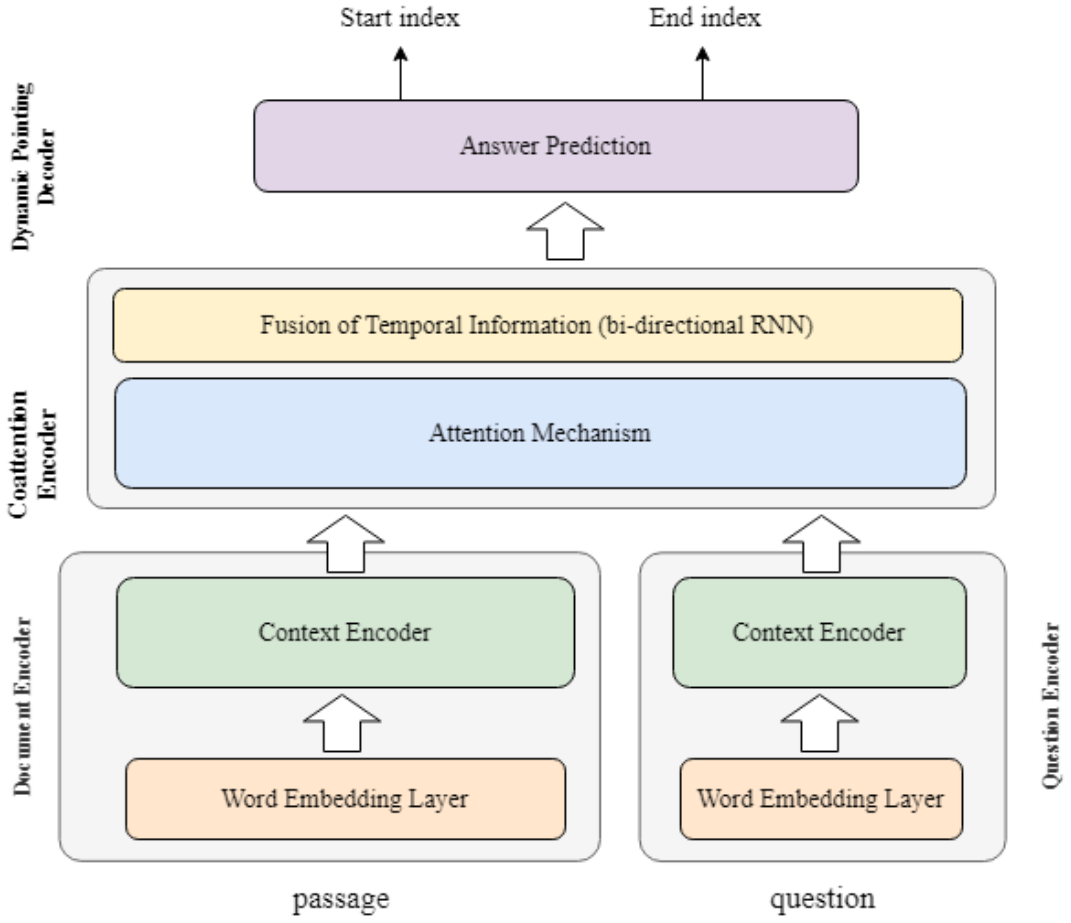


Figure 3.2: Block diagram of the DCN model

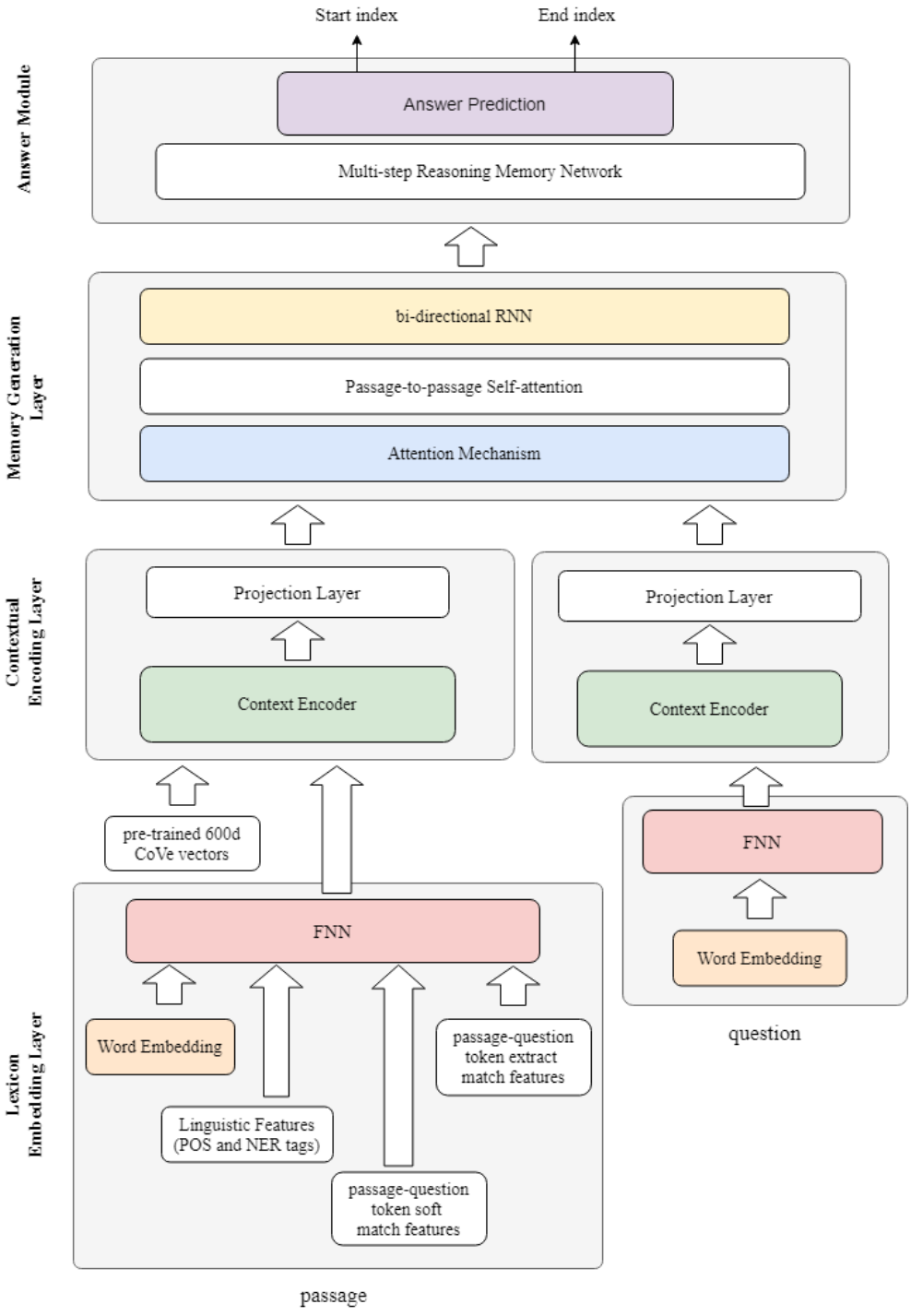


Figure 3.3: Block diagram of the SAN model

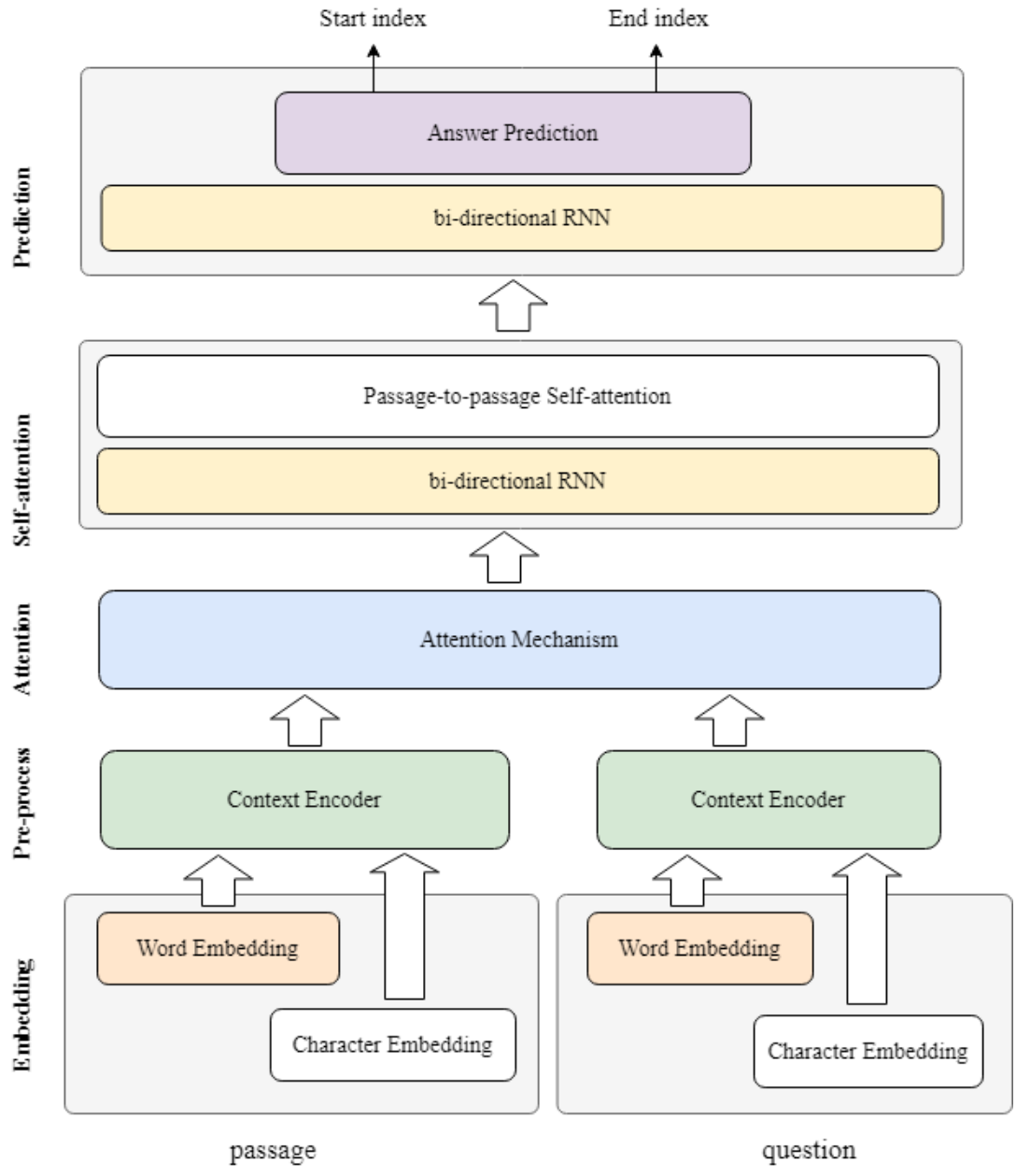


Figure 3.4: Block diagram of the DocQA model

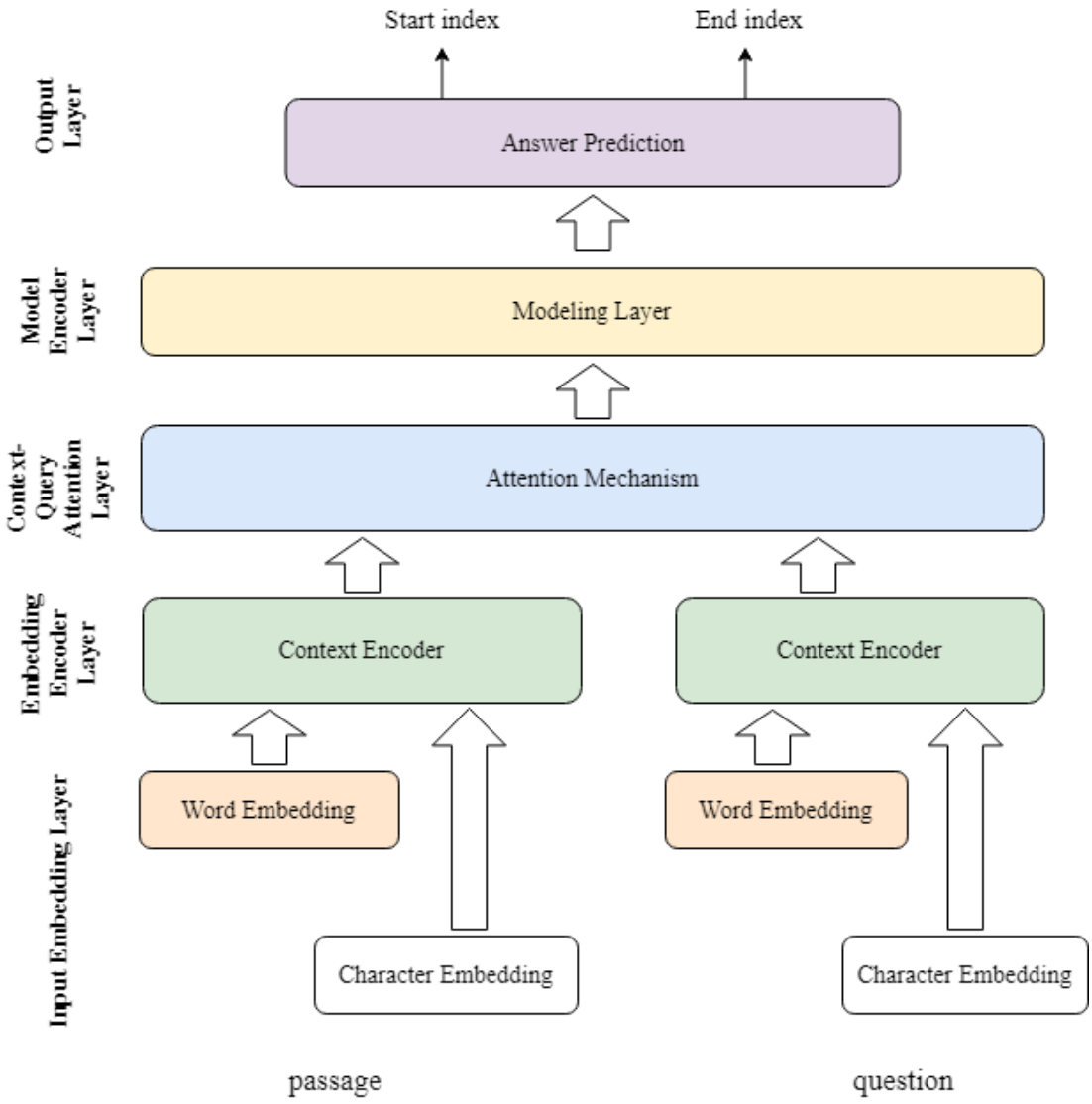


Figure 3.5: Block diagram of the QANet model

to generate passage-aware query summaries. The decoder performs the same functions as the modelling and output layers of BiDAF.

Other examples of such neural QA systems include SAN [26], QANet [29], DocQA [124], DrQA [168], and FusionNet [169]. Some of these models include additional linguistic features in addition to word embeddings [26], [123], [168], [169]. SAN extensively includes several linguistic features and pre-trained vectors, as well as adds a self-attention layer after the attention mechanism, as exhibited in Figure 3.3. DocQA adds an additional self-attention calculation following the attention mechanisms well as displayed in Figure 3.4. QANet demonstrated in Figure 3.5 adopts an alternative way of using RNN to encode sequences, whereas FusionNet focuses on making use of the outputs of all the layers in a stacked biLSTM to create a so-called fully-aware fusion mechanism. Some of these QA systems adopts linear feedforward neural networks to classify the start and end positions of the answer text span [27]–[29], [124], while others use the matrix multiplication to predict the answer span indices [26], [123], [168], [169].

Even though the models mentioned above differ from each other in some aspects, there are four common components in their architectures. First, words are mapped into a high-dimensional vector space called word embeddings. Second, contextual dependencies among the words are captured by an RNN-based context encoder. Third, relevant information from the passage is generated with an attention mechanism. Finally, the output of the attention mechanism is passed into two separate classifiers, which predict the start and end indices of the answer span in the passage, respectively. Furthermore, the way these components are organised is the same as shown in Figure 3.6.

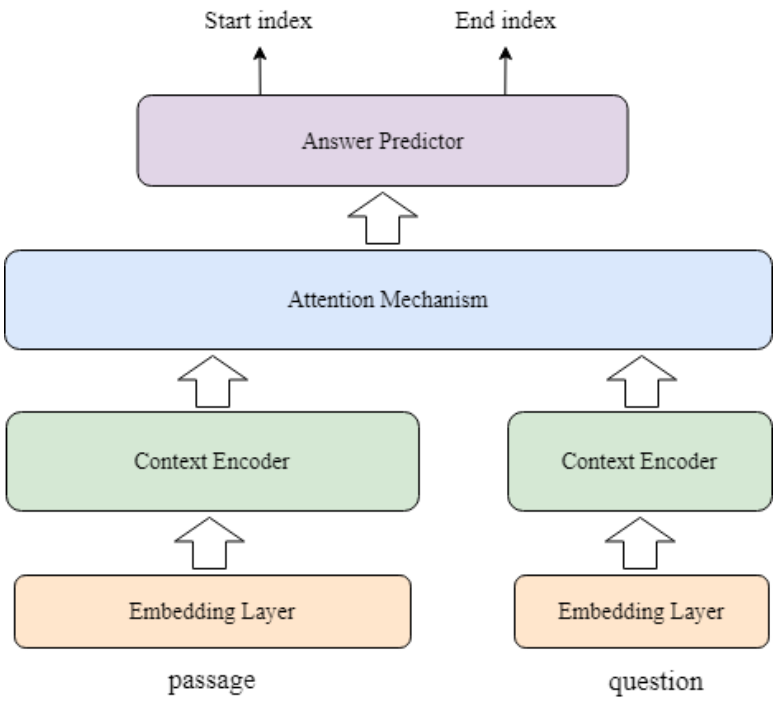


Figure 3.6: The relational illustration of the four components

3.2 The Baseline Model

The baseline model is designed to provide a fair comparison of the effectiveness of various similarity functions detailed in Chapter 4 and several approaches for relevant information generation described in Chapter 5 used in modular attention mechanisms. A block diagram of this model is illustrated in Figure 3.7. It consists of four modules that are used in a range of QA systems that have been described in the previous Section. These modules – embedding layer, context encoder, attention mechanism and answer predictor, are described in detail below.

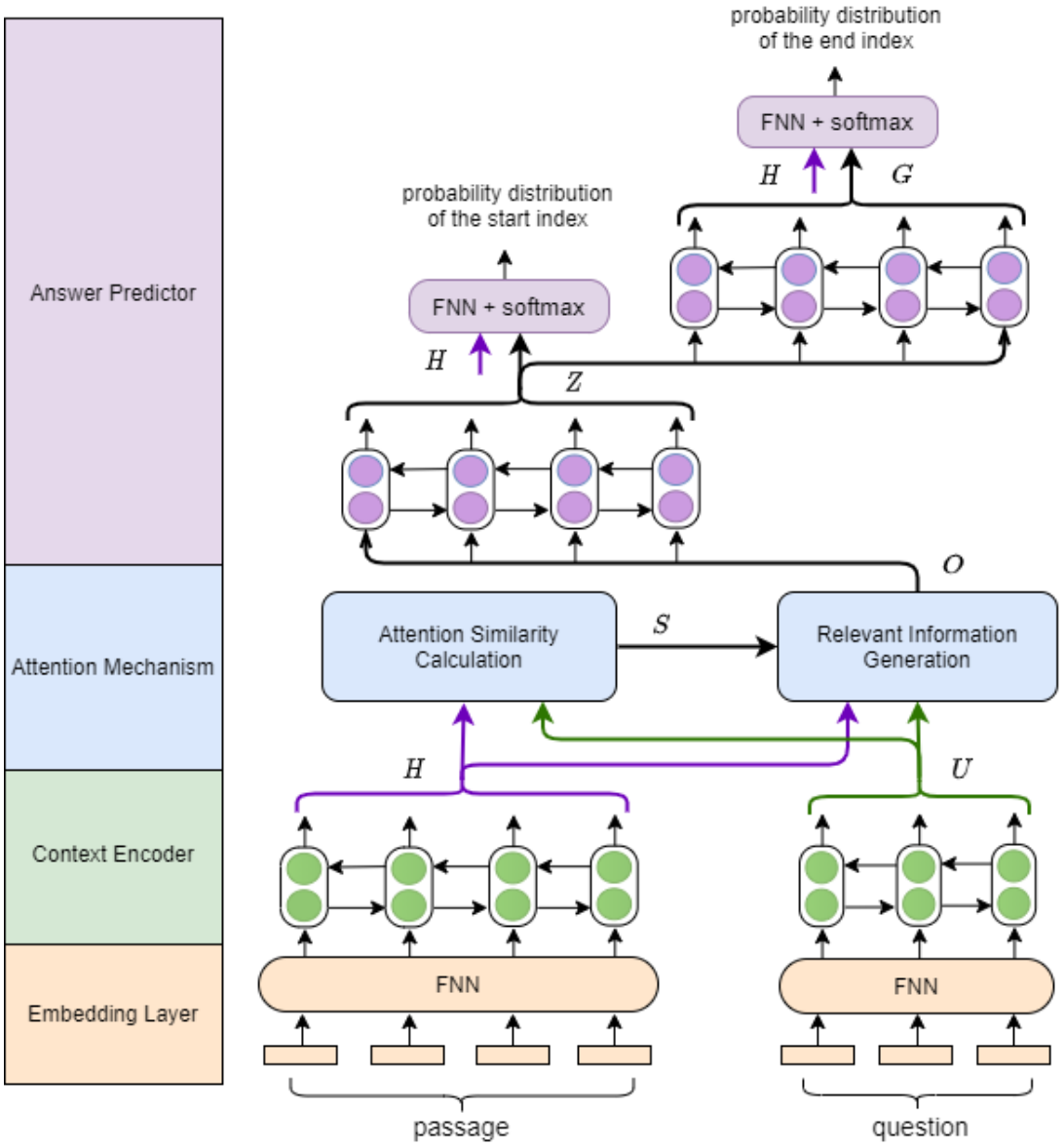


Figure 3.7: Diagram of the baseline model

3.2.1 Embedding Layer

Converting text symbols to word embeddings is required in any neural network-based QA model [20], [26], [27], [31], [32], [169], [170]. Let a passage with M words be denoted by $\{x_1, x_2, \cdots, x_M\}$, and a question with N words by $\{q_1, q_2, \cdots, q_N\}$. In the

word embedding space, the passage is represented as $\{\boldsymbol{e}_1^p, \boldsymbol{e}_2^p, \cdots, \boldsymbol{e}_M^p\} \in \mathbb{R}^{M \times \upsilon}$ and the question as $\{\boldsymbol{e}_1^q, \boldsymbol{e}_2^q, \cdots, \boldsymbol{e}_N^q\} \in \mathbb{R}^{N \times \upsilon}$, where υ is the dimension of the word embedding vector.

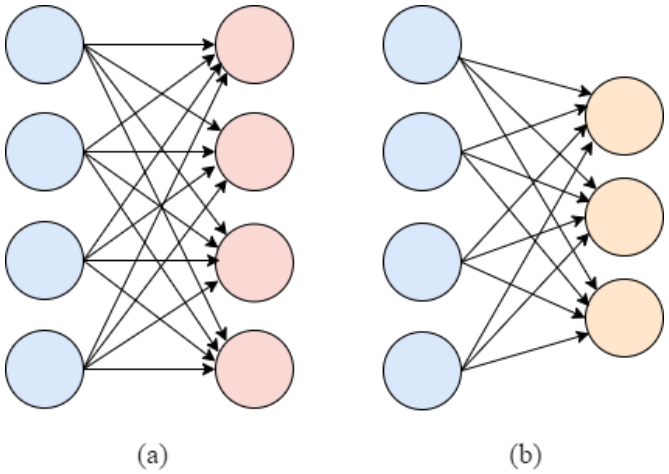


Figure 3.8: The possible usage of the FNN in the embedding layer

Pre-trained word embeddings can be used in line with a number of QA models [26], [29], [124], [137]. Concerning the computational capabilities of the hardware resources on which the baseline model is trained, an adjustment layer is designed by using a feedforward neural network (FNN) to map the word embeddings into another vector space. If the hardware with sufficient memory is available, the dimension of the output layer of the FNN can be set as the same as that of the input layer, referring to Figure 3.8(a). However, if only machines with limited memory can be leveraged to train the model, the dimensionality of the word embeddings can be reduced to the number that suits the hardware, referring to Figure 3.8(b). In the implementation, according to the capability of the computational facilities option (b) is chosen to reduce the 300-dimensional GloVe embeddings to 100-dimensional.

3.2.2 Context Encoder

The context encoder module is used to fuse the positional information of the word sequence into each word representation. Some QA models use unidirectional RNNs to do this work [28], but more QA models adopt the bi-directional RNNs [26], [27], [171]. By using unidirectional RNNs the information on the past context is encoded into the representation of the current time step as the illustration shown in Figure 3.9(a). By using bi-directional RNNs, the information on both the past and the future context can be encoded into the current step as shown in Figure 3.9(b).

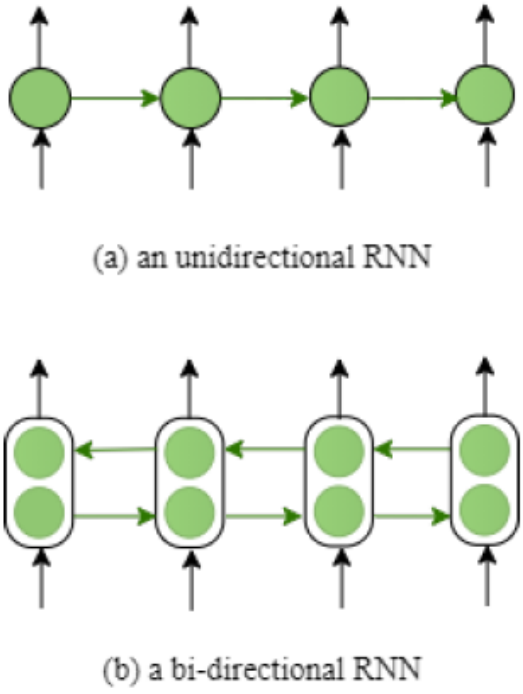


Figure 3.9: Illustration of unidirectional and bi-directional RNNs

In this module, the sequence of words in the passage is passed to a bi-directional LSTM (biLSTM) layer to fuse the positional and contextual information into its hidden states, known as contextual embeddings $H = \{h_t\}_{t=1}^{M} \in \mathbb{R}^{M \times d}$. The sequence of words in the question is also fed into a biLSTM to generate its contextual embeddings

$U = u_t{}_{t=1}^{N} \in \mathbb{R}^{N \times d}$. The calculation of H and U can be expressed as:

$$\boldsymbol{h}_t = biLSTM(\boldsymbol{h}_{t-1}, \boldsymbol{w}_t^p), \tag{3.1}$$

$$\boldsymbol{u}_t = biLSTM(\boldsymbol{u}_{t-1}, \boldsymbol{w}_t^q). \tag{3.2}$$

A more intuitive block diagram of a biLSTM layer is demonstrated in Figure 3.10. The black lines represent the forward direction of a LSTM layer, and the blue lines represent the backward direction.

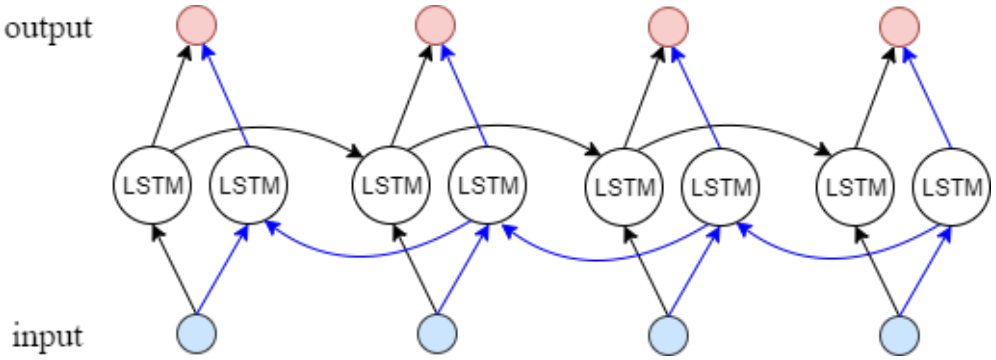


Figure 3.10: Illustration of a bi-directional LSTM layer

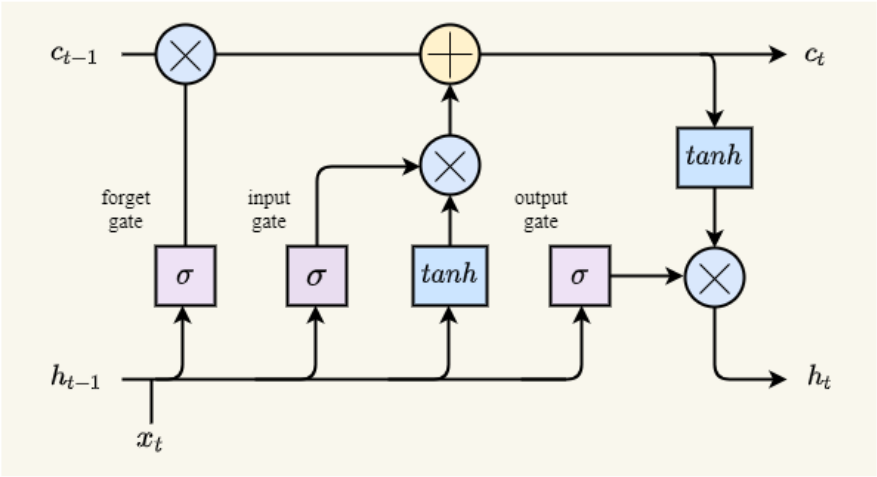


Figure 3.11: The structure of a LSTM cell

A LSTM cell is shown in Figure 3.11. There are three gates – forget gate,

input gate and output gate, to govern what information to discard and what to flow into the next cell. The computation of a LSTM cell at the time step t is as follows:

$$f_t = \sigma(w_f \boldsymbol{x}_t + \boldsymbol{u}_f \boldsymbol{h}_{t-1} + b_f), \tag{3.3}$$

$$i_t = \sigma(w_i \boldsymbol{x}_t + \boldsymbol{u}_i \boldsymbol{h}_{t-1} + b_i), \tag{3.4}$$

$$o_t = \sigma(w_o \boldsymbol{x}_t + \boldsymbol{u}_o \boldsymbol{h}_{t-1} + b_o), \tag{3.5}$$

$$\boldsymbol{c}_t = f_t \circ \boldsymbol{c}_{t-1} + i_t \circ tanh(\boldsymbol{w_a} \boldsymbol{x}_t + \boldsymbol{u}_a \boldsymbol{h}_{t-1} + b_a), \tag{3.6}$$

$$\boldsymbol{h}_t = o_t \circ tanh(\boldsymbol{c}_t). \tag{3.7}$$

Here f_t, i_t, and o_t represent the forget, input and output gates. w_f, w_i, w_o and w_a are the trainable weight vectors associated with the input vector x_t, u_f, u_i, u_o and u_a are the trainable weight vectors associated with another input vector $h_(t-1)$, which is the hidden state of the previous cell.

In a biLSTM layer, each cell in one direction produces a hidden state vector. It means at any time step, there are two LSTM cell, in the forward and backward directions. Their corresponding hidden states are concatenated to pass into the next module. The concatenation is shown in the following equation:

$$\boldsymbol{h}_t = [\boldsymbol{h}_t^{forward}; \boldsymbol{h}_t^{backward}], \tag{3.8}$$

where the semicolon denotes the two vectors are concatenated along their dimension direction.

The advantage of contextual embeddings is that they capture the positional and ordering information of words within a sequence (a sentence or a paragraph) [27], [28], [32], [172]. In other words, contextual embeddings contain the information on the context (what goes before and after) of the current word. As a result, the same word

appearing in different places and representing different meaning will have different contextual embeddings.

3.2.3 Attention Mechanism

As the passage usually contains information that are irrelevant to the question, the main objective of this attention module is to identify the relevant portions of the passage. An example is shown in Figure 3.13. The highlighted parts are the relevant information, which are supposed to be identified by the attention mechanism in this module. The darker the color is, the more relevant the parts are to the question. Numerically this is represented by the similarity scores in the attention mechanism. An attention mechanism usually consists of two parts — similarity score calculation and relevant information generation. They are described in the following subsections.

Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so that the logo could prominently feature the Arabic numerals 50.

Question: Where did Super Bowl 50 take place?

Figure 3.12: Illustration of identifying relevant portions of the passage

3.2.3.1 Similarity Score Calculation

The similarity score calculation component is the first step of an attention mechanism. It computes the scores that quantify the extent of relevance between a piece of texts in one sequence and that in another. More specifically, in this baseline model, this

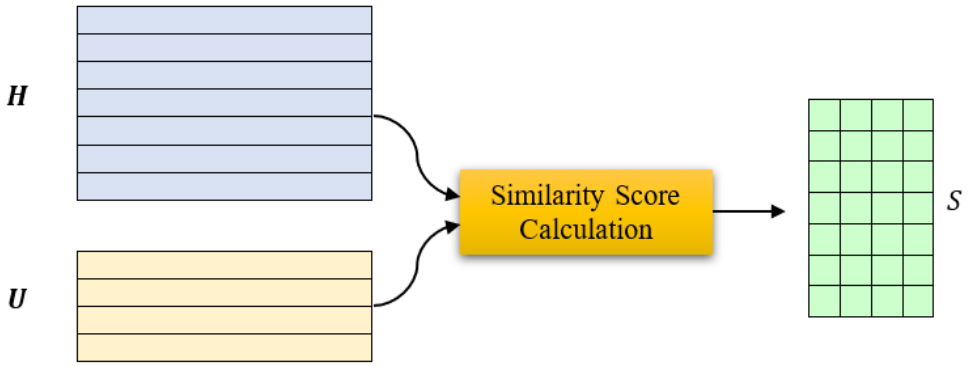


Figure 3.13: Illustration of attention similarity calculation

component calculates word-to-word relevance, each word in the question to each word in the passage.

Given two vectors h_i and u_j representing the i^{th} and j^{th} elements of the contextual embeddings of the passage H and of the question U respectively, the attention similarity score between them is given by

$$s_{ij} = \varphi(\boldsymbol{h}_i, \boldsymbol{u}_j) \tag{3.9}$$

where $\varphi(\cdot)$ is the similarity function. The attention score matrix $S \in \mathbb{R}^{M \times N}$ is obtained by arranging these scores in rows and columns. A matrix illustration is demonstrated in Figure 3.13. Each row represents the similarity between the corresponding word in the passage and all the words in the question. Likewise, each column represents the similarity between the corresponding word in the question and all the words in the passage.

In this baseline model, this computational block purposely isolated from the rest of the model so that φ could be easily replaced in the comparative study. Details of the most common similarity functions are discussed in Chapter 4.

3.2.3.2 Relevant Information Generation

Instead of using the similarity score matrix S directly, it is first normalized. The normalized form of S is known as the attention weight matrix $A \in \mathbb{R}^{M \times N}$ with elements given by

$$\alpha_{ij} = \frac{exp(s_{ij})}{\sum_{k=1}^{N} exp(s_{ik})}. \tag{3.10}$$

Such normalization yields $0 \leq \alpha_{ij} \leq 1$, and they represent the probability of each passage word i being related to each question word j. Note that

$$\sum_{j} \alpha_{ij} = 1, \forall i \tag{3.11}$$

This attention weight matrix A is used to generate attention based summaries. These usually include the context-aware summary $\widetilde{U} \in \mathbb{R}^{M \times d}$, and the question-aware summary $\widetilde{H} \in \mathbb{R}^{N \times d}$. The former is also known as context-to-query summary in some models [27], [124], plays an important role in generating the relevant information in all related attention mechanisms [27], [123], [134], [173]. The latter is only included by some attention mechanisms [17]. Here in the baseline model the context-aware summary is used. It is given by

$$\widetilde{U} = AU \tag{3.12}$$

The relevant information used to predict the answer is function of $\widetilde{U}$ and H and it is denoted by $O = o_{i}{}_{i=1}^{M} \in \mathbb{R}^{M \times d_O}$, where

$$\boldsymbol{o}_i = f(\boldsymbol{h}_i, \widetilde{\boldsymbol{u}}_i). \tag{3.13}$$

f is generally a nonlinear function with trainable weights such as an FNN and d_O is the dimension of the output of f. However, simple concatenation of the operands and their derivative has been shown to yield good results and are used by a number of QA

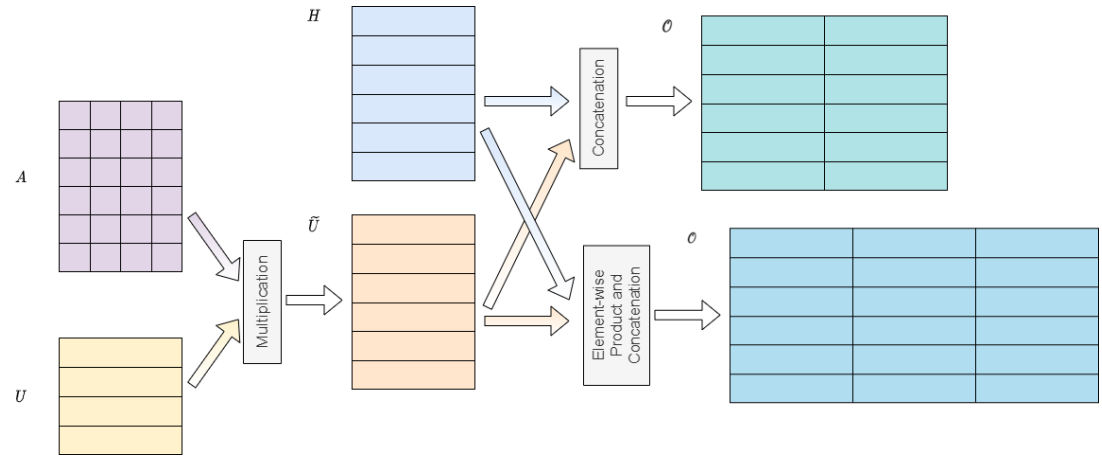


Figure 3.14: Illustration of Relevant Information Generation

models [27], [28]. There are two possible concatenation methods:

$$\boldsymbol{o}_i = [\boldsymbol{h}_i; \widetilde{\boldsymbol{u}}_i] \tag{3.14}$$

with dimension $d_O = 2d$, and

$$\boldsymbol{o}_i = [\boldsymbol{h}_i; \widetilde{\boldsymbol{u}}_i; \boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i], \tag{3.15}$$

where $d_O = 3d$, and $\circ$ denotes the Hadamard product. In general, the relevant information generation component can be illustrated in Figure 3.14.

3.2.4 Answer Predictor

An answer to a question is a text span in the passage for the answer span-based QA tasks. An example is shown in Figure 3.15 with the answer span is highlighted in the passage. The length of an answer span varies depending on the question as well as the passage. Therefore, in order to clearly identify the answer span, its start and end

locations in the passage are rigid and fixed indicators.

Figure 3.15: An example of the answer span in a passage

The answer prediction module is used to predict the start and end indices of an answer span in the associated passage. It is usually carried out through two classifiers. One classifier is for predicting the position of the first answer word in the passage. Similarly, the other classifier predicts the position of the end answer word in the passage. After finding out the start and end indices, the answer can be constructed by using the text from the start position to the end position.

The relevant information generated by the attention mechanism is first passed into a biLSTM to relate the representation at each time step with those preceding and succeeding it. Thus,

$$\boldsymbol{z}_i = biLSTM(\boldsymbol{o}_i,\ \boldsymbol{z}_{i-1}). \tag{3.16}$$

Each cell in this biLSTM also plays the important role of projecting the high-dimensional representation from the attention mechanism to a lower-dimensional space. This layer exists in many QA models with attention [26]–[28], [124], [134].

The probability distributions of the answer's start and end indices in the passage are denoted by P_s and P_e respectively. P_s is obtained from the output of a fully

connected FNN followed by the softmax function. That is,

$$P_s = softmax(\boldsymbol{w}_{P_e}[Z; H] + b_{P_s}), \tag{3.17}$$

where $Z = \{\boldsymbol{z}_i\}_{i=1}^{M} \in \mathbb{R}^{M \times 2d}$, $w_{P_s} \in \mathbb{R}^{4d}$ are the weights of the FNN, and $b_{P_s} \in \mathbb{R}$ is its bias. $P_s = \bigcup_j \{p_{s_j} : j \in \mathcal{J}_s\}$, where $\mathcal{J}_s$ represents the set of all the possible indices that can be taken, and p_{s_j} is the probability at the j^{th} possible value.

The probability distribution of the end index P_e is predicted using another fully connected FNN followed by the softmax function. The inputs to this FNN are H and G, where G is obtained through a biLSTM with input Z. Thus the prediction of the end index is conditioned on that of the start index. This process can be expressed as

$$P_e = softmax(w_{P_e}[G; H] + b_{P_e}), \tag{3.18}$$

where $w_{P_e} \in \mathbb{R}^{4d}$ are the weights of the FNN, and $b_{P_e} \in \mathbb{R}$ is its bias. $P_e = \bigcup_j \{p_{e_j} : j \in \mathcal{J}_e$, where $\mathcal{J}_e\}$ represents the set of all the possible indices, and p_{e_j} is the probability at the j^{th} possible value. $G = \{g_i\}_{i=1}^{M}$ is obtained by

$$g_i = biLSTM(z_i,\ g_{i-1}) \tag{3.19}$$

At the training stage, the two probability distributions are used to calculate the loss function and the objective is to minimize it. The loss function is the sum of the cross entropy [174] of the true and predicted probabilities of the start and end indices, averaged over all the training samples. This can be expressed as

$$L(\theta) = -\frac{1}{K}\sum_{i=1}^{K}(\sum_{j=1}^{J_s} y_{s_j}^{(i)} log(p_{s_j}^{(i)}) + \sum_{j=1}^{J_e} y_{e_j}^{(i)} log(p_{e_j}^{(i)})), \tag{3.20}$$

where θ is the set of all the training weights in the model, K is the number of samples in a training batch, $y_{s_j}^{(i)}$ and $p_{s_j}^{(i)}$ are the true and predicted probabilities of the start index at the j^{th} possible value of the i^{th} training sample, respectively. $y_{e_j}^{(i)}$ and $p_{e_j}^{(i)}$ are the true and predicted probabilities of the end index at the j^{th} possible value of the i^{th} training sample, respectively.

At the testing stage, the predicted start index with the maximum value of $P_e^{(i)}$ is chosen, and likewise for the predicted end index. This can be expressed as

$$start\ index = \underset{p_{s_j}^{(i)} \in P_s^{(i)}}{\arg\max}(P_s^{(i)}), \tag{3.21}$$

$$end\ index = \underset{p_{e_j}^{(i)} \in P_e^{(i)}}{\arg\max}(P_e^{(i)}), \tag{3.22}$$

where $P_s^{(i)}$ represents the probability distribution of the start index of the i^{th} testing sample. Likewise, $P_e^{(i)}$ represents the probability distribution of the end index of the i^{th} testing sample.

3.3 Experiment Results

Experiments are conducted to verify that the baseline model produces results that are commensurate with the original models.

3.3.1 Dataset and Evaluation Metrics

The experiments make use of the SQuAD dataset which is a popular benchmark for testing QA and machine reading comprehension systems [149]. Its passages were collected from a wide range of articles in Wikipedia. The answer to each question

can be found in the corresponding passage, identified by the text-span in the passage. This dataset contains 107,785 question-answer pairs obtained by the crowd sourcing based on the articles. Each answer to a question is the text span from the corresponding passage. It covers a range of question types.

Two evaluation metrics are used – EM and F1. EM is a metric that measures the rate or percentage of exact match with the ground truth. F1 measures the harmonic mean of precision and recall. For each question, precision is the ratio of the number of correct words and the number of words in the predicted answer. Recall is calculated as the number of correct words divided by the number of words in the ground truth answer. The F1 score is computed per question and then averaged across all questions.

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}, \tag{3.23}$$

where

$$Precision = \frac{number\ of\ correct\ words}{number\ of\ words\ in\ the\ predicted\ answer}, \tag{3.24}$$

$$Recall = \frac{number\ of\ correct\ words}{number\ of\ words\ in\ the\ true\ answer}. \tag{3.25}$$

3.3.2 Training Setup

The baseline models in the experiment are trained using the following setup parameters. The word embeddings are 300-dimensional GloVe vectors [1] pre-trained on 840 billon tokens in Common Crawl [94]. The out-of-vocabulary (OOV) words are set to be zero vectors. The output dimension of the FNN in the input module is set to be 100. The training batch is set to be 60, and the number of epochs is 20. The Adadelta optimizer [175] with an initial learning rate of 0.5 is used. A 0.2 dropout rate is applied

[1] downloaded from `https://nlp.stanford.edu/projects/glove/`

to the input for each RNN and linear layers, except for the linear layers in the answer prediction module. An exponential moving average (EMA) of all the trainable weights with a decay rate of 0.999 are utilised. In the testing stage, these averages are used for predicting the answers. The sum of the cross-entropy functions, one for the start index prediction and the other for the end index prediction, are used as the loss function.

3.3.3 Results

The predictive EM and F1 scores of five existing models along with the baseline model with the attention mechanism in each of these models are listed in Table 3.1. These results obtained by the baseline model are lower than the existing QA models. This makes sense because the baseline model only reflects the common invariant modules of the existing models, which means the existing models contain some components that are not included in the baseline model.

Table 3.1: Comparison of five existing QA models with the baseline model with their attention

QA Models	**Original Model**		**Baseline Model with existing model's attention**	
	EM	**F1**	**EM**	**F1**
BiDAF	67.7	77.3	65.6	76.3
DCN	65.4	75.6	64.3	75.0
DocQA	72.1	81.1	65.6	76.3
QANet	73.6	82.7	65.6	76.3
SAN	76.2	84.1	63.5	74.2

Another point is that the dimension of the word embeddings in the baseline model is reduced from 300d to 100d to accommodate the computational capacity of

the hardware. It means the scale of the overall model becomes 1/3 smaller, as a result, the learning capacity of the whole model reduces. This leads to a natural performance decrease of the model. This phenomenon is commonly shown in various neural network based models, such as the different performance of the large, medium and small versions of the GPT2 models [140].

Comparing with the baseline model, the BiDAF has two more components. One is the character embedding and the associated network that integrate the character and word embedding. The word embeddings are 300d, three times of those for the baseline model. After integrating with the character embeddings, the dimension of the input embeddings for BiDAF is 400d, while the baseline model's are 100d. The hidden state of the BiDAF and the baseline model is the same.

The one component in BiDAF is the modeling layer, which consists of two layers of bi-directional LSTM. However, in the baseline model, this component is one layer of bi-directional LSTM, contained in the answer layer. As the BiDAF uses the outputs of both the two layers, this may contribute more information for the subsequent answer prediction.

DocQA and QANet have the same attention mechanism as the BiDAF model, but each of them have different additional components. DocQA has three aspects that not contained in the baseline model. DocQA has the same input as the BiDAF model. The second aspect is that DocQA contains a passage-to-passage self-attention after the attention mechanism. This component allows the relevant information produced by the attention mechanism to further interact with itself, which may be able to learn more useful information for the subsequent answer prediction module. The third aspect is that DocQA applies a unique dropout technique, called variational dropout, to the input of some layers. This may helps to improve the performance of the model.

QANet introduces an alternative way to RNNs by making use of convolutions and self-attention to do what RNNs do in the other models. The overall architecture still follows those of RNN-based QA systems, as QANet was developed as a recurrency-free version of BiDAF. As a result, QANet has all the additional components that BiDAF contains, besides, the non-recurrent operations replacing RNNs.

DCN uses 300d word embeddings as the input. Another point is that DCN incorporated sophisticated components in the answer prediction module. It adopts a LSTM-based sequential model and a new technique called highway maxout network to predict the answer. This sophisticated techniques in the answer module may result in a better performance.

SAN is the most complex model among the five existing ones, and has the most additional components that are not included in the baseline model. First, SAN included three different types of features into the input apart from the 300d pre-trained word embeddings, which are linguistic features and passage-question exact matching features and question enhanced passage word embeddings. In total, the input vector is 600d for passages. In addition, another pre-trained 600d vectors called CoVe are used as part of the input for the context encoder module. Second, after the attention mechanism there is a passage-to-passage self-attention layer. Next, before predicting the answer, SAN adopts a multi-step reasoning memory network and a stochastic dropout strategy. All these three points contributes to a higher performance of the model.

3.4 Summary

This Chapter described the design of a novel baseline neural QA model based on the common characteristics of several representative RNN-based QA models. Experimental

results using the SQuAD dataset verify that the baseline model produces results that are commensurate with these original models. Thus this baseline model can be used for further studies into the design of the attention mechanisms which will be discussed in Chapters 4 and 5.

Chapter 4

Effects of Attention Similarity Functions

The attention layer of RNN-based QA systems was discussed in Section 3.2.3. Two steps are involved – similarity score calculation, and relevant information generation. Studying how the attention similarity function affects the performance of such systems is the focus of this Chapter. In Section 4.1, eleven similarity score functions are described. They can be categorised into two groups – additive and multiplicative. Performance comparisons using these functions are presented in Section 4.2. Based on the insights obtained from these results, a new similarity score function is proposed in Section 4.3. This new function shows superior performance to all the existing functions.

4.1 Similarity Score Functions

As discussed in Section 3.2.3.1, an attention similarity score is computed for the contextual embeddings of the passage and the question. A number of different ways

have been proposed to compute this similarity score. One of the early approaches is the concatenation-based method proposed for machine translation by Bahdanau et al [176]. With this method, the two input vectors, as shown in Figure 3.13, are concatenated and fed into an FNN to produce the similarity score. A slight variation of this method is used in [137]. The two input vectors are first multiplied with their respective trainable matrices. The results are then summed. Other variations could also be found. In [27], the two inputs and their element-wise product are each projected onto a scalar value and subsequently summed. All these aforementioned methods of computation would be classified as additive attention.

A different approach is based on inner products. In [132], the dot product of the two input vectors are computed. In other works [26], [123], [133], the inputs are first transformed before taking their dot product. A simple transformation is to scale the dot product by a factor dependent on the dimension of the input vectors [133]. Another way is to include trainable parameters in the dot product [132], [177]. These methods are classified as multiplicative attention.

These similarity score computation methods can be expressed as mathematical functions. A summary of these functions are listed in Table 4.1. More details of these functions are given below.

4.1.1 Additive Similarity Functions

The trilinear function is often-used in QA models [27], [29], [124]. It contains three trainable weight vectors $\boldsymbol{w}_p$, $\boldsymbol{w}_q$ and $\boldsymbol{w}_{pq}$, associated with $\boldsymbol{h}_i$, $\boldsymbol{u}_j$ and their element-wise product $\boldsymbol{h}_i \circ \boldsymbol{u}_j$ respectively. Another form of this function is $\boldsymbol{w}_\varphi[\boldsymbol{h}_i;\ \boldsymbol{u}_j;\ \boldsymbol{h}_i \circ \boldsymbol{u}_j]$, containing only one trainable weight vector $\boldsymbol{w}_\varphi$ that associates with the concatenation

Table 4.1: Similarity score functions compared in this chapter

Group	**Types**	$\varphi(\boldsymbol{h}_i, \boldsymbol{u}_j)$ forms
Additive	trilinear [124]	$\boldsymbol{w}_p\boldsymbol{h}_i + \boldsymbol{w}_q\boldsymbol{u}_j + \boldsymbol{w}_{pq}(\boldsymbol{h}_i \circ \boldsymbol{u_j})$ or $\boldsymbol{w}_\varphi[\boldsymbol{h}_i; \boldsymbol{u}_j; \boldsymbol{h}_i \circ \boldsymbol{u}_j]$
	FNN [137]	$\boldsymbol{w}_\varphi tanh\left(W_p\boldsymbol{h}_i + W_q\boldsymbol{u}_j\right)$
	concat-FNN [176]	$\boldsymbol{w}_\varphi tanh(W_p[\boldsymbol{h}_i; \boldsymbol{u}_j])$
	bilinear*	$\boldsymbol{w}_p\boldsymbol{h}_i + \boldsymbol{w}_q\boldsymbol{u}_j$
Multiplicative	dot-product [28], [31], [132], [173]	${\boldsymbol{h}_i}^T\boldsymbol{u}_j$
	cosine [178]	$\frac{{\boldsymbol{h}_i}^T\boldsymbol{u}_j}{\|\boldsymbol{h}_i\|_2\|\boldsymbol{u}_j\|_2}$
	scaled dot-product [133], [179]	$\frac{{\boldsymbol{h}_i}^T\boldsymbol{u}_j}{\sqrt{d}}$
	general-1 [132]	${\boldsymbol{h}_i}^T W\boldsymbol{u}_j$
	general-2 [177]	$(\boldsymbol{W}_1\boldsymbol{h}_i)^T(\boldsymbol{W}_2\boldsymbol{u}_j)$
	general-3 [26]	$\langle f\left(\boldsymbol{W}_1\boldsymbol{h}_i\right), f\left(\boldsymbol{W}_2\boldsymbol{u}_j\right)\rangle$
	Hadamard [123]	$\boldsymbol{w}^T(\boldsymbol{h}_i \circ \boldsymbol{u}_j)$

Note: $\boldsymbol{w}_p$, $\boldsymbol{w}_q$, $\boldsymbol{w}_{pq}$, $\boldsymbol{w}_\varphi$ and $\boldsymbol{w}$ are trainable vectors. $\boldsymbol{W}_p$, $\boldsymbol{W}_q$, $\boldsymbol{W}$, $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are trainable matrices. $f()$ is the $ReLU$ activation.

* This is not generally used as an attention similarity function, but including it helps to gain more insights into the other methods.

of the three vectors. The two forms are the same theoretically, as $\boldsymbol{w}_\varphi$ is just the concatenation of the $\boldsymbol{w}_p$, $\boldsymbol{w}_q$ and $\boldsymbol{w}_{pq}$.

The FNN method first multiplies $\boldsymbol{h}_i$ and $\boldsymbol{u}_j$ with two trainable matrices W_p and W_q respectively. From another perspective, this method passes $\boldsymbol{h}_i$ and $\boldsymbol{u}_j$ to two different linear FNN layers. The outputs of these two layers are summed before being compressed by the *tanh* function. The resulting vector is then passed into another linear

layer to produce the similarity score [137].

The concat-FNN function first concatenates the two input vectors $\boldsymbol{h}_i$ and $\boldsymbol{u}_j$. The concatenation is then fed to a nonlinear FNN to produce the similarity score [115]. Comparing with the FNN function, the concat-FNN requires more computational memory.

The bilinear function has not used as an attention similarity function. However, it helps to show the effects of removing the element-wise product term from the trilinear function.

4.1.2 Multiplicative Similarity Functions

The dot-product is the most commonly used similarity function for neural QA models. The dot product between $\boldsymbol{h}_i$ and $\boldsymbol{u}_j$ can be expressed as $\boldsymbol{h}_i{}^T\boldsymbol{u}_j = |\boldsymbol{h}_i||\boldsymbol{u}_j|cos\theta$, where θ is the angle between the two vectors. Hence it is related to the cosine function by the scaling factor $\frac{1}{\|\boldsymbol{h}_i\|_2\|\boldsymbol{u}_j\|_2}$. The advantage of the cosine function is that its magnitude is limited to within ± 1. It was used early on for neural Turing machines [178] to compute the similarity of two representations.

The Transformer model [133] makes use of the scaled dot-product for its self-attention mechanism. The scaling factor is $\sqrt{d}$, where d is the dimension of the vectors. This scaling apparently promotes efficient learning. As the transformer-based pre-trained language models are becoming ubiquitous, the scaled dot-product function is also increasingly utilized [138]–[141].

The *general-1* function generalizes the dot product function by introducing a trainable weight matrix $\boldsymbol{W}$. The *general-2* function further incorporates two trainable

weight matrices, one for each vector. These two methods make it possible to calculate the attention score for the case where the two vectors have different dimensions. The *general-3* method rectifies the linearly transformed input vectors with the ReLU activation function before computing the dot-product. The $ReLU$ function is defined as $f(x) = \max(0, x)$.

The Hadamard method has never been used as a similarity function of attention. However, it was used to calculate the similarity for generating word-in-question features to augment word embedding representations [123]. It has been included to see if other forms of multiplicative similarity functions may be useful. Here, $\boldsymbol{w}$ is a trainable weight vector.

4.2 Experimental Comparisons of the Similarity Functions

With the baseline model, fairly evaluating the effects of the similarity score functions on the performance of QA systems can be conducted. The dataset and the evaluation metrics used for the experiments here are the same as those described in Section 3.3.1. The training configuration is the same as those in Section 3.3.2.

Two different ways of generating the output of the attention mechanism are given by Equations (3.14) and (3.15). For the sake of convenience, these two ways are denoted by O_1 and O_2 respectively. The best EM and F1 scores for models obtained from five training runs are shown in Table 4.2. Figure 4.1 shows box and whisker plots of the corresponding EM scores. The red circles and orange horizontal lines indicate the average and median values respectively.

Table 4.2: EM and F1 scores with different attention similarity score functions for O_1 and O_2

Groups	Similarity Score Function	O_1		O_2	
		EM	**F1**	**EM**	**F1**
Additive	bilinear	49.9	60.6	58.2	69.0
	trilinear	**64.4**	75.0	**65.1**	**75.5**
	FNN	64.1	74.6	64.8	75.2
	concat-FNN	64.0	74.8	**65.1**	75.4
Multiplicative	dot-product	63.8	74.5	64.2	75.0
	cosine	54.5	65.7	60.7	71.8
	scaled dot-product	63.3	74.0	63.5	74.4
	general-1	63.9	74.5	64.8	75.0
	general-2	64.1	74.9	64.0	74.7
	general-3	63.5	74.2	64.9	75.3
	Hadamard	64.3	**75.2**	64.7	75.3

The results in Table 4.2 show that, generally, using O_2 is better than O_1 for both EM and F1 score. The only exception is with the similarity score function *general-2*. This indicates the element-wise product between the contextual passage embeddings and the context-aware query representations in O_2 helps to improve the performance.

The Hadamard function is the best performer overall, with the EM score only slightly below *general-3* for O_2. Figure 4.1 shows that the EM scores vary from 55.0 to 64.3 and the F1 scores from 67.0 to 75.5. In other words, the performance of the models using the Hadamard function in the O_1 setting is not robust across different runs. However, no such problem exists in the O_2 setting. This indicates that the

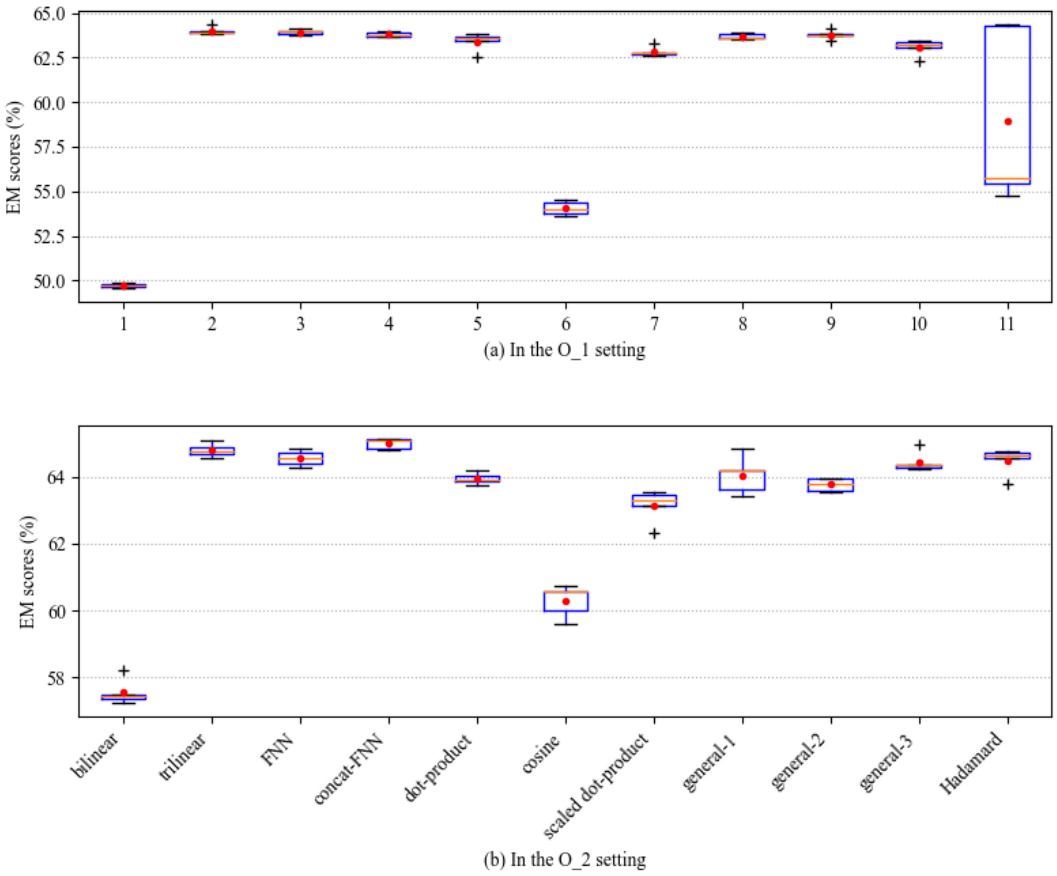


Figure 4.1: The box plots of EM scores under O_1 and O_2

inclusion of the $\boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i$ term plays a vital role in improving the robustness of the model with this similarity score function. Recall that the output of the attention mechanism is passed to a recurrent neural network. A possible reason is that the element-wise product of two vectors contains some information that the neural network cannot learn from the concatenation of the two vectors. Remember that the output of the attention mechanism is passed to a recurrent neural network.

Comparing the results of additive and multiplicative functions, the highest

scores are obtained from the additive group, especially the trilinear attention function. The bilinear has the two terms, $\boldsymbol{w}_p\boldsymbol{h}_i$ and $\boldsymbol{w}_q\boldsymbol{u}_j$, which are also contained in trilinear. However, because of the lack of the $\boldsymbol{h}_i \circ \boldsymbol{u}_i$ term, it produces much lower scores. This suggests that the relation between the two input vectors are not bilinear, and including the $\boldsymbol{h}_i \circ \boldsymbol{u}_i$ term into the bilinear transformation improves the predictive accuracy of the model significantly, by 14.5 and 6.9 absolute points in EM under the O_1 and O_2 settings respectively. Another possible reason is that with more parameters, the trilinear function has a higher capacity for improved accuracy. This aligns with the observations in the literature on network design space [180], [181]. Although the FNN function has similar parameters, it is more nonlinear than linear. Its slightly poorer results compared to the trilinear function suggest that the information of the element-wise product of two vectors may not be fully learned through passing the concatenation of the same two vectors into a feedforward neural network.

It is interesting to note that the scaled dot-product function, a default preference in pre-trained language models, performs the same or slightly worse than the simple dot-product. This could be due to the scaling factor, defined as the squared root of the dimensionality of the vectors, which is quite large. This may cause the magnitudes of the similarity score to be reduced to such an extent that their values are too close to each other, especially after being normalised by the softmax function. In order to confirm this hypothesis, the similarity scores for the dot-product, the scaled dot-product, and the cosine function are retrieved from their corresponding trained models for the same test sample. The histograms of their similarity scores are shown in Figure 4.2. It clearly shows that the range of similarity scores produced using the scaled dot-product function is significantly less than those produced using the dot-product.

Cosine is the other function which produces similarity score within a small finite range by its nature. It produces the poorest results. Thus, it is clear that based

on this type of system architecture, the similarity function should be able to produce a larger range of similarity values to be effective.

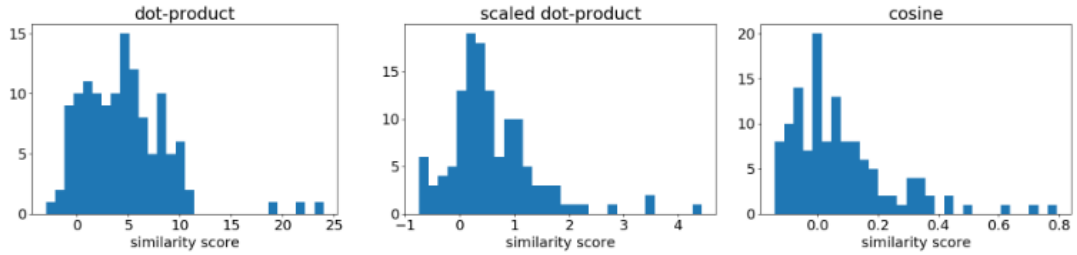


Figure 4.2: Histograms of the similarity scores for the three functions

4.3 Proposing A New Attention Similarity Function

The findings presented in the previous Section produced sufficient insight into designing a better attention similarity function. In the additive group, the trilinear function achieves the best results among all the similarity functions. While the bilinear shows the worst performance. The only difference between these two functions is the element-wise $\boldsymbol{h}_i \circ \boldsymbol{u}_i$ associated term. Projecting the element-wise product along with the two input vectors helps to improve the performance of the model significantly. In the multiplicative group, the *ReLU* function that is used to transform the two input vectors in the *general-3* function helps to improve the performance noticeably in the O_2 setting.

A new similarity score function that combines these two advantages is proposed. It makes use of the *ReLU* transformation together with the trilinear function. This new attention score function is named T-trilinear (transformed trilinear). It can be described mathematically as:

$$\boldsymbol{h}_i^{\mathrm{t}} = ReLU(\boldsymbol{W}_1 \boldsymbol{h}_i + b_1) \tag{4.1}$$

$$\boldsymbol{u}_i^{\mathrm{t}} = ReLU(\boldsymbol{W}_2\boldsymbol{u}_i + b_2) \tag{4.2}$$

$$\varphi(\boldsymbol{h}_i, \boldsymbol{u}_j) = \boldsymbol{w}_p\boldsymbol{h}_i^t + \boldsymbol{w}_q\boldsymbol{u}_i^t + \boldsymbol{w}_pq(\boldsymbol{h}_i^t \circ \boldsymbol{u}_i^t) \tag{4.3}$$

where $\boldsymbol{W}_1$ and $\boldsymbol{W}_2 \in \mathbb{R}^{d\times d}$, b_1 and $b_2 \in \mathbb{R}^{d\times 1}$, $\boldsymbol{w}_p$, $\boldsymbol{w}_q$ and $\boldsymbol{w}_pq \in \mathbb{R}^{d\times 1}$. The two input vectors $\boldsymbol{h}_i$ and $\boldsymbol{u}_i$ are passed to a FNN with the *ReLU* activation before going through the trilinear function to get the similarity score. The *ReLU* function deactivates the neurons to get the desired output while keeping the efficiency of the computation.

4.3.1 Evaluation Results

Table 4.3: Comparison of EM and F1 scores for the best performing attention similarity functions

Attention Score Function	**O_1**		**O_2**	
	EM	**F1**	**EM**	**F1**
trilinear	64.4	75.0	65.1	75.5
FNN	64.1	74.6	64.8	75.2
concat-FNN	64.0	74.8	65.1	75.4
general-3	63.5	74.2	64.9	75.3
T-trilinear	**64.5**	**75.2**	**65.3**	**75.8**

The EM and F1 scores of the T-trilinear function in comparison with trilinear, FNN, concat-FNN and *general-3*, using the baseline model, are shown in Table 4.3. It can be seen that the proposed T-trilinear method achieves the best results under both O_1 and O_2 settings. Figure 4.3 shows the distributions of the results through different runs. It demonstrates that the performance of the T-trilinear function is consistent between runs, in particular in comparison with the Hadamard function.

The loss curves, which show the training convergence behaviour, with these

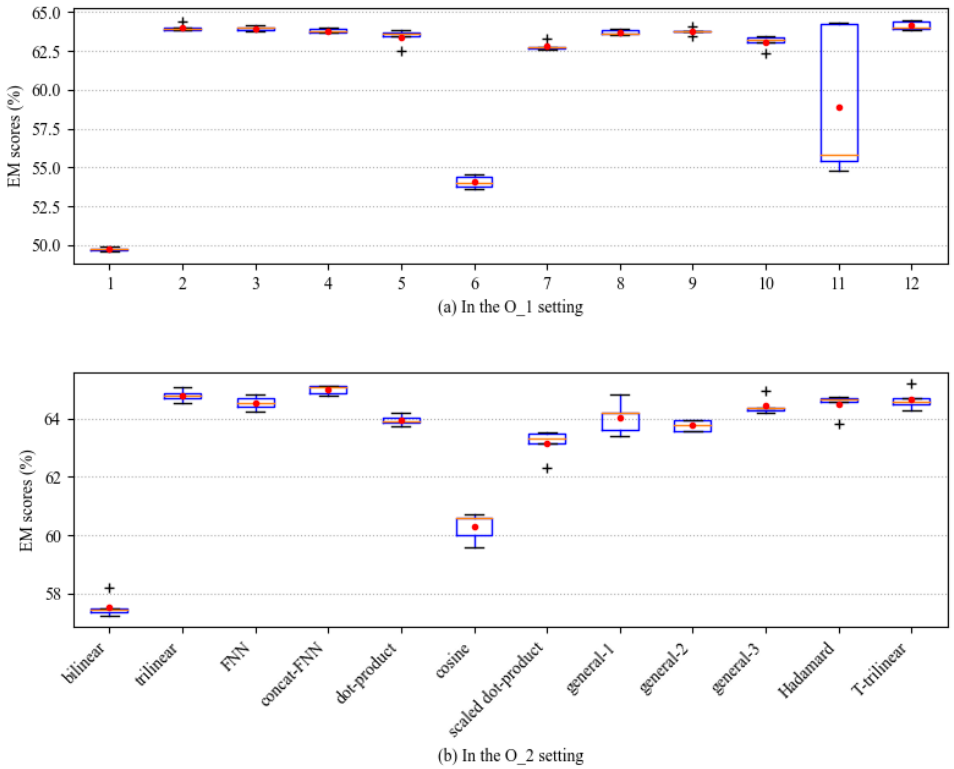


Figure 4.3: The box plots of EM and F1 scores under O_1 and O_2

similarity score functions are shown in Figure 4.4. It can be observed that all the functions converge faster in the O_2 setting than in the O_1 setting. In the O_1 setting, the one that uses the trilinear function converges most rapidly in the early stage, followed by general-3 slightly slower. All the others demonstrate quite similar convergence patterns. The proposed T-trilinear function shows a similar trend to FNN and concat-FNN in the initial stages. This is not surprising as the three have relatively more trainable weights and the models need to search longer for the optimal combinations. After that, their losses continue to decrease and end up at a similar level as the other two functions. In the O_2 setting, all the functions display similar convergence trends and T-trilinear reaches a relatively lower loss in the later stage.

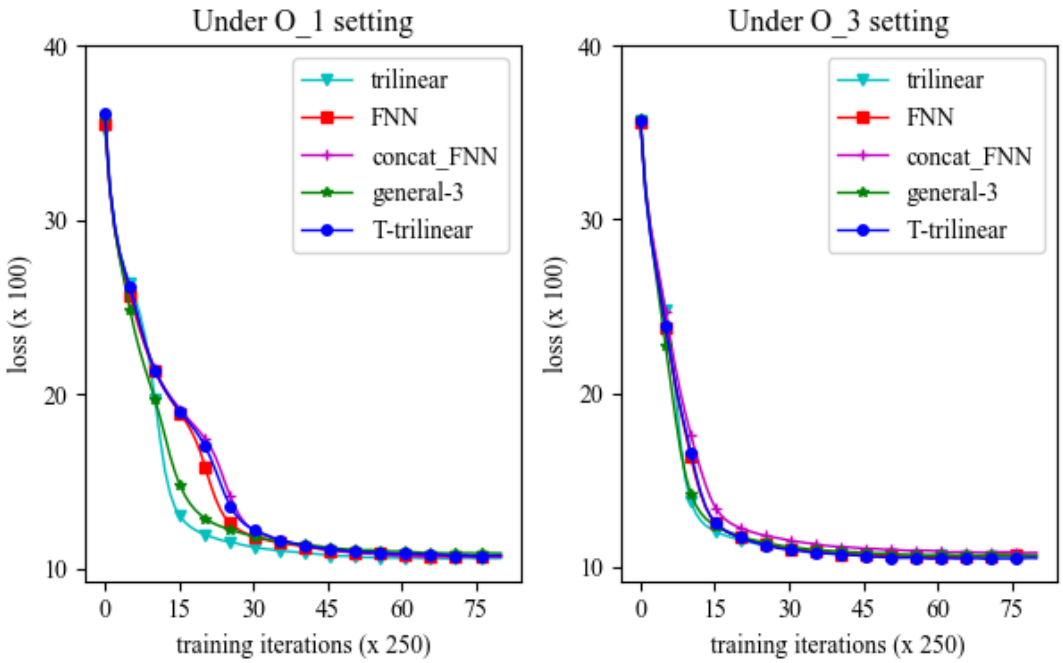


Figure 4.4: Loss curves of the baseline model with the similarity functions listed in Table 4.3

In order to confirm that the proposed T-trilinear function works well with the original QA models, it was implemented on BiDAF, DCN and QANet. The original similarity functions for both BiDAF and QANet is the trilinear function and DCN uses the dot product. The comparison results in Table 4.4 show that using T-trilinear function, both the EM and F1 scores are higher for all three models. These results confirm that the T-trilinear function is indeed more effective compared with the other similarity functions.

A closer look at the extents of performance improvement in Tables 4.4 and 4.3 show that the improvement in the three existing models is more obvious than that of the baseline model. This indicates that the proposed T-trilinear function may work better in "real" models. This is likely due to additional components included in the "real" models but not in the baseline model as the latter only consists of the common

characteristics of these RNN-based QA models.

Table 4.4: EM and F1 scores of three QA models with their original attention similarity score functions in comparison with the proposed T-trilinear function

Models	**Original similarity function**	**Original similarity function**		**T-trilinear**	
		EM	**F1**	**EM**	**F1**
BiDAF	trilinear	67.7	77.3	**68.5**	**77.9**
DCN	dot-product	65.4	75.6	**65.7**	**76.1**
QANet	trilinear	73.6	82.7	**74.9**	**83.1**

4.3.2 Heatmap Visualization

In order to find explanations why the proposed T-trilinear performs well, the similarity scores between words in the passage, the question and the ground truth answer of a test sample chosen from the dataset are examined. Figure 4.5 shows the passage, the question, and the ground truth answer of the chosen sample. The similarity scores between individual words in the passage and those in the question are plotted as heatmaps in Figure 4.6. In comparison, those for trilinear and *general-3* functions are also plotted. The x-axis of the heatmap are words from the question and the y-axis are words from the passage. The colour of each cell in a heatmap represents the attention similarity score between the corresponding words in the x and y axes. The darker the colour, the higher the score. Results in both the O_1 and O_2 settings are shown.

Intuitively the word “day” in the question should be more related to the word span “February 7, 2016" in the passage. Consider Figure 4.6(a) which is for the O_1 setting. For the trilinear and *general-3* functions, the cells corresponding to the answer words “February 7, 2016" display no distinct darker colour compared with other cells. This shows that these two attention functions fail to pay attention to the related text

… The American football conference (AFC) champion Denver broncos defeated the national football conference (NFC) champion Carolina panthers 24–10 to earn their third super bowl title. the game was played on February 7, 2016, at Levi's stadium in the san Francisco bay area at Santa Clara, California. …

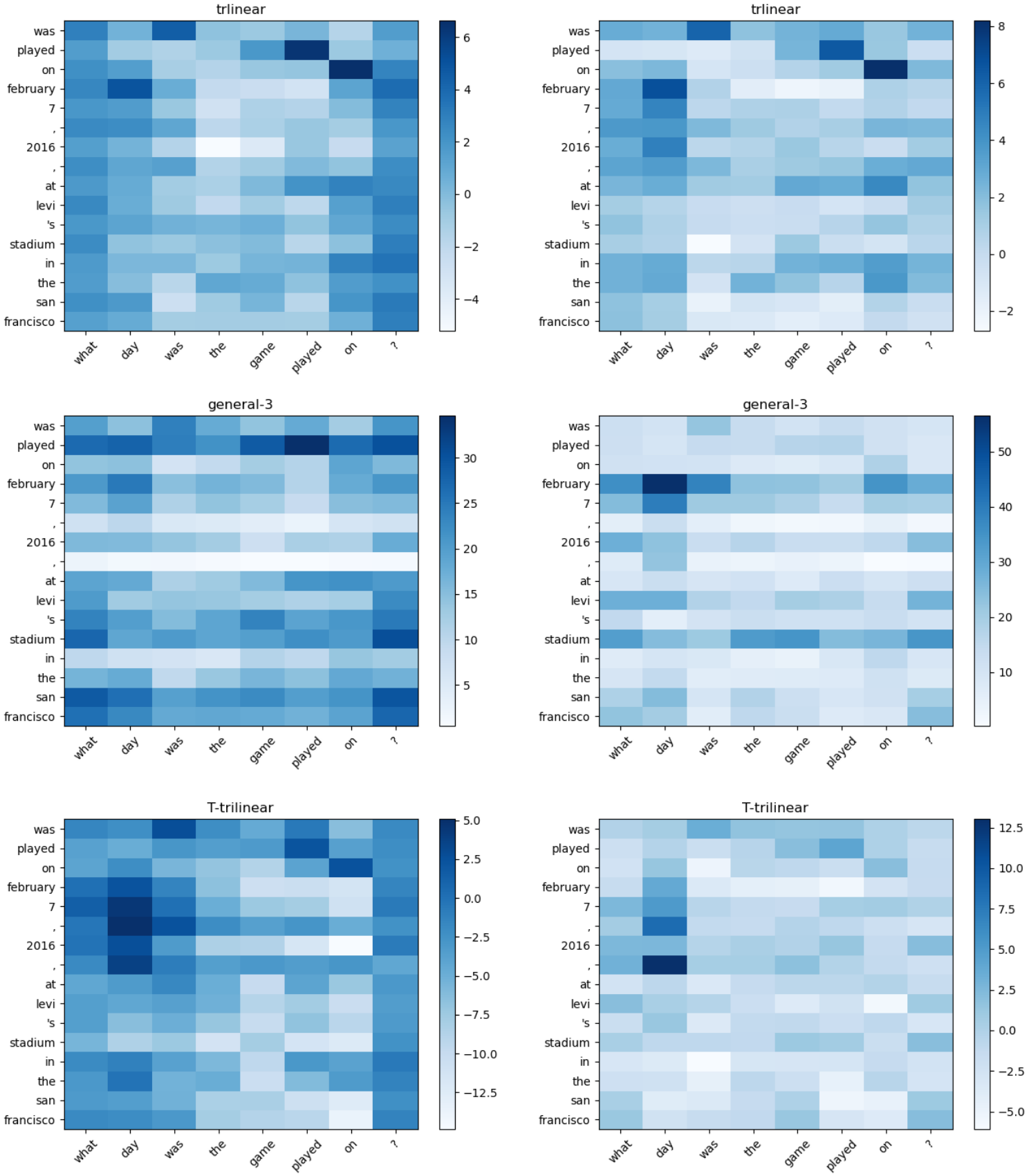
Question: What day was the game played on?

True answer: February 7, 2016

Figure 4.5: The chosen test sample of passage-question-answer

in the passage. However, the heatmap of the proposed T-trilinear function clearly has darker colour in these cells. This shows that the T-trilinear function is more able to pay attention to the right text span in the passage.

With the O_2 setting in Figure 4.6(b), trilinear's darker colour seems to be able to concentrate on a smaller area than in the O_1 setting. It shows slight darker on the answer word range "February 7, 2016", but still has relatively high focus on other words such as *was*, *played* and *on*, which are irrelevant to the question. The *general-3* function exhibits a very different heatmap with O_1. It shows high scores for the relevant words in the passage, especially "February 7", but the pattern still seems row-wise (passage-word-wise) like in O_1. The T-trilinear method can focus on the answer words clearly and pay less attention to words that are irrelevant to the answer except for the dot behind the answer span. This may be because part of the answer information is transferred to the adjacent position (the dot behind 2016) of the answer span through the RNNs in the model.



(a) In O_1 setting

(b) In O_2 setting

Figure 4.6: Heatmaps of attention similarity matrices

4.4 Summary

Attention mechanisms are prevalent techniques for neural network based NLP models in recent years. Attention similarity calculation is an important part of attention mechanisms. In this chapter, several similarity calculation functions that have been used in various NLP tasks are compared. They are tested in the QA context, using a QA baseline model designed in Chapter 3, benchmarked on the SQuAD dataset.

The experimental results demonstrate that the additive similarity function performs better than the multiplicative ones based on the baseline model using the span-based dataset SQuAD. As Section 3.3.3 indicates that the baseline model produces results that are commensurate with these original models, this discovery potentially applies to the RNN-based QA models using the same types of datasets. The trilinear similarity function in the additive group achieved the highest predictive scores. The general-3 function, applying the $ReLU$ operation on top of general-2, achieves the highest EM and F1 scores in the O_2 setting. As for the Hadamard similarity function, it achieved the highest predictive scores in the O_1 setting compared with those based on the inner product and has competitive scores as general-3 in O_2. However, its results are not consistent across multiple training runs in O_1. The two most commonly used functions – dot-product and scaled dot-product functions are among the worst performers.

Based on these results, a new function T-trilinear is proposed, which introduces the $ReLU$ transformation applied in general-3 to trilinear. Experimental results show that T-trilinear demonstrates the highest predictive scores as well as has stable performance across multiple runs. A heatmap visualization of the attention score matrix explains why this T-trilinear function is effective.

Chapter 5

Effects of Relevant Information Generation Methods

The second step of the attention mechanism is relevant information generation, as described in Section 2.2.3. Two early approaches are proposed as part of the BiDAF [27] and the DCN [28] QA models. Based on the similarity scores calculated, contextualized representations of the passage and question are computed. These representations are in the form of context-to-question summary (or simply question summary) and question-to-context summary (or passage summary). Relevant information is generated by aggregating these summaries and other information.

A common strategy to generate relevant information is by concatenation. The information that are being concatenated differ from one QA system to another. Liu et al. [26] proposed simply concatenating the passage representation and the question summary. Xiong et al. [28] and Li et al. [171] also included the passage summary into the concatenation beside the passage representation and the question summary. Some other QA systems concatenate not only these three inputs, but also their element-wise

products [27], [29], [182]. Instead of using the concatenated vectors directly, it can serve as an input to a feedforward network (FNN) [124]. While any arbitrary information generation methods can be utilized, it is not clear how they will affect the performance of the systems.

In this Chapter, a study of the relative effectiveness of various concatenation-based relevant information generation methods is presented. This study utilize the baseline model designed in Chapter 3. The insights gained through this study allow two new methods to be formulated. The first method includes additional terms into the concatenation. In the second method, the concatenated information is further processed by a feedforward network (FNN) with the $ReLU$ activation function. Results show that both these methods outperform all the existing ones, with the second one achieving the highest predictive scores.

5.1 Relevant Information Generation

As discussed in Section 2.2.3, relevant information is generated as the output of the attention mechanism. There are two steps involved. The first one is to generate the context-aware summaries by using the similarity scores, and the second is to format the output by aggregating the summaries along with other information.

5.1.1 Summary Generation

Two summaries are to be generated, which are query summary $\widetilde{\boldsymbol{U}}$ and passage summary $\widetilde{\boldsymbol{H}}$. The former aims to contain the relevant passage information based on the question representations, and the latter the relevant question information based on the passage

information. Recall that the input of the the attention mechanism is the contextualized passage representations $\boldsymbol{H}$ and the contextualized question representations $\boldsymbol{U}$.

5.1.1.1 Query Summary

The query summary, also knowns as context-to-query summary, is computed in the same way across different QA models [27], [29], [172]. It is given by

$$\widetilde{\boldsymbol{U}} = A\boldsymbol{U}, \tag{5.1}$$

where $\widetilde{\boldsymbol{U}} = \{\widetilde{\boldsymbol{u}}_i\}_{i=1}^{M} \in \mathbb{R}^{M \times d}$ and A is the attention weight matrix in Equation 3.10.

5.1.1.2 Passage Summary

The passage summary, also known as query-to-context summary, is utilized in some neural QA models [27], [28], [171]. There are two ways of generating this summary in the literature. One is to make use of the similarity score matrix once, denoted as first-order in this thesis. The other method utilises the similarity score matrix twice, which is called second-order in this work.

The First-order method

In the first-order method, the similarity score matrix is turned to be a vector $\boldsymbol{b}$ by choosing the maximum values from each row [27], [124]. Then the softmax operation is applied to this vector to compute the probabilities. The mathematical calculation is expressed as

$$b_i = max\left(S_{i:}\right), \tag{5.2}$$

$$\beta_i = \frac{\exp(b_i)}{\sum_{j=1}^{N} \exp(b_i)}, \tag{5.3}$$

where $S_{i:}$ denotes the i^{th} row of the matrix S, and b_i is the i^{th} element is the vector b.

The passage representations are weighted by the probability vector β and summed to produce the passage summary vector $\widetilde{\boldsymbol{h}}$:

$$\widetilde{\boldsymbol{h}} = \sum_{i=1}^{M} \beta_i \boldsymbol{H}_{:i}. \tag{5.4}$$

where $\boldsymbol{H}_{:i}$ is the i^{th} column of $\boldsymbol{H}$. $\widetilde{\boldsymbol{h}}$ is repeated M times in the row direction to produce the summary matrix $\widetilde{\boldsymbol{H}} = \left\{\widetilde{\boldsymbol{h}}_i\right\}_{i=1}^{M}$.

The Second-order method

In the second-order method, the attention similarity matrix is leveraged twice [28], [29]. First, the similarity matrix is normalized with the softmax operation along each column to produce the question-to-passage attention weight matrix

$$A^H = \left\{a_{ij}^H\right\}_{i=1\ldots M}^{j=1\ldots N}, \tag{5.5}$$

where each column $a_{:j}^H$ indicates how much a word in the question is relevant to each word in the passage. The passage representations are multiplied with A^H to calculate the query summary $\widetilde{\boldsymbol{H}} = \left\{\widetilde{\boldsymbol{h}}_j\right\}_{j=1}^{N}$ where

$$\widetilde{\boldsymbol{h}}_j = \sum_{i=1}^{M} {a^H}_{ij} \boldsymbol{h}_i. \tag{5.6}$$

The similarity matrix is then normalized with the softmax operation along each row to produce the passage-to-query attention weight matrix $A^U = \left\{a_{ij}^U\right\}_{i=1\ldots M}^{j=1\ldots N}$ where each row ${a^U}_{:j}$ indicates the relevance of a word in the passage to each word in

Table 5.1: Output formation methods

Method	Expression
RIG_1	$[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i]$ [26]
RIG_2	$\left[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i;\ \widetilde{\boldsymbol{h}}_i\right]$ [28], [171]
RIG_3	$[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i; \boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i]$ [123]
RIG_4	$\left[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i;\ \boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i;\ \boldsymbol{h}_i \circ \widetilde{\boldsymbol{h}}_i\right]$ [27], [29]
RIG_5	$\left[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i;\ \boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i;\ \widetilde{\boldsymbol{h}}_i \circ \widetilde{\boldsymbol{u}}_i\right]$ [124]
RIG_6	$FNN_{ReLU}\left(\left[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i; \boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i; \widetilde{\boldsymbol{h}}_i \circ \widetilde{\boldsymbol{u}}_i\right]\right)$ [124]

FNN_{ReLU} denotes a feedforward neural network with a $ReLU$ activation function

the query. The query summary $\widetilde{\boldsymbol{H}}$ is then multiplied by A^U to generate the query-aware passage summary $\widetilde{\boldsymbol{H}} = \left\{\widetilde{\boldsymbol{h}}_i\right\}_{i=1}^{M}$ where

$$\widetilde{\boldsymbol{h}}_i = \sum_{j=1}^{N} a_{ij}^{U} \widetilde{\boldsymbol{h}}_j. \tag{5.7}$$

5.1.2 Output Formation

After obtaining the summaries above, the output of the attention mechanism is to be generated. The output is denoted by $\boldsymbol{\mathcal{O}} = o_i{}_{k=1}^{M} \in \mathbb{R}^{M \times d_{\mathcal{O}}}$ where

$$\boldsymbol{o}_i = f(\boldsymbol{h}_i,\ \widetilde{\boldsymbol{u}}_i,\ \widetilde{\boldsymbol{h}}_i). \tag{5.8}$$

Here, $f()$ is the function that maps the input vectors into the output vector. Some QA models do not generate a passage summary $\widetilde{\boldsymbol{h}}_i$. For such models, Equation (5.8) is reduced to $o_i = f(\boldsymbol{h}_i,\ \widetilde{\boldsymbol{u}}_i)$. Table 5.1 lists the six different concatenation-based functions.

5.2 Comparisons of Output Formation Methods

Using the baseline model, the effects of five pure concatenation based relevant information generation methods can be fairly evaluated. These five methods are listed in Table 5.1. The dataset and the evaluation metrics used for the experiments here are the same as those described in Section 3.3.1. The training configurations are the same as those introduced in Section 3.3.2.

The best EM and F1 scores obtained from five training runs are shown in Table 5.2. Figure 5.1 are scatter plots showing pair-wise comparisons. The second-order method of passage summary generation is chosen to calculate $\widetilde{\boldsymbol{h}}_i$ as it gives better performances than the first-order methods. Details comparing results using first and second order methods are provided in Appendix A.

Table 5.2: EM and F1 scores of output formation methods RIG_1 to RIG_5

Method	Expression	EM	F1
RIG_1	$[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i]$	64.4	75.0
RIG_2	$\left[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i; \widetilde{\boldsymbol{h}}_i\right]$	64.9	75.4
RIG_3	$[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i; \boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i]$	65.1	75.5
RIG_4	$\left[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i; \boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i; \boldsymbol{h}_i \circ \widetilde{\boldsymbol{h}}_i\right]$	65.6	76.3
RIG_5	$\left[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i; \boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i; \widetilde{\boldsymbol{h}}_i \circ \widetilde{\boldsymbol{u}}_i\right]$	65.8	76.4

The following observations can be made from the pair-wise comparison results:

1. Figure 5.1(a) shows that RIG_2 performs better than RIG_1 by 0.7 point for the EM score. The corresponding gain in the F1 score is 0.8. This indicates that the term $\boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i$ contains useful information which is missing from $[\boldsymbol{h}_i,\ \widetilde{\boldsymbol{u}}_i]$. Thus,

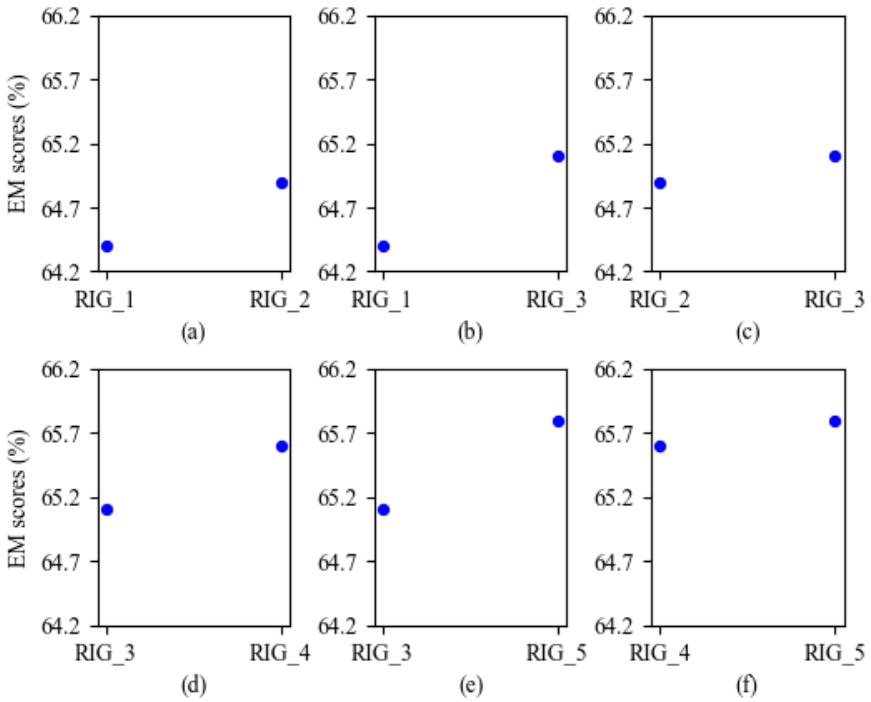


Figure 5.1: Scatter plots of Pair-wise comparisons of EM scores

including $\boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i$ into the concatenation $[\boldsymbol{h}_i,\ \widetilde{\boldsymbol{u}}_i]$ can improve the performance of the model.

2. Figure 5.1(b) shows that RIG_3 outperforms RIG_1 by 0.8 points for the EM score. For F1 score, RIG_3 is better by 0.9. This suggests that $\widetilde{\boldsymbol{h}}_i$ contains information which is not in $[\boldsymbol{h}_i,\ \widetilde{\boldsymbol{u}}_i]$. Therefore, incorporating $\widetilde{\boldsymbol{h}}_i$ into $[\boldsymbol{h}_i,\ \widetilde{\boldsymbol{u}}_i]$ should improve the performance of the model.

3. The results of RIG_2 and RIG_3 show that RIG_3 performs marginally better. This indicates that $\boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i$ is a better option than $\widetilde{\boldsymbol{h}}_i$.

4. Comparing RIG_3 and RIG_4 demonstrated show that RIG_4 performs better than RIG_3. This implies that the term $\widetilde{\boldsymbol{h}}_i \circ \widetilde{\boldsymbol{u}}_i$, contained in RIG_4 but not in RIG_3, plays an important role in improving the performance of the model.

5. Figure 5.1(e) shows that RIG_5 performs much higher than RIG_3. It indicates that the term $\widetilde{\boldsymbol{h}}_i \circ \widetilde{\boldsymbol{u}}_i$, contained in RIG_5 but not in RIG_3, carries more meaningful information.

6. With RIG_5 performing better than RIG_4 as shown in Figure 5.1(f), it would indicate that the term $\widetilde{\boldsymbol{h}}_i \circ \widetilde{\boldsymbol{u}}_i$ contains more relevant information than $\boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i$.

In summary, the two terms $\widetilde{\boldsymbol{h}}_i$ and $\boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i$ contain the useful information that is not held in the concatenation $[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i]$. Including $\boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i$ produces slightly better performance than including $\widetilde{\boldsymbol{h}}_i$, so $\widetilde{\boldsymbol{h}}_i$ can be replaced by $\boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i$. $\boldsymbol{h}_i \circ \widetilde{\boldsymbol{h}}_i$ and $\widetilde{\boldsymbol{h}}_i \circ \widetilde{\boldsymbol{u}}_i$ contain the information that is missing in $[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i;\ \boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i]$. Including $\widetilde{\boldsymbol{h}}_i \circ \widetilde{\boldsymbol{u}}_i$ helps the model perform better than including $\boldsymbol{h}_i \circ \widetilde{\boldsymbol{h}}_i$. Therefore, $\widetilde{\boldsymbol{h}}_i \circ \widetilde{\boldsymbol{u}}_i$ is a slightly better replacement of $\boldsymbol{h}_i \circ \widetilde{\boldsymbol{h}}_i$. As a result, the concatenation $\left[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i; \boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i; \widetilde{\boldsymbol{h}}_i \circ \widetilde{\boldsymbol{u}}_i\right]$ is the best combination.

5.3 Proposing A New Method

In seeking a method to improve upon the concatenation $\left[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i; \boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i; \widetilde{\boldsymbol{h}}_i \circ \widetilde{\boldsymbol{u}}_i\right]$, inspiration came from a neural QA system called DMN+ [32], which will be presented in Chapter 6. This system makes use of two terms: $|\boldsymbol{h}_i - \widetilde{\boldsymbol{u}}_i|$ and $\left|\widetilde{\boldsymbol{h}}_i - \widetilde{\boldsymbol{u}}_i\right|$ in its attention gates calculations. It will be interesting to find out if the absolute values of these differences provide additional information that is not captured in the concatenation methods. The addition of these two terms gives the following output formation:

$$\boldsymbol{o}_i = \left[\boldsymbol{h}_i; \widetilde{\boldsymbol{u}}_i; \boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i; \widetilde{\boldsymbol{h}}_i \circ \widetilde{\boldsymbol{u}}_i; \left|\boldsymbol{h}_i - \widetilde{\boldsymbol{u}}_i\right|; \left|\widetilde{\boldsymbol{h}}_i - \widetilde{\boldsymbol{u}}_i\right|\right]. \tag{5.9}$$

The highest predictive scores across five training runs are obtained using this

proposed method. The results in Table 5.3 shows that it performs better than RIG_5, the best among the five methods. Its performance is similar to that obtained with RIG_6 which has the addition of a $ReLU$ FNN. Thus, this suggests that these two terms capture additional relevant information that is not contained in the other terms.

Table 5.3: EM and F1 scores of different output formation methods

Method	**Expression**	**EM**	**F1**
RIG_5	$\left[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i; \boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i; \widetilde{\boldsymbol{h}}_i \circ \widetilde{\boldsymbol{u}}_i\right]$	65.8	76.4
RIG_6	$FNN_{ReLU}\left(\left[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i; \boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i; \widetilde{\boldsymbol{h}}_i \circ \widetilde{\boldsymbol{u}}_i\right]\right)$	66.3	76.9
Proposed New Method	$\left[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{u}}_i; \boldsymbol{h}_i \circ \widetilde{\boldsymbol{u}}_i; \widetilde{\boldsymbol{h}}_i \circ \widetilde{\boldsymbol{u}}_i; \lvert\boldsymbol{h}_i - \widetilde{\boldsymbol{u}}_i\rvert; \lvert\widetilde{\boldsymbol{h}}_i - \widetilde{\boldsymbol{u}}_i\rvert\right]$	66.5	76.9

Note that the only difference between RIG_5 and RIG_6, is that the concatenated output is put through an FNN. This indicates that applying an FNN with the $ReLU$ activation helps to improve the performance of the model. This suggests that the performance of the new method could be further improved if an FNN is used to process its outputs. Therefore, the new method is revised to

$$\boldsymbol{o}_i = ReLU\left(\boldsymbol{w} b_i + b\right), \tag{5.10}$$

where b_i is obtained from Equation (5.9), $\boldsymbol{w} \in \mathbb{R}^{6d \times d^*}$ and $b \in \mathbb{R}$. The output dimension d^* can be chosen flexibly, such as $6d$ or $4d$, where d is the dimension of $\boldsymbol{h}_i$ and $\widetilde{\boldsymbol{u}}_i$.

Table 5.4 shows the highest predictive scores across five training runs for the proposed methods in comparison with RIG_6. It is apparent that the revised new method achieves the highest scores than proposal-1. This confirms that applying an FNN with the ReLU activation helps to improve the performance of the model.

The loss curves, which show the training convergence trend, are plotted in

Table 5.4: EM and F1 scores of the proposed new methods

Method	EM	F1
RIG_6	66.3	76.9
Proposed New Method	66.5	76.9
Revised New Method	**66.9**	**77.5**

The dimension of the output of the FNN in the revised new method is the same as the input.

Figure 5.2. It can be observed that the proposed new methods converge faster than all the other methods. RIG_1 and RIG_3 are the slowest.

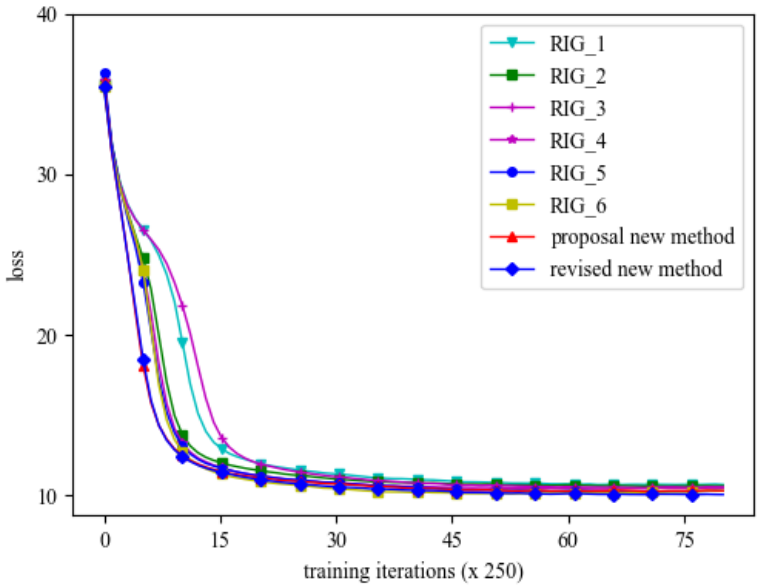


Figure 5.2: The loss curves of different methods

5.3.1 Effects of FNN Output Dimension

As the revised new method involves an FNN, it is important to investigate the effect of the output dimension. In fact, this effect has not been discussed in related publications.

The dimension of the input of the FNN is $6d$. Six different output to input ratios, from 1:1 and gradually reduced to 1:6, are considered. The results are tabulated in Table 5.5, and plotted as graphs in Figure 5.3.

Table 5.5: EM and F1 scores of the revised new method with different output dimensions

Output:Input Ratio	**Output dimension**	**EM**	**F1**
1:1	6d	66.9	77.5
5:6	5d	66.8	77.5
2:3	4d	67.0	77.5
1:2	3d	66.7	77.4
1:3	2d	66.2	76.9
1:6	d	65.8	76.8

The input dimension is $6d$.

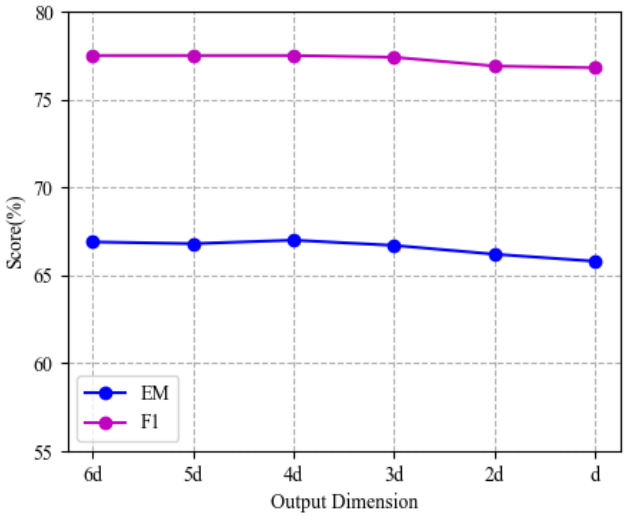


Figure 5.3: EM and F1 scores of the revised new method with different output dimensions

It can be seen that the EM and F1 scores do not exhibit much change as the output dimension changed from $6d$ to $3d$. However, as the output dimension drops

further, EM score drops significantly. This indicates reducing the output dimension to one third of the input dimension affects the predictive accuracy of the exactly matching answers. Since the F1 scores decrease only slightly, the model can still predict the answers that share some common words with the ground truth answers. Therefore, compressing the input representation to a vector with a length less than one third of the input dimension is undesirable. It may be because this causes a loss of useful information in the output vector.

5.4 Summary

Relevant information generation is the second and important part of modular attention mechanisms. In this Chapter several generation methods that have been used in various neural QA models are compared. They are tested in the baseline model developed in Chapter 3, benchmarked on the SQuAD dataset.

The experimental results show that including the element-wise products in RIG_3, RIG_4 and RIG_5 into the output concatenation helps the model achieve better results. On top of this, applying a FNN with the ReLU activation to the concatenation can further boost the performance.

Based on these results, a new method is proposed by including two additional terms into the best existing pure concatenation-based method. Experimental results show that it outperforms the best EM and F1 scores among the pure concatenation methods. This proposed method is further revised by applying an FNN with the ReLU activation. Experimental results demonstrate that the revised version shows the best performance among all the compared methods. In addition, an investigation on the effects of different ratios of the output dimension of this FNN to its input dimension is

carried out. The results suggest that setting this ratio to be between 1:1 and 1:2 helps the model to achieve better performance than the ratio lower than 1:2.

Chapter 6

The Roles of RNN in Memory Networks

As Section 2.2.2 describes that some RNN-based QA systems utilize the attention to do multiple computational hops, this type of QA systems target the tasks that require multi-step reasoning. They are usually known as memory-augmented networks, as the multiple computational hops are wrapped in a memory module. This type of QA networks mainly make use of RNNs to perform two roles. One is for producing the contextual representations in the attention mechanism, and the other is to generate sentence representations in the input module. In this Chapter, the effects of these two roles of RNNs on the capabilities of the QA systems are investigated. The DMN+ QA model is used for this part of the research. Details of this and related models are described in Sections 6.1 and 6.2.

It has been noted in Section 2.2.3 that RNNs can be used to produce relevant information vectors in the attention mechanism. This type of attention is called *gated attention*. It is developed to tackle a special type of tasks in QA called multi-hop

reasoning, as reviewed in Section 2.3.2. QA systems with gated attention are able to achieve very good accuracies on the tasks that requires sequential and deductive reasoning skills in multi-step reasoning, using benchmark datasets such as the bAbI dataset described in Section 2.3.2. Out of the 20 different tasks in bAbI, the only one that these systems perform poorly is Task-16 which involves inductive reasoning. The prediction accuracy is typically only around 50%. In this task, the order of the relevant facts in the story is not important, which means reordering these supporting sentences will not change the correct answer. This raises the question whether the presence of RNNs in the system may actually adversely affect the performance of the system for such tasks.

As will be shown in Section 6.3, the order of the supporting facts in the training dataset does influence the predictive performance of the trained system. Two methods are proposed to overcome this problem. The first one is through data augmentation. The second one is by replacing the RNNs, in the form of GRU, from the attention mechanism in the episodic memory module to one without RNN. The resulting model with an RNN-free attention mechanism is shown to be able to resolve the fact order preserving phenomenon and improve the generalization capability of the model effectively. This will be discussed in detail in Section 6.4.

With the above modification, the only remaining RNN in the model is the one present in the input module. By replacing this RNN by a feedforward network, the accuracy for the induction task was found to improve significantly compared with DMN+ and the other existing QA systems. For other non-sequential tasks, the results indicate that the performance of this modified model is very similar to other RNN-based systems. Details are presented in Section 6.5. By combining RNN-free and RNN-based models as an ensemble system, good results could be achieved for all 20 tasks in the bAbI dataset.

6.1 Review of Related QA Networks

The precursor of the neural QA systems tackling multi-hop reasoning tasks is the memory networks (MemNN) [153]. MemNN is the first to combine inference with a memory component to tackle the multi-hop reasoning question answering tasks. In the memory component, the memory can be updated from one computational hop to another. This memory mechanism plays an important role of inspiring the subsequent QA models. MemNN employs an RNN to predict the textual responses with being fed the sequence of question and supporting memories. MemNN requires not only the input and desired output but also the additional labels for the intermediate stage in the model during the training process, hence it cannot be trained in an end-to-end setting.

Shortly later an end-to-end version of MemNN was proposed which is called MemN2N [31]. This model does not require more labelled information than the input and output. In each computational hop, the attention weighted sentence representations are passed into an RNN to generate an internal context vector. The application of the RNNs in MemN2N plays an important role of inspiring subsequent QA systems, such as dynamic memory networks (DMN) [20]. However, DMN was still trained in the same fashion as MemNN that requires additional labels. Hence, it is not suitable to be used in real scenarios.

The end-to-end models was populated with the proposal of the DMN+ QA system [32]. DMN+ fully makes use of RNNs for not only the attention mechanism, but also the sentence encoding in the input module and the question encoding. It achieves lower error than its competitor the MemN2N model. Since then several such end-to-end neural QA models, including t-MEM-NN [21] and EnDMN [125], have been proposed. The t-MEM-NN model imposes Students'tt-distribution to the priors of the parameters of the MeMN2N model [31]. EnDMN focuses on the question module by incorporating

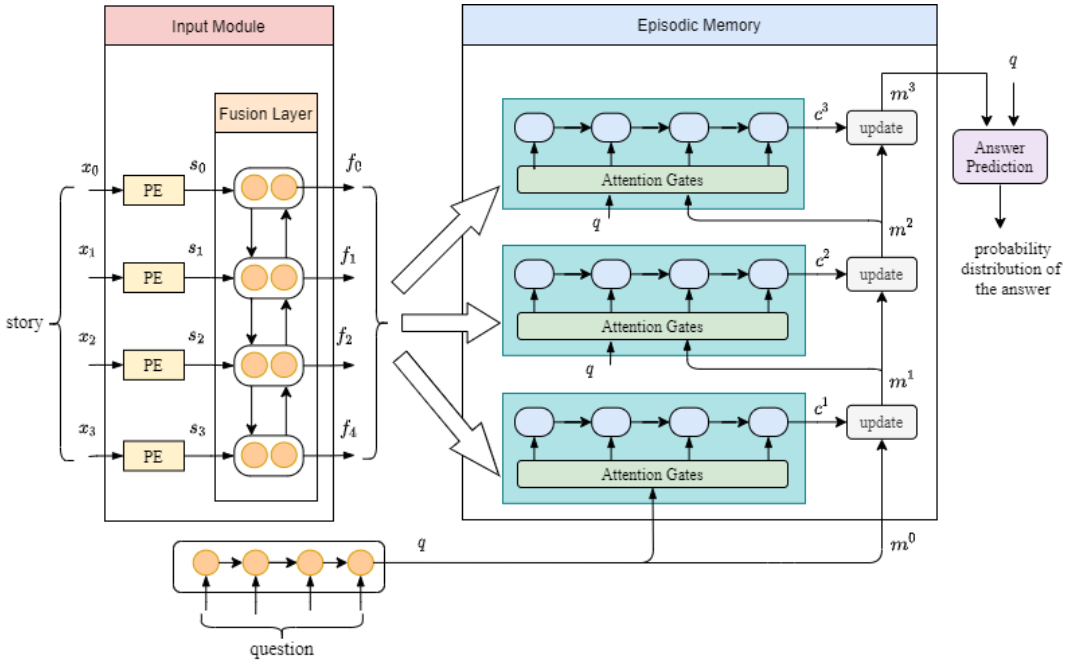


Figure 6.1: The block diagram of the DMN+ QA system

a gated RNN to create the global and salient features for the question [125]. The RNNs in these systems play an important role in learning sequential and contextual information that are required for the multi-hop reasoning tasks.

None of the aforementioned end-to-end neural QA models are able to solve the inductive reasoning tasks satisfactorily. As DMN+ is among the first to make fully use of RNNs and has inspired several subsequent end-to-end RNN-based QA models, it is chosen to be a case study.

6.2 The DMN+ QA System

The architecture of the DMN+ system is illustrated in Figure 6.1. All the words in stories and questions are converted into the high dimensional space using word embeddings

$\{w_i\}_{i=1}^{v} \in \mathbb{R}^d$ before being fed into the model, where d denotes the dimension of the word embeddings and v is the size of the vocabulary.

6.2.1 Input Module

The input module aims to produce the sentence vectors by incorporating the positional information of words within each sentence through the positional encoder (PE), and the sequential information of sentences within a story through a bi-directional RNN. The vector x_t represents the sequence of word embeddings $\left\{w_k^{(t)}\right\}_{k=0}^{K} \in \mathbb{R}^d$ in the t^{th} sentence of the story, where K is the total number of words in the t^{th} sentence and d is the dimension of the embedding vector. x_t is encoded into an initial sentence vector s_t through a position encoder (PE) [153] in order to capture the positional information of each word within a sentence and incorporate it into s_t. More specifically, the PE is a weighting matrix $PE \in \mathbb{R}^{K \times d}$. The sentence vector is calculated by

$$s_t = \sum_{t=1}^{K} PE \circ x_t, \tag{6.1}$$

where $\circ$ is the element-wise multiplication. The element in the k^{th} row and j^{th} column in the matrix PE is calculated with this equation:

$$PE_{kj} = (1 - \frac{j+1}{J}) - \frac{k}{d}(1 - \frac{2(j+1)}{J}), \tag{6.2}$$

where J is the number of words in the sentence.

These positional encoded sentence vectors $\{s_t\} \in \mathbb{R}^d$ are fed into the context fusion layer, which is made of a bi-directional GRU consisting a forward and backward GRU. The outputs of the two unidirectional GRU are then passed through a summation operation to produce the output of the context fusion module $\{f_t\} \in \mathbb{R}^d$, which is the

contextualized sentence representations.

As a result, the information of the preceding and succeeding sentences is fused into the current sentence in the story, so that each sentence representation contains the context of the story.

6.2.2 Question Module

Different from the processing of the story, the question only consisting of one sequence of words is passed through a unidirectional GRU. The final sate of this GRU is taken as the question vector q. In this way, the contextual information of the question is encoded into q. This can be expressed as the equations below

$$h_i = GRU(h_{i-1}, q_i), \tag{6.3}$$

$$q = h_Q, \tag{6.4}$$

where Q denotes the total number of words in the question, as the result, q is the final hidden state of the GRU.

6.2.3 Episodic Memory Module

The episodic memory module aims to dynamically retrieve information from the memory and update it. This dynamic and iterative nature of this module allows it to take different facts at each hop to obtain the information for the next hop, and eventually to produce the relevant information for the answer module. The contextualized sentence representations $\{f_t\} \in \mathbb{R}^d$ form the input to the episodic memory. The episodic

memory module consists of multiple computational hops to retrieve information from the memory by paying attention to a subset of sentences. Each hop contains two components – the gated attention mechanism and and the episodic memory vector update layer. The gated attention consists of attention gates calculation and attention based GRU to generate the contextual vector. The attention mechanism takes the sentence representation, question vector and the current memory vector as the input and computes the attention gate $g_t^i \in \mathbb{R}$ that represents the relevant extent of each sentence in the current hop.

The attention gates $\{g_t^i\} \in \mathbb{R}$ are then used as the update gates in the standard GRU to form the attention based GRU, called AttnGRU. The sentence vectors $\{f_t\}$ are passed to the AttnGRU to obtain the final state as the contextual vector c^i. In this way, the positional and ordering information of the sentences as well as those sentences with relatively higher attention weights are preserved in c^i. This context representation is then passed to the memory update layer which is a FNN with the $ReLU$ activation to generate the memory m^i.

6.2.4 Answer Module

The episodic memory vector of the last hop m^I and the question vector q is used to predict the probability distribution of the answer:

$$P = softmax(W^{(a)}[q;\, m^I]), \tag{6.5}$$

where ; denotes the concatenation of q and m^I, I is the total number of the computational hops, and $W^{(a)}$ is a trainable vector. $P = \bigcup_j \{p_j : j \in \mathcal{J}\}$, where $\mathcal{J}$ represents the set of all the possible indices that can be taken, and p_j is the probability at the j^{th} possible

value.

At the training stage, the probability distribution is used to calculate the cross entropy loss function and the objective is to minimize it:

$$L(\theta) = -\frac{1}{K}\sum_{i=1}^{K}\sum_{j=1}^{J} y_j^{(i)} log(p_j^{(i)}), \tag{6.6}$$

where θ is the set of all the training weights, K is the number of training samples in a training batch, $y_j^{(i)}$ and $p_j^{(i)}$ denote the true probability and the predicted probability at the j^{th} possible value of the i^{th} training sample, respectively.

The answer to the question is the one with the largest probability in $P^{(i)}$. That is,

$$answer = \underset{p_j^{(i)} \epsilon P^{(i)}}{\arg\max}(P^{(i)}), \tag{6.7}$$

where $P^{(i)}$ is the probability distribution of the i^{th} testing sample.

6.3 Effects of Changing the Order of Facts

Experiments are conducted to study the effects of changing the order of facts on the performance of the DMN+ system. Details of the experimental setup and the results are described in this Section. A solution from the perspective of the training data, data augmentation, is proposed for tackling the issue discovered by the experimental results.

6.3.1 Dataset

The dataset used in the experiments is the bAbI dataset, created and released by Facebook [30]. This dataset is the first and representative multi-step reasoning QA benchmark, and has encouraged the development of many neural QA systems [20], [21], [32], [153]. It consists of 20 synthetic tasks and each task requires different reasoning abilities to answer the questions. Each task contains samples of [story, question, answer, indices of supporting facts]. As the models investigated and proposed in this Chapter are trained in the end-to-end fashion, the indices of supporting facts are not used. This dataset contains both English and Hindi languages. The English version (en-10k) is used, which has 10k training samples and 1k testing samples. The 10k training samples are split into an 8:2 ratio for training and validation purposes, respectively.

6.3.2 Training Configurations

The Adam optimizer [183] is used for training with a learning rate of 0.001. The Glorot uniform initialization [77] procedure is used for all the trainable weights. The word embeddings are randomly initialized with a uniform distribution in the range $[-\sqrt{3},\ \sqrt{3}]$ and trained along with the other weights. Three hops are used in the episodic memory to balance the predictive accuracy and the computational efficiency. In order to compare the performance of the systems for individual tasks, they are trained independently with the data for each task.

Table 6.1: Possible orders of 3 supporting facts

Order patterns	Orders of facts
1	F1, F2, F3
2	F1, F3, F2
3	F2, F1, F3
4	F2, F3, F1
5	F3, F1, F2
6	F3, F2, F1

6.3.3 Experimental Results

Task 16 in the bAbI dataset requires inductive reasoning ability of the model over three supporting facts to infer the answer to a question. The accuracy on this task achieved by neural QA systems is much lower than whose on the other tasks, thus, task 16 shall be focused. For the data in this task, manipulating the order of the supporting facts does not affect the answer. There are therefore 6 possible permutations of the order of the three supporting facts, as shown in Table 6.1. F1, F2 and F3 means the first fact needs to be found, the second and the third. Samples of the 6 patterns are shown in Figure 6.2. The training set is categorized into these 6 order patterns. There are a total of 1666 samples with each pattern, which are then divided into the training, validation and testing sets. 20% of the samples are used for the testing set and the rest samples are split with 9:1 for the training and validation sets.

A separate DMN+ network is trained with the samples of each of the 6 patterns respectively, resulting in 6 different trained models. Each of these 6 models is then tested with the test samples of all the 6 patterns. The test accuracies are shown in Table 6.2 and plotted graphically in Figure 6.3.

1 Greg is a frog. 2 Bernhard is a swan. 3 Julius is a frog. 4 Bernhard is white. 5 Julius is green. 6 Lily is a frog. 7 Brian is a frog. 8 Lily is gray. 9 Brian is gray. 10 What color is Greg? gray **1 7 9**	1 Brian is a rhino. 2 Lily is a rhino. 3 Julius is yellow. 4 Bernhard is a frog. 5 Greg is a lion. 6 Bernhard is white. 7 Julius is a lion. 8 Brian is green. 9 Greg is yellow. 10 What color is Lily? green **1 8 2**
1 Greg is a lion. 2 Greg is yellow. 3 Lily is a swan. 4 Julius is a swan. 5 Bernhard is a frog. 6 Bernhard is white. 7 Brian is a rhino. 8 Brian is gray. 9 Lily is white. 10 What color is Julius? white **4 3 9**	1 Greg is white. 2 Brian is a rhino. 3 Julius is gray. 4 Bernhard is green. 5 Greg is a swan. 6 Bernhard is a rhino. 7 Lily is a swan. 8 Brian is green. 9 Julius is a lion. 10 What color is Lily? white **5 7 1**
1 Greg is a lion. 2 Brian is a frog. 3 Brian is green. 4 Greg is white. 5 Bernhard is a swan. 6 Julius is a swan. 7 Lily is a lion. 8 Bernhard is gray. 9 Julius is gray. 10 What color is Lily? white **7 1 4**	1 Brian is a frog. 2 Julius is a frog. 3 Greg is a lion. 4 Greg is yellow. 5 Lily is a lion. 6 Brian is green. 7 Lily is yellow. 8 Bernhard is a swan. 9 Bernhard is gray. 10 What color is Julius? green **6 2 1**

Figure 6.2: Samples of the 6 patterns of the orders of facts

It is obvious that the accuracy of each trained network is highest for the pattern in which they are trained. For example, the highest accuracy for Network 1 is 73.3% for the test samples with pattern 1. But its accuracy is significantly lower for patterns 2, 4, and 6. This trend is the same for the other five trained networks. This shows that the trained models memorized the sequence of facts in the training dataset, and they do not generalize well to other orders of facts which are not present in the training data.

Table 6.2: The test accuracy (%) on each pattern of DMN+ trained on each pattern

Test pattern	Trained Network					
	1	2	3	4	5	6
1	73.3	43.3	65.7	34.7	43.7	38.0
2	51.3	68.0	55.0	55.7	67.3	51.7
3	73.0	46.0	78.3	38.3	41.7	37.0
4	38.3	44.7	30.0	70.0	42.0	67.7
5	54.7	67.3	46.7	53.3	64.7	48.7
6	36.0	45.3	28.7	69.3	42.7	72.0

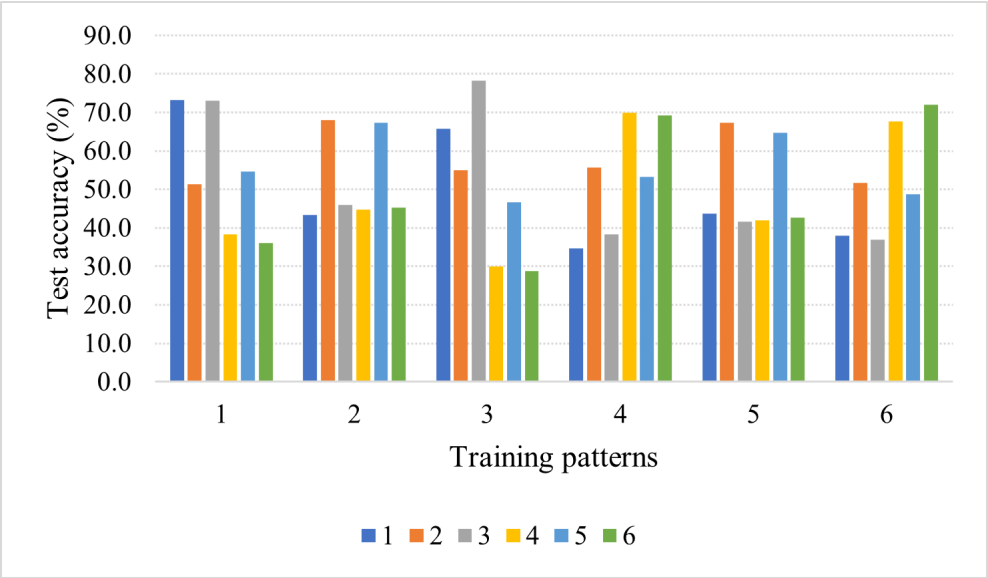


Figure 6.3: Graphical presentation of the results in Table 6.2

Table 6.3: The test accuracy (%) of DMN+ trained on different patterns

Test patterns	Trained Network						
	1	2	3	4	5	6	augmented
1	73.3	43.3	65.7	34.7	43.7	38.0	54.3
2	51.3	68.0	55.0	55.7	67.3	51.7	62.0
3	73.0	46.0	78.3	38.3	41.7	37.0	59.3
4	38.3	44.7	30.0	70.0	42.0	67.7	56.3
5	54.7	67.3	46.7	53.3	64.7	48.7	62.0
6	36.0	45.3	28.7	69.3	42.7	72.0	54.0

6.3.4 Data Augmentation Solution

Since the original training data may not present a full complement of the ordering of facts, a well-known technique in artificial neural networks is to supplement these data with augmented data. In this context, augmentation involves permutating the order of the sentences in the original dataset, keeping the answer unchanged. The network will then be trained by this larger, augmented dataset.

A network is trained on this augmented dataset and tested on each pattern. The results are shown in Table 6.3. For the convenience of comparison, the results in Table 6.2 are also included in this table. Figure 6.4 is a graphical presentation of the results in Table 6.3.

These results show that the network trained with augmented data performed more or less equally for all test patterns. This clearly demonstrates that the network memorizes the order of the facts in the training samples. That is why the network trained with augmented data is able to achieve the level of accuracies if it were trained

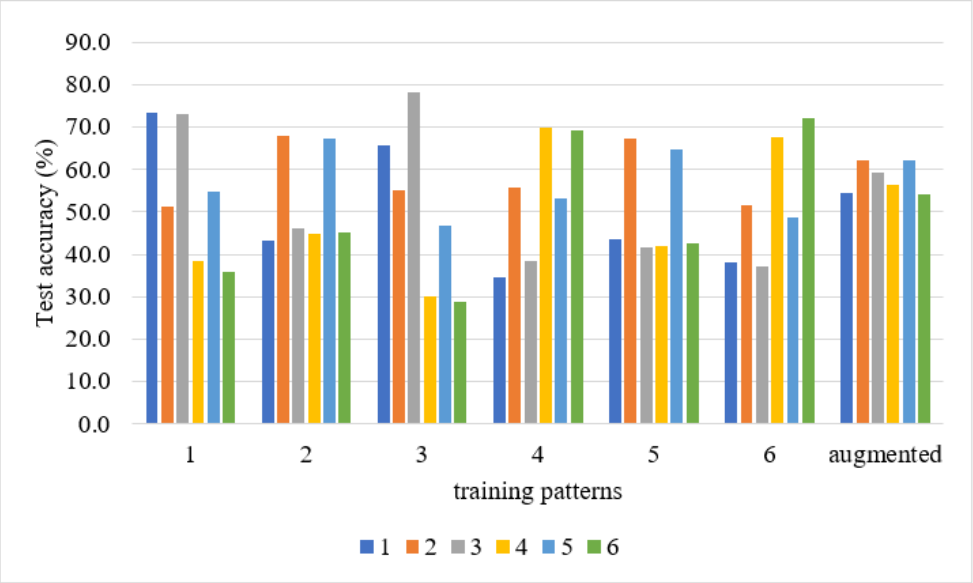


Figure 6.4: Graphical presentation of the results in Table 6.3

and tested with the same patterns.

6.4 RNNs in the Gated Attention

Apart from seeking the solution from the perspective of the data manipulation, another direction can be from the view of the structure of the model itself. As we know that RNNs preserve the ordering information of sequences and there are two main places in DMN+ that utilize RNNs to capture such information of sentences, one is the gated attention and the other is the fusion layer in the input module. The former is to be investigated in this Section, and the latter in Section 6.5.

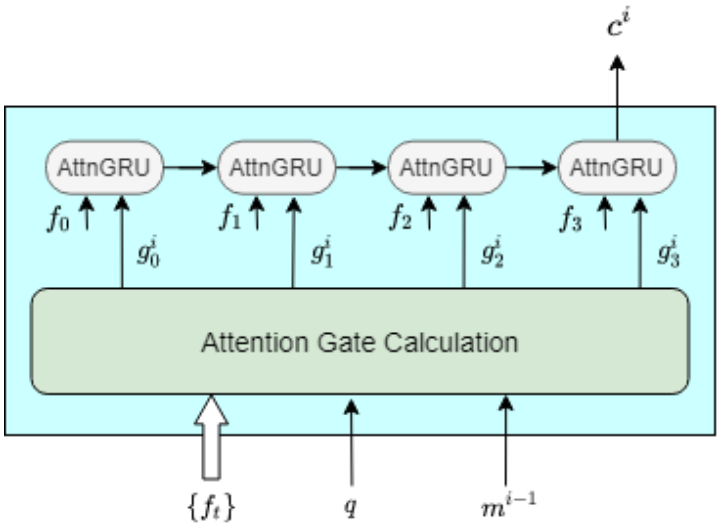


Figure 6.5: The gated attention mechanism in the episodic memory module

6.4.1 RNN-based Attention Mechanism in DMN+

The gated attention mechanism in the episodic memory module is responsible for generating the contextual vector c^i, which is then passed into the memory update layer to update the memory. Figure 6.5 demonstrates the details in this gated attention mechanism. the attention gates $\{g_t^i\}$ are calculated through a FNN followed by a softmax normalization as below

$$z_t^i(f_t, m^i, q) = [f_t \circ q; f_t \circ m^i; |f_t - q|; |f_t - m^i|], \tag{6.8}$$

$$\alpha_t^i = tanh(W^{(1)} z_t^i(f_t, m^i, q) + b^{(1)}), \tag{6.9}$$

$$g_t^i = softmax(W^{(2)} \alpha_t^i + b^{(2)}), \tag{6.10}$$

where m^i is the memory vector in hop i, $\circ$ represents the element-wise product, and $|\cdot|$ denotes the absolute value. $W^{(1)}$ and $W^{(2)}$ are the trainable weights in the FNN. $b^{(1)}$ and $b^{(2)}$ are the biases.

Recall that a standard GRU illustrated in the left side of Figure 6.6 has two gates, the update gate z_t and the reset gate r_t. In this case the input x_t is the sentence vector f_t. The calculation of the GRU cell is expressed as:

$$z_t = \sigma(W^{(z)} f_t + U^{(z)} h_{t-1} + b^{(z)}), \tag{6.11}$$

$$r_t = \sigma(W^{(r)} f_t + U^{(r)} h_{t-1} + b_{(r)}), \tag{6.12}$$

$$\widetilde{h}_t = tanh(W f_t + r_t U h_{t-1} + b), \tag{6.13}$$

$$h_t = z_t \widetilde{h}_t + (1 - z_t) h_{t-1}, \tag{6.14}$$

where $W^{(z)}$, $U^{(z)}$, $W^{(r)}$, $U^{(r)}$, W and U are trainable weights, and $b^{(z)}$, $b^{(r)}$ and b are the biases.

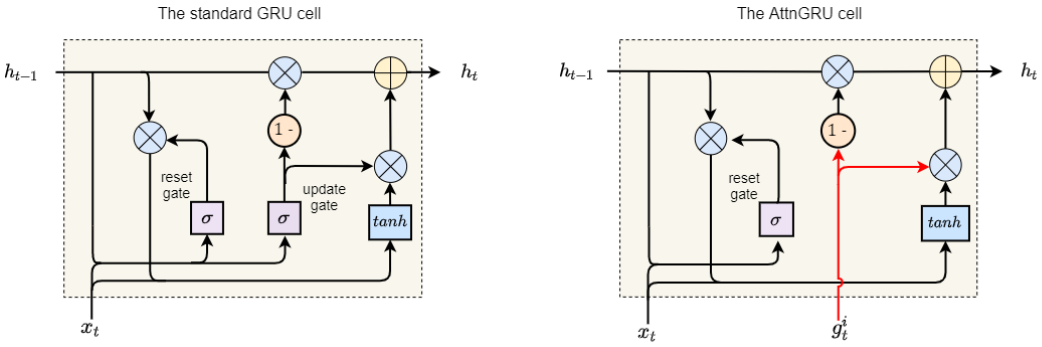


Figure 6.6: The comparison of a standard GRU cell and the AttnGRU cell

In the attention based GRU – AttnGRU, the update gate in a standard GRU is replaced by the attention gate g_t^i, which is illustrated in the right side of Figure 6.6. Thus, Equation (6.11) should be discarded and Equation (6.14) needs to be modified to the following one:

$$h_t = g_t^i \widetilde{h}_t + (1 - g_t^i) h_{t-1}. \tag{6.15}$$

In this way, the sentences with relatively higher attention gates along with their ordering information within the story are preserved in the final hidden state of the attention based GRU, which acts as the contextual vector c^i. The retaining of the ordering information

is very likely to cause the model to memorize the order of facts in the training data demonstrated in Section 6.3.3.

6.4.2 MoDMN+: DMN+ with a New Attention Mechanism

The RNN in the form of AttnGRU in the episodic memory plays an important role of preserving the ordering information of the sentences with relatively high attention in the story. This may cause the phenomenon shown in Section 6.3.3, which demonstrates that trained model cannot generalized well to the test samples with the orders of facts that did not appear in the training data. Therefore, an attention mechanism without using RNNs could potentially be an appropriate solution.

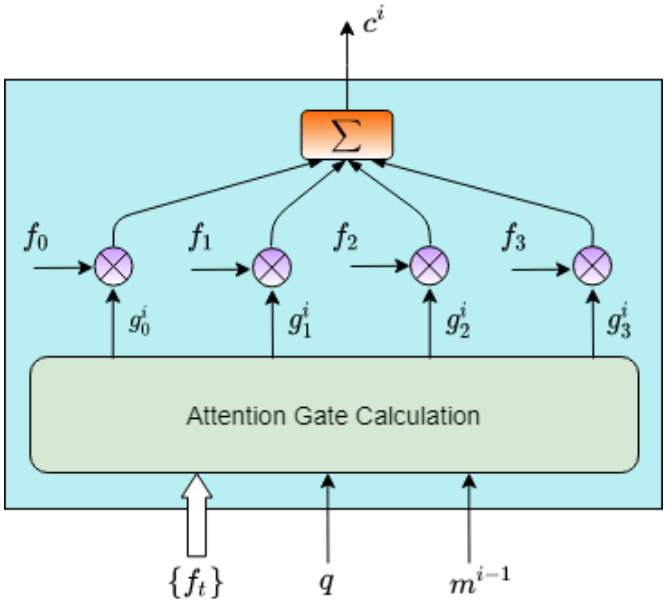


Figure 6.7: The proposed RNN-free attention mechanism

Here an RNN-free attention mechanism is designed by replacing the RNN in the gated attention with a RNN-free operation, as shown in Figure 6.7. The way of the attention gate calculation remains the same. The contextual vector c^i is generated

through a weighted summation operation in the following equation

$$c^i = \sum_{t=0}^{t=K} g^{it} f_t, \tag{6.16}$$

where $\sum_t g_t^i = 1$, and K is the total number of sentences in the story.

A new model called MoDMN+ is proposed by replacing the RNN-based gated attention with the new RNN-free attention. In MoDMN+, the information related to the order of sentences is not incorporated into the contextual vector c^i. Consequently, it should have a better generalization capability when the trained model performs on the test samples with different orders of facts.

6.4.3 Experimental Results

Six MoDMN+ networks are trained in the same way as in Section 6.3.3. The results for the 6 test patterns are shown in Table 6.4 and graphically illustrated in Figure 6.8. Comparing to the results shown in Table 6.2 in Section 6.3.3, the large variations in test accuracies across six different test patterns of the model trained on each pattern are significantly reduced.

For the convenience of comparison, six radar charts corresponding to six training patterns are plotted in Figure 6.9. Figure 6.9(a) depicts the models trained on the data of pattern 1 and test on data with all the six patterns, and Figure 6.9(b) trained on pattern 2, and likewise for the remaining radar charts. The six testing patterns (1, 2, 3, 4, 5, 6) are shown around each radar chart. The red lines represent the results of DMN+, and the blue lines depict those of MoDMN+. The circles from small (inside) to large (outside) denote the test accuracies 20%, 40%, 60%, and 80%.

Table 6.4: The test accuracy (%) on each pattern of DMN+ trained on different patterns

Test pattern	Order of facts	Trained pattern					
		1	2	3	4	5	6
1	F1, F2, F3	61.0	52.0	54.3	52.7	54.7	51.0
2	F1, F3, F2	48.3	66.3	48.7	48.3	65.3	45.0
3	F2, F1, F3	53.7	49.3	60.3	55.7	51.7	50.3
4	F2, F3, F1	53.0	49.0	60.0	53.0	56.0	53.0
5	F3, F1, F2	51.7	69.0	52.7	51.7	64.7	44.3
6	F3, F2, F1	50.0	49.7	53.3	52.0	43.0	56.3

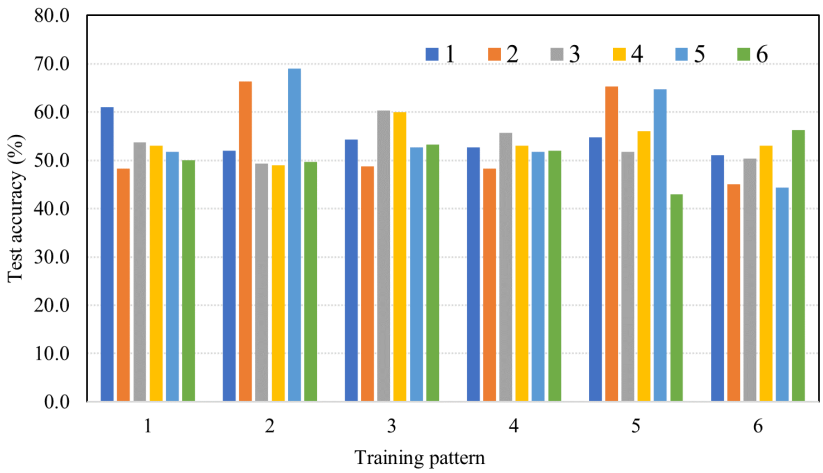


Figure 6.8: Graphical presentation of the results in Table 6.4

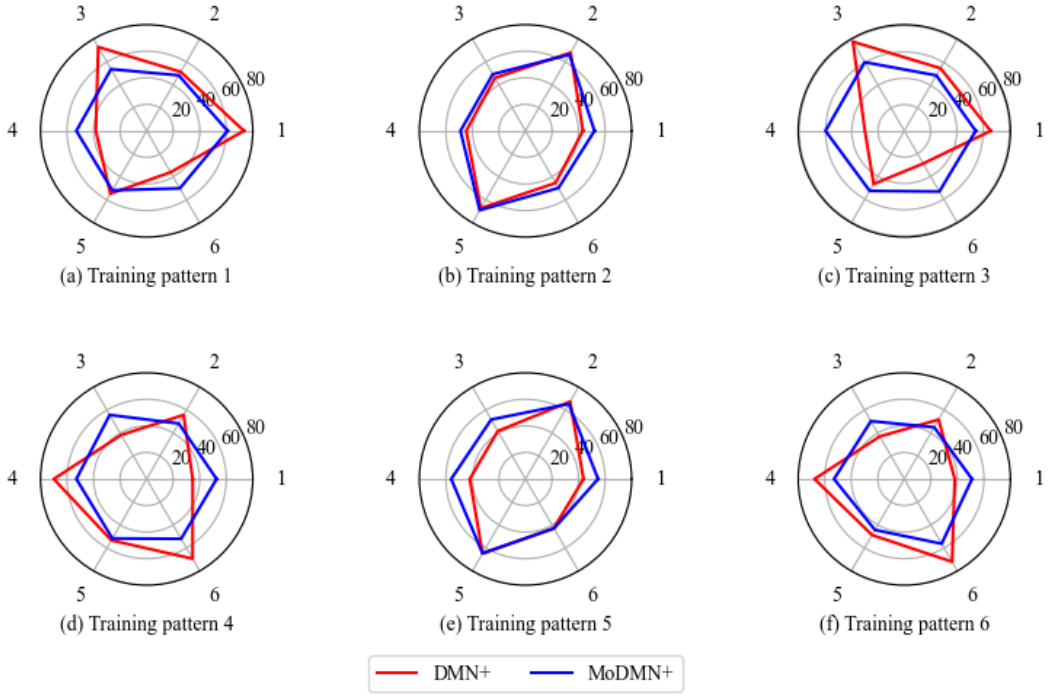


Figure 6.9: Radar chart comparing the testing accuracies of DMN+ and MoDMN+

All the charts in Figure 6.9 indicate that MoDMN+ model (blue lines) has more stable predictive accuracies across the six testing patterns than the DMN+ model. Take Figure 6.9 (d) for example, MoDMN+ trained on the training data of pattern 4 demonstrates approximately equal results for all the test patterns. However, DMN+ has very high results for test pattern 4 and 6 with accuracies 70.0% and 69.3% respectively, while much lower for test pattern 1 and 3 with accuracies 34.7% and 38.3% individually. The similar occurrence exhibits in all the other charts. This suggests that the RNN, in the form of AttnGRU, in the episodic memory module does play an important role in this fact-order preserving phenomenon. Replacing this AttnGRU with a non-recurrent calculation does help to improve the generalization capability of the model for the tasks where the order of facts is not relevant.

However, the test accuracies of the MoDMN+ model are still far lower than 95% the passing threshold. This indicates that some other factor(s) is affecting the model's performance on the inductive reasoning tasks.

6.5 RNNs in the Input Module

This section focuses on the effects of the RNNs in the input module, in the form of a bi-directional GRU. First, the RNN-based input module used by both DMN+ and MoDMN+ is described in detail in Section 6.5.1. Second, an RNN-free input module is proposed in Section 6.5.2. In the same Section a new QA model called ff-DMN is designed with the RNN-free input module. Section 6.5.3 reports the experimental results and analysis.

6.5.1 RNN-based Input Module

The input module shown in Figure 6.10 aims to produce the sentence representations by incorporating the positional information of words within a sentence and the sequential information of sentences within a story. The position encoder is employed to capture the order of words into the initial sentence vectors s_i. The fusion layer comprised of a bi-directional GRU is responsible for encoding the order of sentences into the contextualized sentence embeddings f_i.

The forward GRU expressed in Equation (6.17) captures the information of the preceding steps (sentences in this case) to blend into the current step, so that each sentence vector carries the preceding context. In the same way, the backward GRU in Equation (6.18) integrates the succeeding context into the representation of the current

sentence.

$$\overrightarrow{h}_t = \overrightarrow{GRU}(h_{t-1}, s_t), \tag{6.17}$$

$$\overleftarrow{h}_t = \overleftarrow{GRU}(h_{t+1}, s_t), \tag{6.18}$$

$$f_t = \overrightarrow{h}_t + \overleftarrow{h}_t. \tag{6.19}$$

Hence, the contextualized representation f_i of each sentence captures the ordering context of both the sentences before and those behind it. This may potentially affect the performance the model when it is trained on the data with specific order patterns and test on different patterns.

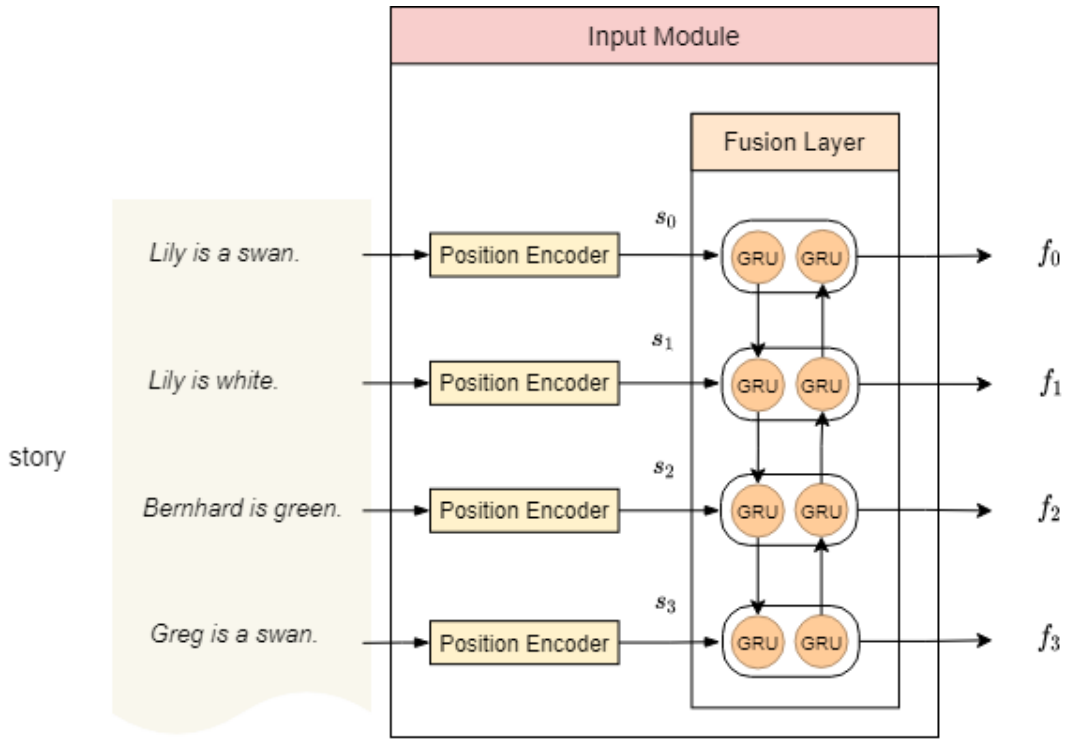


Figure 6.10: The RNN-based Input module in the DMN+ model

6.5.2 ff-DMN: MoDMN+ with a New Input Module

In order to eliminate the impact of the RNN in the input module, a RNN-free method is proposed to produce the sentence representations as illustrated in Figure 6.11.

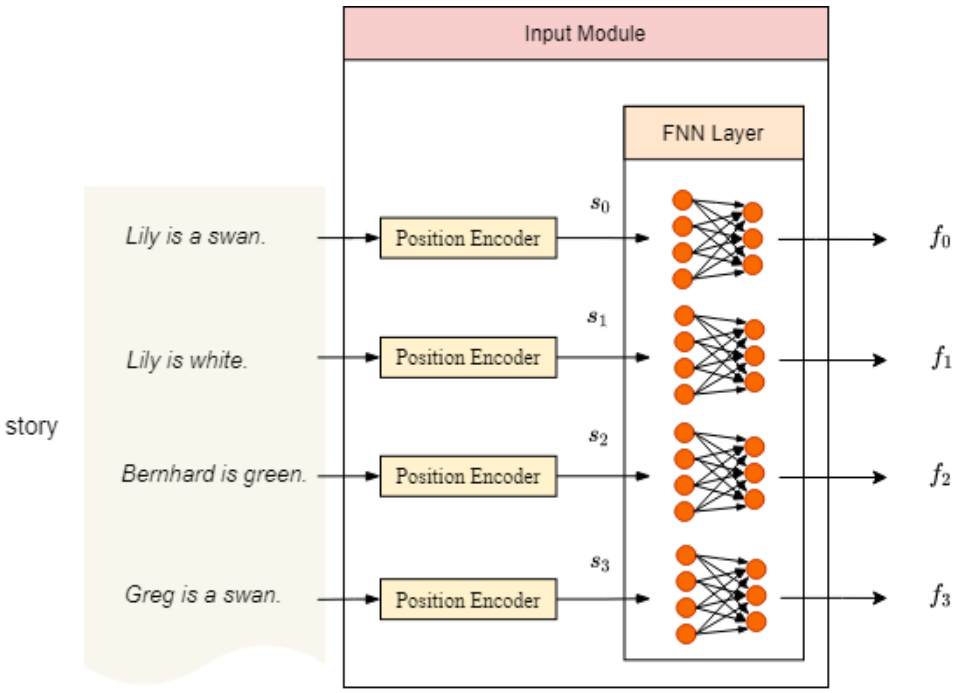


Figure 6.11: The proposed RNN-free input module

In this RNN-free input module, the bi-directional GRU is replaced with a feedforward neural network. Hence, the ordering information of the sentences in a story is not retained in the sentence representations. The question is also encoded by the position encoder as well followed by the feedforward network. By replacing the RNN-based input module from MoDMN+ with the RNN-free input module, the ff-DMN system is proposed. The overall block diagram of the ff-DMN model is shown in Figure 6.12. There is no RNNs in this model, instead only feedforward neural networks are utilised in the input module, question module, gate calculation and the answer prediction.

In the episodic memory module, the input of the update layer is only the

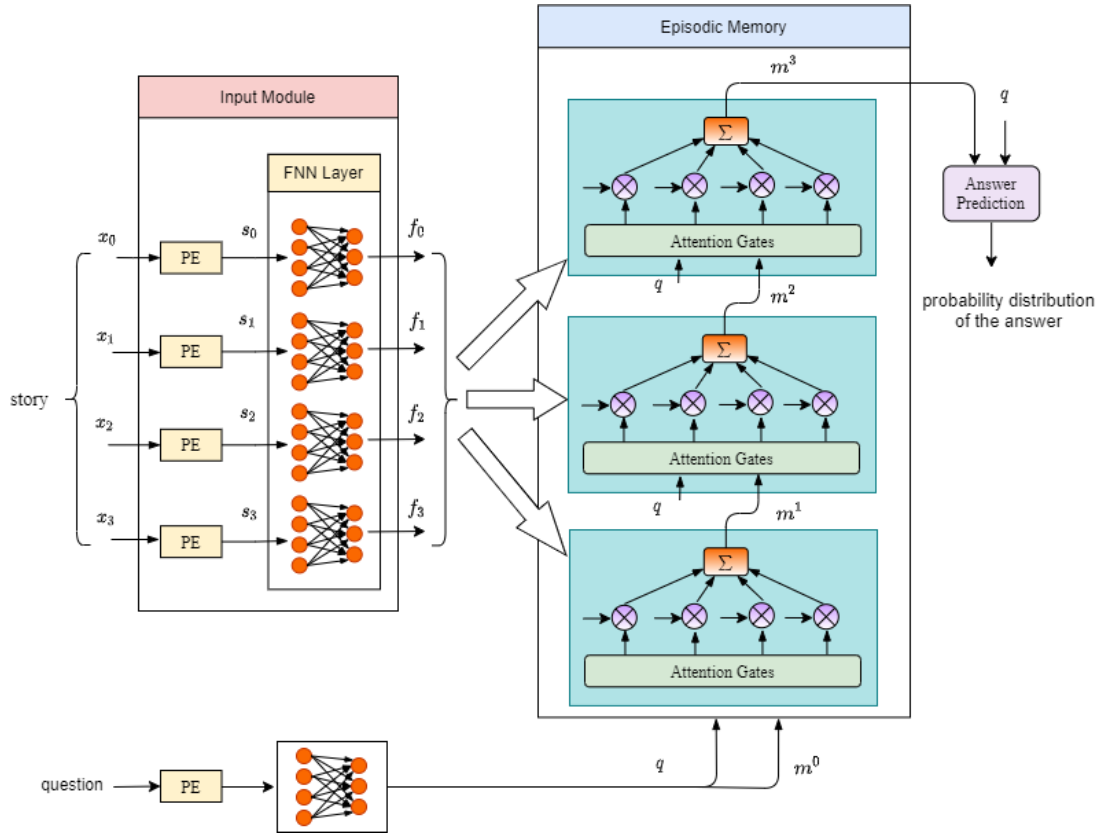


Figure 6.12: The block diagram of the proposed ff-DMN model

contextual vector without including the previous memory vector. This is because the previous memory already takes parts in the generation of contextual vector at each hop and including it into the update layer prevents the model to achieve good performance on some tasks according to our preliminary experiments.

6.5.3 Experimental Results

The performance of the proposed ff-DMN model on the inductive reasoning tasks is reported first, and then extend to the other non-sequential tasks.

6.5.3.1 Inductive Reasoning Task

The prediction accuracies of the trained ff-DMN and the other QA systems benchmarked on the induction task in bAbI dataset are shown in Figure 6.13, which clearly depicts that the proposed ff-DMN outperforms all the other systems significantly. It improves upon the basic LSTM by 74.2% and upon DMN+ by 42.5%. It suggests that incorporating RNNs may hinder the system to learn to reason inductively.

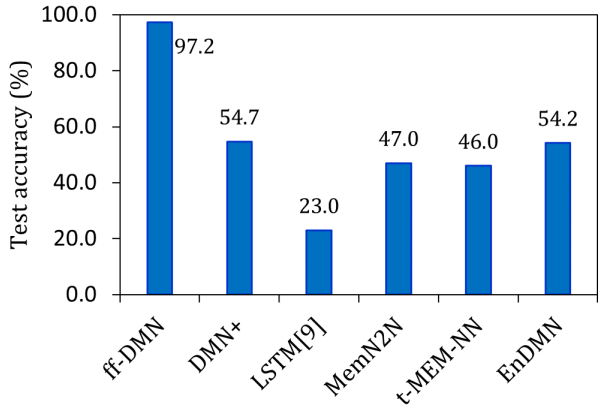


Figure 6.13: Test accuracies of different QA systems on task 16 induction

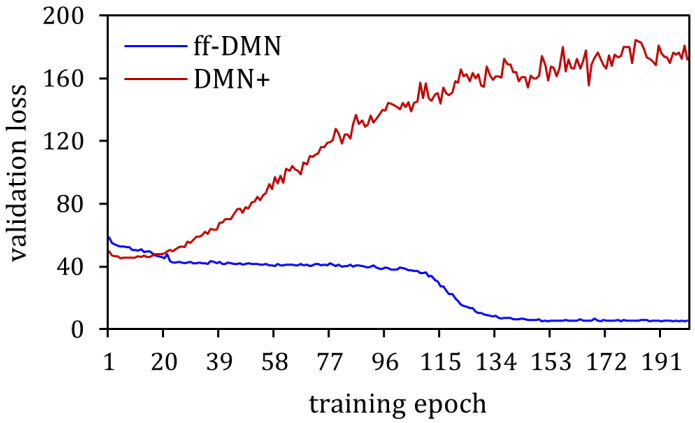


Figure 6.14: Validation losses of ff-DMN and DMN+ on task 16

Figure 6.14 shows the validation losses of the two models on task 16 during training. As training progresses, the loss for ff-DMN continues to trend downwards.

However, for DMN+, after the first few epochs the validation loss starts to increase. This suggests that DMN+ has a serious overfitting problem on this task.

6.5.3.2 Non-sequential Tasks

Out of the 20 tasks in the dataset, there are 8 tasks, which are task 4, 5 15, 16, 17, 18, 19 and 20. In these tasks the order of the facts is unimportant. A sample of each of these tasks is provided in Figure 6.15.

Table 6.5: Prediction accuracies of ff-DMN and DMN+ on non-sequential tasks

Non-sequential Tasks	**ff-DMN**	**DMN+**	**LSTM**	**MemN2N**	***t*-MEM-NN**	**EnDMN**
4: Two Argument Relations	**100.0**	**100.0**	61.0	96.2	96.0	**100.0**
5: Three Argument Relations	92.7	99.5	70.0	85.9	88.0	**99.4**
15: Basic Deduction	**100.0**	**100.0**	21.0	100.0	**100.0**	**100.0**
16: Basic Induction	**97.2**	54.7	23.0	47.9	46.0	54.2
17: Positional Reasoning	**96.9**	95.8	51.0	49.9	53.0	94.9
18: Size Reasoning	**99.5**	97.9	52.0	86.4	91.0	98.5
19: Path Finding	98.3	**100.0**	8.0	12.6	14.0	**100.0**
20: Agent's Motivations	**100.0**	**100.0**	91.0	100.0	**100.0**	**100.0**
Mean test accuracy	**98.1**	93.5	47.1	72.4	73.5	93.4

In order to see the performance of ff-DMN on the other seven non-sequential tasks, the results of all eight non-sequential tasks are shown in Table 6.5. These results clearly show that ff-DMN is very competitive with all the other QA models. It outperforms all the other models on tasks 16, 17 and 18. This suggests that incorporating RNNs could adversely affect the performance of the non-sequential tasks.

Task 4: Two Argument Relations

1 The kitchen is north of the bedroom.
2 The bedroom is north of the garden.
What is north of the bedroom? kitchen 1
What is the the bedroom north of? garden 2

Task 5: Three Argument Relations

1 Bill travelled to the office.
2 Bill picked up the football there.
3 Bill went to the bedroom.
4 Bill gave the football to Fred.
5 Fred handed the football to Bill.
6 Jeff went back to the office.
Who received the football? Bill 5

Task 15: Basic Deduction

1 Mice are afraid of wolves.
2 Gertrude is a mouse.
3 Cats are afraid of sheep.
4 Winona is a mouse.
5 Sheep are afraid of wolves.
6 Wolves are afraid of cats.
7 Emily is a mouse.
8 Jessica is a wolf.
What is gertrude afraid of? wolf 2 1

Task 16: Basic Induction

1 Lily is a frog.
2 Bernhard is a frog.
3 Bernhard is green.
4 Brian is a lion.
5 Brian is white.
6 Julius is a swan.
7 Julius is green.
8 Lily is green.
9 Greg is a swan.
What color is Greg? green 9 6 7

Task 17: Positional Reasoning

1 The triangle is to the right of the pink rectangle.
2 The pink rectangle is above the red square.
Is the red square above the triangle? no 2 1
Is the triangle above the red square? yes 1 2

Task 18: Size Reasoning

1 The box of chocolates fits inside the chest.
2 The box is bigger than the chest.
3 The box is bigger than the suitcase.
4 The suitcase fits inside the box.
5 The container is bigger than the box of chocolates.
Does the box fit in the box of chocolates? no 1 2
Is the box of chocolates bigger than the box? no 1 2

Task 19: Path Finding

1 The office is east of the hallway.
2 The kitchen is north of the office.
3 The garden is west of the bedroom.
4 The office is west of the garden.
5 The bathroom is north of the garden.
How do you go from the kitchen to the garden? s,e 2 4

Task 20: Agent's Motivations

1 Yann is hungry.
2 Yann journeyed to the kitchen.
3 Yann took the apple there.
4 Jason is tired.
5 Jason moved to the bedroom.
6 Sumit is tired.
Why did yann go to the kitchen? hungry 1
Where does Sumit go? bedroom 6

Figure 6.15: Samples of 8 non-sequential tasks

6.5.3.3 Sequential Tasks

There are 12 sequential tasks in the dataset. In order to compare the performance of ff-DMN and DMN+, their results are reported in Table 6.6. The mean test accuracy shows that DMN+ outperforms ff-DMN. This is understandable, because DMN+ makes use of RNNs to capture sequential information of sentences, while ff-DMN does not utilize RNNs at all. For sequential tasks, preserving the sequential information of facts is critical for tackling this type of tasks.

Task 3: Three Supporting Facts

1 Fred travelled to the kitchen yesterday.
2 Julie journeyed to the park yesterday.
3 Mary journeyed to the park yesterday.
4 This morning Mary journeyed to the cinema.
5 Mary moved to the kitchen this evening.
6 This afternoon Mary moved to the office.
7 This morning Fred went to the bedroom.
8 This morning Julie went to the bedroom.
Where was Julie before going to the bedroom? park 8 2

Task 14: Time Reasoning

1 John went to the garden.
2 John discarded the apple there.
3 Sandra travelled to the bedroom.
4 Daniel moved to the bathroom.
5 Sandra got the milk.
6 Daniel travelled to the garden.
7 Sandra went back to the bathroom.
8 Daniel took the apple there.
9 Mary went back to the hallway.
10 Daniel went to the hallway.
11 Sandra went to the kitchen.
12 Mary journeyed to the bedroom.
13 Sandra journeyed to the hallway.
14 Daniel put down the apple.
15 Daniel put down the football there.
16 Sandra journeyed to the garden.
Where was the football before the garden? bathroom 15 6 4

Figure 6.16: Samples of task 3 and task 14

Take a closer look at the results of these tasks, ff-DMN does not demonstrate lower accuracies on all the 12 tasks. It has very competitive performance to DMN+ on task 14. It would be interesting to know why this happens. A sample of task 14 is displayed on the right side of Figure 6.16. Although the events happen in a sequential way, each sentence – the description of each event, contains a time word or phrase. The sequential information can be picked up through these time terms without using RNNs.

On the contrary, ff-DMN has the lowest accuracy on task 3 three supporting facts. A sample from task 3 is shown in the left side of Figure 6.16. There is no time term in each sentence, so the sequential knowledge of these events needs to be inferred by the order of the sentences. Without RNNs to capture such ordering information, ff-DMN has difficulty to achieve satisfactory performance on such tasks.

Table 6.6: Prediction accuracies of ff-DMN and DMN+ on sequential tasks

Sequential Tasks	**ff-DMN**	**DMN+**
1: Single Supporting Fact	64.3	**100.0**
2: Two Supporting Facts	52.2	**99.7**
3: Three Supporting Facts	29.4	**98.9**
6: Yes/No Questions	90.5	**100.0**
7: Counting	90.4	**97.6**
8: Lists/Sets	96.1	**100.0**
9: Simple Negation	92.1	**100.0**
10: Indefinite Knowledge	84.9	**100.0**
11: Basic Coreference	71.2	**100.0**
12: Conjunction	54.5	**100.0**
13: Compound	52.9	**100.0**
14: Time Reasoning	**99.8**	**99.8**
Mean test accuracy	73.2	**99.7**

6.6 Ensemble Model

Since ff-DMN performs better on the inductive reasoning tasks as well as some other non-sequential tasks and DMN+ works better on the sequential tasks and the remaining non-sequential tasks, a complete QA system could be designed by combining the

strengths of these two models.

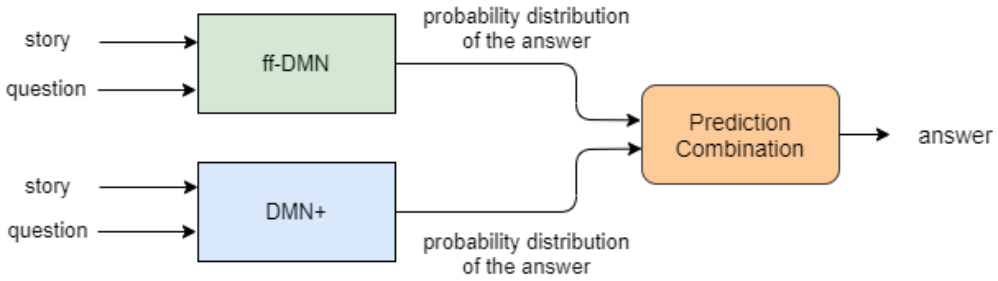


Figure 6.17: Illustration of the ensemble model

An approach is to use the ensemble learning [184], [185] method, which is very popular in machine learning to design an ensemble model. Figure 6.17 illustrates the structure of this ensemble model. It consists of both ff-DMN and DMN+, which are trained independently. At the prediction stage (also known as the testing stage), the two models work independently to arrive at their independent predictions of the probability distributions of the answer. There are two steps in the prediction combination module. The first step is to aggregate the two predicted probability distributions. Here the averaging ensemble method is utilized, which is one of the popular ensemble strategies [186], [187]. The second step is to determine the answer with the maximum value of the averaged probability distribution. These can be expressed as

$$\hat{P}^{(a)} = \frac{1}{2}(P_{ff\text{-}DMN} + P_{DMN+}), \tag{6.20}$$

$$answer = \arg\max_{\hat{p_j}^{(a)} \in \hat{P}^{(a)}} \hat{P}^{(a)}, \tag{6.21}$$

where $P_{ff\text{-}DMN}$ is the predicted probability distribution of ff-DMN, and P_{DMN+} is that of DMN+. $\hat{P}^{(a)}$ is the averaged probability distribution, and $\hat{P}^{(a)} = \bigcup_j \{p_j^{(a)} : j \in \mathcal{J}\}$ where $\mathcal{J}$ represents the set of all the possible indices.

The results listed in Table 6.7 show that the ensemble model outperforms the other systems on all the 20 tasks. The mean test accuracy of 99.6% is the highest among

Table 6.7: Test accuracies of the proposed ensemble model and the other QA systems

Tasks	**Ensemble**	**DMN+**	**LSTM**	**MemN2N**	***t*-MEM-NN**	**EnDMN**
1: Single Supporting Fact	**100.0**	**100.0**	50.0	**100.0**	**100.0**	**100.0**
2: Two Supporting Facts	**99.8**	99.7	20.0	99.7	82.0	98.6
3: Three Supporting Facts	**98.9**	**98.9**	20.0	97.9	57.0	88.8
4: Two Argument Relations	**100.0**	**100.0**	61.0	96.2	96.0	**100.0**
5: Three Argument Relations	**99.8**	99.5	70.0	85.9	88.0	**99.4**
6: Yes/No Questions	**100.0**	**100.0**	48.0	99.9	78.0	**100.0**
7: Counting	**98.8**	97.6	49.0	98.0	81.0	97.7
8: Lists/Sets	**100.0**	**100.0**	45.0	99.1	89.0	**100.0**
9: Simple Negation	**100.0**	**100.0**	64.0	99.7	86.0	**100.0**
10: Indefinite Knowledge	**100.0**	**100.0**	44.0	**100.0**	84.0	**100.0**
11: Basic Coreference	**100.0**	**100.0**	72.0	99.9	97.0	**100.0**
12: Conjunction	**100.0**	**100.0**	74.0	**100.0**	99.0	**100.0**
13: Compound Coreference	**100.0**	**100.0**	94.0	**100.0**	90.0	**100.0**
14: Time Reasoning	**100.0**	99.8	27.0	99.9	89.0	98.5
15: Basic Deduction	**100.0**	**100.0**	21.0	**100.0**	**100.0**	**100.0**
16: Basic Induction	**97.6**	54.7	23.0	47.9	46.0	54.2
17: Positional Reasoning	**97.3**	95.8	51.0	49.9	53.0	94.9
18: Size Reasoning	**99.9**	97.9	52.0	86.4	91.0	**98.5**
19: Path Finding	**100.0**	**100.0**	8.0	12.6	14.0	**100.0**
20: Agent's Motivations	**100.0**	**100.0**	91.0	**100.0**	**100.0**	**100.0**
Mean test accuracy	**99.6**	97.2	51.9	88.7	81.0	96.5

all the systems. The ensemble model passes all the 20 tasks by achieving the accuracies above the threshold 95% defined in [20], while none the other models is able to do.

6.7 Summary

In this chapter, by using the DMN+ QA model the role of RNNs is investigated. It is shown that applying RNNs to preserve the positional and ordering information of facts in RNN-based neural QA systems could cause substantial performance degradation in some circumstances. This is caused by the RNNs memorizing the order of facts in the training dataset. When these trained networks are presented with test samples where the order of facts is not adequately present in the training data, they performed poorly.

The results showed that this problem can be alleviated by training data augmentation without changing the original model. An RNN-free attention mechanism is also proposed to replace the RNN-based attention mechanism in the episodic memory module, which leads to the proposal of the MoDMN+ model. The results demonstrate that MoDMN+ can effectively alleviate the problem.

After identifying the adverse effects of RNN-based attention mechanism, RNNs in the input module are discarded and a new RNN-free input module is proposed. This leads to the development of another new QA model – ff-DMN. The results show that ff-DMN performs significantly better on the inductive reasoning tasks than the other models. It also shows better performance on some of the other seven non-sequential tasks in the bAbI dataset. Further, combining the strength of ff-DMN and DMN+ the proposed ensemble model outperforms on all the 20 tasks.

Chapter 7

Conclusions

In this thesis, in-depth analyses of the attention mechanisms and of the roles of RNNs in RNN-based QA models have been conducted. Through these analyses, improvements have been proposed and evaluated. These improvements have been shown, through appropriate simulation studies, to enhance the performance of such QA systems. They also offer new insights into the inner workings of various parts of these systems. The research contributions of this thesis are in line with the five research objectives that have been identified in Section 1.3. They are summarised as follows.

1. **Design of the Baseline Model**

   A novel baseline model that captured the essential ingredients of RNN-based QA models was design. It has been verified, through simulation studies, to produce performance results that are commensurate with five leading RNN-based QA models in the literature. Therefore, it can be effectively used for further studies that were conducted for the design of the attention mechanisms.

2. **Improved Attention Similarity Function**

   Using the baseline model, the effectiveness of eleven attention similarity score functions are compared. The insights gained in this study inspired a new function, called T-trilinear function, proposed by applying the $ReLU$ operation to the trilinear function. Experimental results confirmed that the proposed T-trilinear function outperforms all the existing functions.

3. **Improved Relevant Information Generation Methods**

   The baseline model has also been used to compare several relevant information generation methods in attention mechanisms. A new method with the inclusion of two additional terms has been proposed as a result. Its output is further processed through an FNN which produced better performances in comparison with all existing methods.

4. **RNNs in Gated Attention**

   One important place that RNNs have been used in the RNN-based QA systems tackling multi-hop reasoning tasks is the gated attention. Using the DMN+ QA model as a case study, it was found that RNNs in gated attention has the undesirable effect of causing the model to memorize the order of supporting facts. Consequently, the trained model does not generalize well to those order of facts that has not appeared in the training samples. Two approaches were proposed to solve this problem from two different perspectives – the training data and the architecture of the model itself. By combining data augmentation with an RNN-free gated attention, a modified DMN+, called MoDMN+, is proposed. Experimental results show that MoDMN+ can effectively tackle the problem.

5. **RNNs in the Input Module of DMN+**

   A new RNN-free input module was introduced into the DMN+ QA model. This new model, called ff-DMN, is shown to be able to solve inductive reasoning tasks

from the bAbI dataset with a predictive accuracy of 97.2%. In contrast, other QA models can only achieve at most 54.7% for the same tasks. Furthermore, an ensemble model was developed and it showed superior performance on all 20 tasks in this dataset.

The research presented in this thesis provides a better understanding of how different ways to compute attention affect the performance of an RNN-based neural QA system. It also offers new discoveries of the role RNNs play with regard to inductive reasoning tasks. All these contribute to our understanding of such neural QA systems and help us to design better systems such as what has been proposed in this thesis.

7.1 Future Work

This research could be extended further in several different ways. In the study of the role of RNNs, the inductive dataset used was synthetic. It will be interesting to apply the proposed models to a non-synthetic dataset which requires the similar type of inductive reasoning. Furthermore, the inductive reasoning task used in the experiments requires reasoning over multiple sentences within a story or paragraph. Future work could explore reasoning over larger chunks of texts, such as multiple paragraphs within a document or even over several documents.

The baseline model was only used to study attention mechanism in this thesis. However, it could also be used to compare methods used in other modules. Comparing different features as part of the input for the model would also be worth exploring.

The way self-attention in Transformer type of architectures works is different from the modular attention that is studied in this thesis. Therefore, a possible direction

of future research is to understand how self-attention works and what factors affect its performance.

References

[1] J. Bullock, A. Luccioni, K. H. Pham, C. S. N. Lam and M. Luengo-Oroz, 'Mapping the landscape of artificial intelligence applications against covid-19,' *Journal of Artificial Intelligence Research*, vol. 69, pp. 807–845, 2020, ISSN: 1076-9757.

[2] J. I. Criado and J. R. Gil-Garcia, 'Creating public value through smart technologies and strategies: From digital services to artificial intelligence and beyond,' *International Journal of Public Sector Management*, 2019, ISSN: 0951-3558.

[3] H. Mehr, H. Ash and D. Fellow, 'Artificial intelligence for citizen services and government,' *Ash Cent. Democr. Gov. Innov. Harvard Kennedy Sch., no. August*, pp. 1–12, 2017.

[4] S. Zhao, F. Blaabjerg and H. Wang, 'An overview of artificial intelligence applications for power electronics,' *IEEE Transactions on Power Electronics*, 2020, ISSN: 0885-8993.

[5] R. Bavaresco, D. Silveira, E. Reis *et al.*, 'Conversational agents in business: A systematic literature review and future research directions,' *Computer Science Review*, vol. 36, p. 100 239, 2020, ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2020.100239.

[6] L. Laranjo, A. G. Dunn, H. L. Tong *et al.*, 'Conversational agents in healthcare: A systematic review,' *Journal of the American Medical Informatics Association*, vol. 25, no. 9, pp. 1248–1258, 2018, ISSN: 1067-5027.

[7] J. Trivedi, 'Examining the customer experience of using banking chatbots and its impact on brand love: The moderating role of perceived risk,' *Journal of internet Commerce*, vol. 18, no. 1, pp. 91–111, 2019, ISSN: 1533-2861.

[8] D. Zumstein and S. Hundertmark, 'Chatbots–an interactive technology for personalized communication, transactions and services,' *IADIS International Journal on WWW/Internet*, vol. 15, no. 1, 2017, ISSN: 1645-7641.

[9] A. L. Guzman, 'Voices in and of the machine: Source orientation toward mobile virtual assistants,' *Computers in Human Behavior*, vol. 90, pp. 343–350, 2019, ISSN: 0747-5632.

[10] A. Maedche, C. Legner, A. Benlian *et al.*, 'Ai-based digital assistants,' *Business & Information Systems Engineering*, vol. 61, no. 4, pp. 535–544, 2019, ISSN: 1867-0202.

[11] A. B. Brendel, L. M. Kolbe and S. Diederich, 'Designing anthropomorphic enterprise conversational agents,' *Business & Information Systems Engineering*, vol. 62, no. 3, pp. 193–209, 2020.

[12] A. B. Kocaballi, S. Berkovsky, J. C. Quiroz *et al.*, 'The personalization of conversational agents in health care: Systematic review,' *Journal of medical Internet research*, vol. 21, no. 11, e15360, 2019.

[13] S. Barko-Sherif, D. Elsweiler and M. Harvey, 'Conversational agents for recipe recommendation,' in *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval*, 2020, pp. 73–82.

[14] H. Io and C. Lee, 'Chatbots and conversational agents: A bibliometric analysis,' in *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, IEEE, 2017, pp. 215–219, ISBN: 1538609487.

[15] B. F. Green Jr, A. K. Wolf, C. Chomsky and K. Laughery, 'Baseball: An automatic question-answerer,' in *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, ACM, 1961, pp. 219–224.

[16] J. Weizenbaum, 'Eliza—a computer program for the study of natural language communication between man and machine,' *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966, ISSN: 0001-0782.

[17] W. A. Woods and R. Kaplan, 'Lunar rocks in natural english: Explorations in natural language question answering,' *Linguistic Structures Processing*, vol. 5, pp. 521–569, 1977.

[18] W. J. Plath, 'Request: A natural language question-answering system,' *IBM Journal of Research and Development*, vol. 20, no. 4, pp. 326–335, 1976, ISSN: 0018-8646.

[19] A. Bouziane, D. Bouchiha, N. Doumi and M. Malki, 'Question answering systems: Survey and trends,' *Procedia Computer Science*, vol. 73, pp. 366–375, 2015, ISSN: 1877-0509.

[20] A. Kumar, O. Irsoy, P. Ondruska *et al.*, 'Ask me anything: Dynamic memory networks for natural language processing,' in *Proceedings of the 33rd International Conference on Machine Learning*, vol. 48, New York, USA, 2016, pp. 1378–1387.

[21] K. Tolias and S. P. Chatzis, 'T-exponential memory networks for question-answering machines,' *IEEE transactions on neural networks and learning systems*, 2018, ISSN: 2162-237X.

[22] A. Juárez-González, A. Téllez-Valero, C. Denicia-Carral, M. Montes-y-Gómez and L. Villaseñor-Pineda, 'Using machine learning and text mining in question answering,' in *Workshop of the Cross-Language Evaluation Forum for European Languages*, Springer, 2006, pp. 415–423.

[23] Y. Chali, S. R. Joty and S. A. Hasan, 'Complex question answering: Unsupervised learning approaches and experiments,' *Journal of Artificial Intelligence Research*, vol. 35, pp. 1–47, 2009, ISSN: 1076-9757.

[24] D. Ferrucci, E. Brown, J. Chu-Carroll *et al.*, 'Building watson: An overview of the deepqa project,' *AI magazine*, vol. 31, no. 3, pp. 59–79, 2010, ISSN: 0738-4602.

[25] D. Ferrucci, A. Levas, S. Bagchi, D. Gondek and E. T. Mueller, 'Watson: Beyond jeopardy!' *Artificial Intelligence*, vol. 199, pp. 93–105, 2013, ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2012.06.009.

[26] X. Liu, Y. Shen, K. Duh and J. Gao, 'Stochastic answer networks for machine reading comprehension,' in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia, Jul. 2018, pp. 1694–1704. DOI: http://dx.doi.org/10.18653/v1/P18-1157.

[27] M. Seo, A. Kembhavi, A. Farhadi and H. Hajishirzi, 'Bidirectional attention flow for machine comprehension,' in *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, Toulon, France, Apr. 2017.

[28] C. Xiong, V. Zhong and R. Socher, 'Dynamic coattention networks for question answering,' in *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, Toulon, France, Apr. 2017.

[29] A. W. Yu, D. Dohan, M.-T. Luong *et al.*, 'Qanet: Combining local convolution with global self-attention for reading comprehension,' in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, May 2018.

[30] J. Weston, A. Bordes, S. Chopra *et al.*, 'Towards ai-complete question answering: A set of prerequisite toy tasks,' in *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, Caribe Hilton, San Juan, Puerto Rico, 2016.

[31] S. Sukhbaatar, J. Weston and R. Fergus, 'End-to-end memory networks,' in *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS'15)*, vol. 2, Montreal, Canada, 2015, pp. 2440–2448.

[32] C. Xiong, S. Merity and R. Socher, 'Dynamic memory networks for visual and textual question answering,' in *Proceedings of the 33rd International Conference on Machine Learning*, vol. 48, New York, USA, 2016, pp. 2397–2406.

[33] R. F. Simmons, 'Answering english questions by computer: A survey,' *Communications of the ACM*, vol. 8, no. 1, pp. 53–70, 1965, ISSN: 0001-0782.

[34] R. F. Simmons, 'Natural language question-answering systems: 1969,' *Communications of the ACM*, vol. 13, no. 1, pp. 15–30, 1970, ISSN: 0001-0782.

[35] S. KVNO and A. OETTINOER, 'Syntactic structure and ambiguity in english,' in *Proceedings of the joint computer conference*, Spartan Books, Baltimore, 1963.

[36] J. A. Craig, S. C. Berezner, H. C. Carney and C. R. Longyear, 'Deacon: Direct english access and control,' in *Proceedings of the joint computer conference*, 1966, pp. 365–380.

[37] F. B. Thompson, J. A. Craig, G. D. Gibbons, J. W. Gwynn and J. S. Pruett, 'Deacon breadboard summary,' GENERAL ELECTRIC CO SANTA BARBARA CA TEMPO, Report, 1964.

[38] W. A. Woods, 'Progress in natural language understanding: An application to lunar geology,' in *Proceedings of the national computer conference and exposition*, 1973, pp. 441–450.

[39] B. W. Ballard, 'A general computational treatment of comparatives for natural language question answering,' in *26th Annual Meeting of the Association for Computational Linguistics*, 1988, pp. 41–48.

[40] A. C. Graesser, S. E. Gordon and L. E. Brainerd, 'Quest: A model of question answering,' *Computers & Mathematics with Applications*, vol. 23, no. 6-9, pp. 733–745, 1992, ISSN: 0898-1221.

[41] S. M. Harabagiu and D. Moldovan, 'An intelligent system for question answering,' *University of Southern California*, pp. 1–5, 1996.

[42] G. A. Miller, 'Wordnet: A lexical database for english,' *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995, ISSN: 0001-0782.

[43] K. M. Colby and F. D. Hilf, 'Multidimensional analysis in evaluating a simulation of paranoid thought,' Stanford University, Report, 1973.

[44] K. M. Colby, R. C. Parkison and W. S. Faught, 'Pattern-matching rules for the recognition of natural language dialogue expressions,' Stanford University, Report, 1974.

[45] K. M. Colby, S. Weber and F. D. Hilf, 'Artificial paranoia,' *Artificial Intelligence*, vol. 2, no. 1, pp. 1–25, 1971, ISSN: 0004-3702.

[46] R. Wallace, 'Artificial linguistic internet computer entity (alice),' *City*, 1995.

[47] K. M. Colby, 'Ten criticisms of parry,' *SIGART Bull.*, no. 48, pp. 5–9, 1974, ISSN: 0163-5719. DOI: 10.1145/1045200.1045202.

[48] A. Ratnaparkhi, 'A maximum entropy model for part-of-speech tagging,' in *Conference on empirical methods in natural language processing*, 1996.

[49] M. J. Pickering and R. P. Van Gompel, 'Syntactic parsing,' in *Handbook of psycholinguistics*. Elsevier, 2006, pp. 455–503.

[50] B. Mohit, 'Named entity recognition,' in *Natural language processing of semitic languages*. Springer, 2014, pp. 221–245.

[51] H. T. Dang, D. Kelly and J. J. Lin, 'Overview of the trec 2007 question answering track,' in *Text REtrieval Conference (TREC)*, vol. 7, 2007, pp. 63–80.

[52] E. M. Voorhees, 'The trec-8 question answering track report,' in *Text REtrieval Conference (TREC)*, vol. 99, 1999, pp. 77–82.

[53] P. Gupta and V. Gupta, 'A survey of text question answering techniques,' *International Journal of Computer Applications*, vol. 53, no. 4, 2012, ISSN: 0975-8887.

[54] E. Breck, J. Burger, L. Ferro, D. House, M. Light and I. Mani, 'A sys called qanda,' in *The Eighth Text REtrieval Conference (TREC)*, Citeseer, 1999.

[55] E. Brill, S. Dumais and M. Banko, 'An analysis of the askmsr question-answering system,' in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, 2002, pp. 257–264.

[56] C. Denicia-Carral, M. Montes-y-Gómez, L. Villaseñor-Pineda and R. G. Hernández, 'A text mining approach for definition question answering,' in *International Conference on Natural Language Processing (in Finland)*, Springer, 2006, pp. 76–86.

[57] J. Prager, J. Chu-Carroll, K. Czuba, C. Welty, A. Ittycheriah and R. Mahindru, 'Ibm's piquant in trec2003,' IBM THOMAS J WATSON RESEARCH CENTER YORKTOWN HEIGHTS NY, Report, 2003.

[58] J. Prager, D. Radev, E. Brown and A. Coden, 'The use of predictive annotation for question answering in trec8,' in *The Eighth Text REtrieval Conference (TREC)*, Citeseer, 1999.

[59] J. Kupiec, 'Murax: A robust linguistic approach for question answering using an on-line encyclopedia,' in *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, 1993, pp. 181–190.

[60] U. Hermjakob, 'Parsing and question classification for question answering,' in *Proceedings of the ACL 2001 workshop on open-domain question answering*, 2001.

[61] M. A. Pasca and S. M. Harabagiu, 'High performance question/answering,' in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, 2001, pp. 366–374.

[62] Y. Chen, J. D. Elenee Argentinis and G. Weber, 'IBM Watson: How cognitive computing can be applied to big data challenges in life sciences research,' *Clinical Therapeutics*, vol. 38, no. 4, pp. 688–701, 2016, ISSN: 0149-2918. DOI: https://doi.org/10.1016/j.clinthera.2015.12.001.

[63] S. S. Murtaza, P. Lak, A. Bener and A. Pischdotchian, 'How to effectively train ibm watson: Classroom experience,' in *the 49th Hawaii International Conference on System Sciences (HICSS)*, IEEE, 2016, pp. 1663–1670, ISBN: 0769556701.

[64] L. Deng and D. Yu, 'Deep learning: Methods and applications,' *Found. Trends Signal Process.*, vol. 7, no. 3–4, pp. 197–387, 2014, ISSN: 1932-8346. DOI: https://doi.org/10.1561/2000000039.

[65] Y. LeCun, Y. Bengio and G. Hinton, 'Deep learning,' *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[66] I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*. MIT press, 2016, ISBN: 0262337371.

[67] A. K. Jain, J. Mao and K. M. Mohiuddin, 'Artificial neural networks: A tutorial,' *Computer*, vol. 29, no. 3, pp. 31–44, 1996, ISSN: 0018-9162.

[68] B. Yegnanarayana, *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009, ISBN: 8120312538.

[69] R. Reed and R. J. MarksII, *Neural smithing: supervised learning in feedforward artificial neural networks*. Mit Press, 1999, ISBN: 0262181908.

[70] H.-F. Yang, K. Lin and C.-S. Chen, 'Supervised learning of semantics-preserving hash via deep convolutional neural networks,' *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 2, pp. 437–451, 2017, ISSN: 0162-8828.

[71] P. Baldi, 'Autoencoders, unsupervised learning, and deep architectures,' in *Proceedings of ICML workshop on unsupervised and transfer learning*, JMLR Workshop and Conference Proceedings, 2012, pp. 37–49.

[72] P. Vincent, H. Larochelle, Y. Bengio and P.-A. Manzagol, 'Extracting and composing robust features with denoising autoencoders,' in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[73] D. Svozil, V. Kvasnicka and J. Pospichal, 'Introduction to multi-layer feed-forward neural networks,' *Chemometrics and intelligent laboratory systems*, vol. 39, no. 1, pp. 43–62, 1997, ISSN: 0169-7439.

[74] S. Lawrence, C. L. Giles, A. C. Tsoi and A. D. Back, 'Face recognition: A convolutional neural-network approach,' *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997, ISSN: 1045-9227.

[75] Y. LeCun and Y. Bengio, 'Convolutional networks for images, speech, and time series,' *The handbook of brain theory and neural networks*, vol. 3361, no. 10, 1995.

[76] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, 'Empirical evaluation of gated recurrent neural networks on sequence modeling,' *arXiv preprint arXiv:1412.3555*, 2014.

[77] R. Pascanu, T. Mikolov and Y. Bengio, 'On the difficulty of training recurrent neural networks,' in *International conference on machine learning*, 2013, pp. 1310–1318.

[78] M. Schuster and K. K. Paliwal, 'Bidirectional recurrent neural networks,' *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997, ISSN: 1053-587X.

[79] Y. Bengio, 'Learning deep architectures for AI,' *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009, ISSN: 1935-8237.

[80] Y. Bengio, A. Courville and P. Vincent, 'Representation learning: A review and new perspectives,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013, ISSN: 0162-8828.

[81] A. Graves, A.-r. Mohamed and G. Hinton, 'Speech recognition with deep recurrent neural networks,' in *IEEE International Conference on Acoustics, Speech and Signal Processing Proceedings (ICASSP)*, IEEE, 2013, pp. 6645–6649, ISBN: 1479903566.

[82] Y. LeCun, Y. Bengio and G. Hinton, 'Deep learning,' *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, ISSN: 0028-0836.

[83] A.-R. Mohamed, G. E. Dahl and G. Hinton, 'Acoustic modeling using deep belief networks,' *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012, ISSN: 1558-7916.

[84] F. J. Huang, Y.-L. Boureau and Y. LeCun, 'Unsupervised learning of invariant feature hierarchies with applications to object recognition,' in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2007, pp. 1–8, ISBN: 1424411793.

[85] A. Krizhevsky, I. Sutskever and G. E. Hinton, 'Imagenet classification with deep convolutional neural networks,' in *Advances in neural information processing systems (NIPS)*, 2012, pp. 1097–1105.

[86] W. B. Croft and J. Lafferty, *Language modeling for information retrieval*. Springer Science & Business Media, 2013, vol. 13, ISBN: 9401701717.

[87] K. Moholkar, K. Rathod, K. Rathod, M. Tomar and S. Rai, 'Sentiment classification using recurrent neural network,' in *Intelligent Communication Technologies and Virtual Mobile Networks*, Springer, 2019, pp. 487–493.

[88] V. H. Phung and E. J. Rhee, 'A deep learning approach for classification of cloud image patches on small datasets,' *Journal of information and communication convergence engineering*, vol. 16, no. 3, pp. 173–178, 2018, ISSN: 2234-8255.

[89] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer and R. Harshman, 'Indexing by latent semantic analysis,' *Journal of the American Society for Information Science*, vol. 41, no. 6, p. 391, 1990, ISSN: 0002-8231.

[90] K. Lund and C. Burgess, 'Producing high-dimensional semantic spaces from lexical co-occurrence,' *Behavior Research Methods, Instruments, & Computers*, vol. 28, no. 2, pp. 203–208, 1996, ISSN: 0743-3808.

[91] D. L. Rohde, L. M. Gonnerman and D. C. Plaut, 'An improved model of semantic similarity based on lexical co-occurrence,' *Communications of the ACM*, vol. 8, no. 627-633, p. 116, 2006.

[92] J. A. Bullinaria and J. P. Levy, 'Extracting semantic representations from word co-occurrence statistics: A computational study,' *Behavior research methods*, vol. 39, no. 3, pp. 510–526, 2007, ISSN: 1554-351X.

[93] R. Lebret and R. Collobert, 'Word embeddings through hellinger pca,' in *Preceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Gothenburg, Sweden, 2014, pp. 482–490.

[94] J. Pennington, R. Socher and C. Manning, 'Glove: Global vectors for word representation,' in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, Doha, Qatar, Oct. 2014, pp. 1532–1543. DOI: http://dx.doi.org/10.3115/v1/D14-1162.

[95] C. Liu, W. Li, B. Demarest *et al.*, 'Iucl at semeval-2016 task 6: An ensemble model for stance detection in twitter,' in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 2016, pp. 394–400.

[96] H. Bøhler, P. Asla, E. Marsi and R. Sætre, 'Idi @ ntnu at semeval-2016 task 6: Detecting stance in tweets using shallow features and glove vectors for word representation,' in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 2016, pp. 445–450.

[97] Q. Zhou, N. Yang, F. Wei, C. Tan, H. Bao and M. Zhou, 'Neural question generation from text: A preliminary study,' in *National CCF Conference on Natural Language Processing and Chinese Computing*, Springer, 2017, pp. 662–671.

[98] L.-C. Yu, J. Wang, K. R. Lai and X. Zhang, 'Refining word embeddings for sentiment analysis,' in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 534–539.

[99] D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning representations by back-propagating errors,' *Nature*, vol. 323, no. 6088, p. 533, 1986, ISSN: 1476-4687.

[100] Y. Bengio, R. Ducharme, P. Vincent and C. Jauvin, 'A neural probabilistic language model,' *Journal of Machine Learning Research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[101] R. Collobert and J. Weston, 'A unified architecture for natural language processing: Deep neural networks with multitask learning,' in *Proceedings of the 25th International Conference on Machine Learning*, ACM, 2008, pp. 160–167, ISBN: 1605582050.

[102] T. Mikolov, K. Chen, G. Corrado and J. Dean, 'Efficient estimation of word representations in vector space,' *arXiv preprint arXiv:1301.3781*, 2013.

[103] D. Ayata, M. Saraclar and A. Ozgur, 'Busem at semeval-2017 task 4a sentiment analysis with word embedding and long short term memory rnn approaches,' in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 2017, pp. 777–783.

[104] S. Poria, E. Cambria and A. Gelbukh, 'Aspect extraction for opinion mining with a deep convolutional neural network,' *Knowledge-Based Systems*, vol. 108, pp. 42–49, 2016, ISSN: 0950-7051.

[105] J. H. Lau and T. Baldwin, 'An empirical evaluation of doc2vec with practical insights into document embedding generation,' in *Proceedings of the 1st Workshop on Representation Learning for NLP*, 2016, pp. 78–86.

[106] M. Kusner, Y. Sun, N. Kolkin and K. Weinberger, 'From word embeddings to document distances,' in *International Conference on Machine Learning*, 2015, pp. 957–966.

[107] L. Shen, J. Xu and R. Weischedel, 'A new string-to-dependency machine translation algorithm with a target dependency language model,' *Proceedings of ACL-08: HLT*, pp. 577–585, 2008.

[108] F. J. Pineda, 'Generalization of back-propagation to recurrent neural networks,' *Physical review letters*, vol. 59, no. 19, p. 2229, 1987.

[109] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký and S. Khudanpur, 'Recurrent neural network based language model,' in *Proceedings of the 11th Annual Conference of the International Speech Communication Association (Interspeech)*, vol. 2, 2010, pp. 1045–1048.

[110] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber *et al.*, 'Gradient flow in recurrent nets: The difficulty of learning long-term dependencies,' *A field guide to dynamical recurrent neural networks. IEEE Press*, 2001.

[111] S. Hochreiter and J. Schmidhuber, 'Long short-term memory,' *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997, ISSN: 0899-7667.

[112] K. Cho, B. van Merrienboer, C. Gulcehre *et al.*, 'Learning phrase representations using rnn encoder–decoder for statistical machine translation,' in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[113] W. De Mulder, S. Bethard and M.-F. Moens, 'A survey on the application of recurrent neural networks to statistical language modeling,' *Computer Speech & Language*, vol. 30, no. 1, pp. 61–98, 2015, ISSN: 0885-2308.

[114] M. Sundermeyer, R. Schlüter and H. Ney, 'Lstm neural networks for language modeling,' in *Thirteenth annual conference of the international speech communication association*, 2012.

[115] K. Cho, B. van Merrienboer, D. Bahdanau and Y. Bengio, 'On the properties of neural machine translation: Encoder-decoder approaches,' in *Proceedings of the 8th Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, Doha, Qatar, Oct. 2014, pp. 103–111. DOI: http://dx.doi.org/10.3115/v1/W14-4012.

[116] B. Zhang, D. Xiong, J. Xie and J. Su, 'Neural machine translation with gru-gated attention model,' *IEEE transactions on neural networks and learning systems*, vol. 31, no. 11, pp. 4688–4698, 2020, ISSN: 2162-237X. DOI: https://doi.org/10.1109/TNNLS.2019.2957276.

[117] K. Baktha and B. Tripathy, 'Investigation of recurrent neural networks in the field of sentiment analysis,' in *2017 International Conference on Communication and Signal Processing (ICCSP)*, IEEE, 2017, pp. 2047–2050, ISBN: 1509038000.

[118] C. Yin, J. Tang, Z. Xu and Y. Wang, 'Memory augmented deep recurrent neural network for video question answering,' *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3159–3167, 2019, ISSN: 2162-237X.

[119] J. Eapen, D. Bein and A. Verma, 'Novel deep learning model with cnn and bi-directional lstm for improved stock market index prediction,' in *2019 IEEE 9th annual computing and communication workshop and conference (CCWC)*, IEEE, 2019, pp. 0264–0270, ISBN: 1728105544.

[120] A. Graves and J. Schmidhuber, 'Framewise phoneme classification with bidirectional lstm and other neural network architectures,' *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005, ISSN: 0893-6080.

[121] M. Tan, C. d. Santos, B. Xiang and B. Zhou, 'Lstm-based deep learning models for non-factoid answer selection,' in *International Conference on Learning Representations (workshop track)*, 2015.

[122] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad and S. W. Baik, 'Action recognition in video sequences using deep bi-directional lstm with cnn features,' *IEEE access*, vol. 6, pp. 1155–1166, 2017, ISSN: 2169-3536.

[123] D. Weissenborn, G. Wiese and L. Seiffe, 'Making neural qa as simple as possible but not simpler,' in *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, vol. abs/1703.04816, Vancouver, Canada, 2017, pp. 271–280. DOI: https://doi.org/10.18653/v1/K17-1028.

[124] C. Clark and M. Gardner, 'Simple and effective multi-paragraph reading comprehension,' in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia, 2018, pp. 845–855. DOI: http://dx.doi.org/10.18653/v1/P18-1078.

[125] C. Yue, H. Cao, K. Xiong, A. Cui, H. Qin and M. Li, 'Enhanced question understanding with dynamic memory networks for textual question answering,' *Expert Systems with Applications*, vol. 80, pp. 39–45, 2017, ISSN: 0957-4174.

[126] X. Zhu, Z. Li, X. Li, S. Li and F. Dai, 'Attention-aware perceptual enhancement nets for low-resolution image classification,' *Information Sciences*, vol. 515, pp. 233–247, 2020. DOI: https://doi.org/10.1016/j.ins.2019.12.013.

[127] Y. Zhou, J. Li, H. Chen, Y. Wu, J. Wu and L. Chen, 'A spatiotemporal attention mechanism-based model for multi-step citywide passenger demand prediction,' *Information Sciences*, vol. 513, pp. 372–385, 2020. DOI: https://doi.org/10.1016/j.ins.2019.10.071.

[128] H. Choi, K. Cho and Y. Bengio, 'Fine-grained attention mechanism for neural machine translation,' *Neurocomputing*, vol. 284, pp. 171–176, 2018. DOI: https://doi.org/10.1016/j.neucom.2018.01.007.

[129] A. Osman and W. Samek, 'Drau: Dual recurrent attention units for visual question answering,' *Computer Vision and Image Understanding*, vol. 185, pp. 24–30, 2019. DOI: https://doi.org/10.1016/j.cviu.2019.05.001.

[130] D. T. Tran, A. Iosifidis, J. Kanniainen and M. Gabbouj, 'Temporal attention-augmented bilinear network for financial time-series data analysis,' *IEEE transactions on neural networks and learning systems*, vol. 30, no. 5, pp. 1407–1418, 2019. DOI: http://dx.doi.org/10.1109/TNNLS.2018.2869225.

[131] I. Sutskever, O. Vinyals and Q. V. Le, 'Sequence to sequence learning with neural networks,' in *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS'14)*, vol. 2, Montreal, Canada, 2014, pp. 3104–3112.

[132] M.-T. Luong, H. Pham and C. D. Manning, 'Effective approaches to attention-based neural machine translation,' in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Lisbon, Portugal, Sep. 2015, pp. 1412–1421. DOI: http://dx.doi.org/10.18653/v1/D15-1166.

[133] A. Vaswani, N. Shazeer, N. Parmar *et al.*, 'Attention is all you need,' in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*, Long Beach, California, USA, 2017, pp. 5998–6008.

[134] Y. Wang, K. Liu, J. Liu *et al.*, 'Multi-passage machine reading comprehension with cross-passage answer verification,' in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia, Jul. 2018, pp. 1918–1927. DOI: http://dx.doi.org/10.18653/v1/P18-1178.

[135] D. Hu, 'An introductory survey on attention mechanisms in nlp problems,' in *Proceedings of SAI Intelligent Systems Conference*, 2019, pp. 432–448. DOI: https://doi.org/10.1007/978-3-030-29513-4_31.

[136] S. Wang and J. Jiang, 'Machine comprehension using match-lstm and answer pointer,' in *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, Toulon, France, Apr. 2016.

[137] W. Wang, N. Yang, F. Wei, B. Chang and M. Zhou, 'Gated self-matching networks for reading comprehension and question answering,' in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada, Jul. 2017, pp. 189–198. DOI: http://dx.doi.org/10.18653/v1/P17-1018.

[138] A. Radford, K. Narasimhan, T. Salimans and I. Sutskever, *Improving language understanding by generative pre-training*, Blog, 2018.

[139] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, 'Bert: Pre-training of deep bidirectional transformers for language understanding,' in *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, Minneapolis, Minnesota, USA, Jun. 2019, pp. 4171–4186. DOI: http://dx.doi.org/10.18653/v1/N19-1423.

[140] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, *Language models are unsupervised multitask learners*, Blog, 2019.

[141] T. B. Brown, B. Mann, N. Ryder *et al.*, 'Language models are few-shot learners,' *arXiv preprint arXiv:2005.14165.*, 2020.

[142] W. Fedus, B. Zoph and N. Shazeer, 'Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity,' *arXiv preprint arXiv:2101.03961*, 2021.

[143] A. Ramesh, M. Pavlov, G. Goh *et al.*, 'Zero-shot text-to-image generation,' *arXiv preprint arXiv:2102.12092*, 2021.

[144] A. Romero, *Gpt-3 scared you? meet wu dao 2.0: A monster of 1.75 trillion parameters*, Blog, Jun. 2021. [Online]. Available: https://towardsdatascience.com/gpt-3-scared-you-meet-wu-dao-2-0-a-monster-of-1-75-trillion-parameters-832cd83db484.

[145] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le and R. Salakhutdinov, 'Transformer-xl: Attentive language models beyond a fixed-length context,' in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, 2019, pp. 2978–2988. DOI: https://doi.org/10.18653/v1/P19-1285.

[146] Y. Zhu, R. Kiros, R. Zemel *et al.*, 'Aligning books and movies: Towards story-like visual explanations by watching movies and reading books,' in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 19–27.

[147] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov and Q. V. Le, 'Xlnet: Generalized autoregressive pretraining for language understanding,' *Advances in neural information processing systems*, vol. 32, 2019.

[148] C. Raffel, N. Shazeer, A. Roberts *et al.*, 'Exploring the limits of transfer learning with a unified text-to-text transformer,' *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[149] P. Rajpurkar, J. Zhang, K. Lopyrev and P. Liang, 'Squad: 100,000+ questions for machine comprehension of text,' in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, USA, Nov. 2016, pp. 2383–2392. DOI: http://dx.doi.org/10.18653/v1/D16-1264.

[150] A. Trischler, T. Wang, X. Yuan *et al.*, 'NewsQA: A machine comprehension dataset,' in *Proceedings of the 2nd Workshop on Representation Learning for NLP*, Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 191–200. DOI: http://doi.org/10.18653/v1/W17-2623.

[151] M. Joshi, E. Choi, D. S. Weld and L. 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)*, 2017, pp. 1601–1611.

[152] T. Nguyen, M. Rosenberg, X. Song *et al.*, 'Ms marco: A human generated machine reading comprehension dataset,' in *Advances in neural information processing systems (NIPS)*, 2016.

[153] J. Weston, S. Chopra and A. Bordes, 'Memory networks,' in *International Conference on Learning Representations (ICLR)*, 2015.

[154] Z. Yang, P. Qi, S. Zhang *et al.*, 'Hotpotqa: A dataset for diverse, explainable multi-hop question answering,' in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2369–2380.

[155] J. Welbl, P. Stenetorp and S. Riedel, 'Constructing datasets for multi-hop reading comprehension across documents,' *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 287–302, 2018.

[156] A. Talmor and J. 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)*, 2018, pp. 641–651.

[157] M. Richardson, C. J. Burges and E. Renshaw, 'Mctest: A challenge dataset for the open-domain machine comprehension of text,' in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 193–203.

[158] G. Lai, Q. Xie, H. Liu, Y. Yang and E. Hovy, 'Race: Large-scale reading comprehension dataset from examinations,' in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 785–794.

[159] T. Khot, P. Clark, M. Guerquin, P. Jansen and A. Sabharwal, 'Qasc: A dataset for question answering via sentence composition,' in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 8082–8090.

[160] K. M. Hermann, T. Kocisky, E. Grefenstette *et al.*, 'Teaching machines to read and comprehend,' *Advances in neural information processing systems*, vol. 28, pp. 1693–1701, 2015.

[161] F. Hill, A. Bordes, S. Chopra and J. Weston, 'The goldilocks principle: Reading children's books with explicit memory representations,' in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015.

[162] T. Onishi, H. Wang, M. Bansal, K. Gimpel and D. McAllester, 'Who did what: A large-scale person-centered cloze dataset,' in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2230–2235.

[163] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins and K. Toutanova, 'BoolQ: Exploring the surprising difficulty of natural yes/no questions,' 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)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 2924–2936. DOI: http://doi.org/10.18653/v1/N19-1300.

[164] D. Dzendzik, C. Vogel and J. Foster, 'Is it dish washer safe? automatically answering "yes/no" questions using customer reviews,' in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 1–6. DOI: http://doi.org/10.18653/v1/N19-3001.

[165] Q. Jin, B. Dhingra, Z. Liu, W. Cohen and X. Lu, 'PubMedQA: A dataset for biomedical research question answering,' in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 2567–2577. DOI: http://doi.org/10.18653/v1/D19-1259.

[166] M. Saeidi, M. Bartolo, P. Lewis *et al.*, 'Interpretation of natural language rules in conversational machine reading,' in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 2087–2097. DOI: http://doi.org/10.18653/v1/D18-1233.

[167] S. Reddy, D. Chen and C. D. Manning, 'CoQA: A Conversational Question Answering Challenge,' *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 249–266, May 2019, ISSN: 2307-387X. DOI: http://doi.org/10.1162/tacl_a_00266.

[168] D. Chen, A. Fisch, J. Weston and A. Bordes, 'Reading wikipedia to answer open-domain questions,' in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada, 2017, pp. 1870–1879. DOI: https://doi.org/10.18653/v1/P17-1171.

[169] H.-Y. Huang, C. Zhu, Y. Shen and W. Chen, 'Fusionnet: Fusing via fully-aware attention with application to machine comprehension,' in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

[170] S. Kobayashi, R. Tian, N. Okazaki and K. Inui, 'Dynamic entity representation with max-pooling improves machine reading,' in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California, USA, Jun. 2016, pp. 850–855. DOI: http://dx.doi.org/10.18653/v1/N16-1099.

[171] R. Li, Z. Jiang, L. Wang, X. Lu and M. Zhao, 'Directional attention weaving for text-grounded conversational question answering,' *Neurocomputing*, vol. 391, pp. 13–24, 2020, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.01.056.

[172] Y. Feldman and R. El-Yaniv, 'Multi-hop paragraph retrieval for open-domain question answering,' in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, 2019, pp. 2296–2309. DOI: https://doi.org/10.18653/v1/P19-1222.

[173] C. Xiong, V. Zhong and R. Socher, 'Dcn+: Mixed objective and deep residual coattention for question answering,' in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, May 2018.

[174] J. Shore and R. Johnson, 'Properties of cross-entropy minimization,' *IEEE Transactions on Information Theory*, vol. 27, no. 4, pp. 472–482, 1981.

[175] M. D. Zeiler, 'Adadelta: An adaptive learning rate method,' *arXiv preprint arXiv:1212.5701*, 2012.

[176] D. Bahdanau, K. Cho and Y. Bengio, 'Neural machine translation by jointly learning to align and translate,' in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015.

[177] D. Britz, A. Goldie, M.-T. Luong and Q. Le, 'Massive exploration of neural machine translation architectures,' in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Copenhagen, Denmark, Sep. 2017, pp. 1442–1451. DOI: http://dx.doi.org/10.18653/v1/D17-1151.

[178] A. Graves, G. Wayne and I. Danihelka, 'Neural turing machines,' *arXiv preprint arXiv:1410.5401.*, 2014.

[179] X. Geng, L. Wang, X. Wang, B. Qin, T. Liu and Z. Tu, 'How does selective mechanism improve self-attention networks?' In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, 2020, pp. 2986–2995. DOI: http://dx.doi.org/10.18653/v1/2020.acl-main.269.

[180] I. Radosavovic, J. Johnson, S. Xie, W.-Y. Lo and P. Dollár, 'On network design spaces for visual recognition,' in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1882–1890.

[181] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He and P. Dollár, 'Designing network design spaces,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 428–10 436.

[182] H.-g. Lee and H. Kim, 'Gf-net: Improving machine reading comprehension with feature gates,' *Pattern Recognition Letters*, vol. 129, pp. 8–15, 2020, ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2019.10.030.

[183] D. P. Kingma and J. Ba, 'Adam: A method for stochastic optimization,' *arXiv preprint arXiv:1412.6980*, 2014.

[184] R. Polikar, 'Ensemble learning,' in *Ensemble machine learning*, Springer, 2012, pp. 1–34.

[185] O. Sagi and L. Rokach, 'Ensemble learning: A survey,' *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, e1249, 2018.

[186] H. Inoue and H. Narihisa, 'Improving generalization ability of self-generating neural networks through ensemble averaging,' in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2000, pp. 177–180.

[187] L. Yu, W. Huang, K. K. Lai and S. Wang, 'A reliability-based rbf network ensemble model for foreign exchange rates predication,' in *International Conference on Neural Information Processing*, Springer, 2006, pp. 380–389.

Glossary

AI Artificial Intelligence

AttnGRU Attention based Gated Recurrent Uni

biLSTM Bidirectional Long Short-Term Memory

BERT Bidirectional Encoder Representations from Transformers

BiDAF Bidirectional Dynamic Attention Flow

BOW Bag-of-Words

CBOW Continuous Bag-of-Words

CNNs Convolutional Neural Networks

DCN Dynamic Coattention Networks

DMN Dynamic Memory Networks

DMN+ Dynamic Memory Networks plus

DNN Deep Neural Network

EM Exact Match

EMA Exponential Moving Average

FNN Feedforward Neural Network

FNNs Feedforward Neural Networks

GRU Gated Recurrent Unit

GPT Generative Pre-Training

HAL Hyperspace Analogue to Language

LSTM Long Short-Term Memory

MemNN Memory Networks

NLP Natural Language Processing

OOV Out of Vocabulary

PE Positional Encoder

PPMI Positive Pointwise Mutual Information

QA Question Answering

RIG Relevant Information Generation

RNN Recurrent Neural Network

RNNs Recurrent Neural Networks

SQuAD Stanford Question Answering Dataset

SAN Stochastic Answer Networks

Appendix A

Comparison of Passage Summary Generation Methods

As stated in Section 5.1.1.2, there are two general approaches to generate the passage summary. The two approaches are called the first-order and second-order in this work.

The output formation is chosen to be $\left[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{h}}_i\right]$ and $\left[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{h}}_i;\ \boldsymbol{h}_i \circ \widetilde{\boldsymbol{h}}_i\right]$, which only include the passage representations and the passage summary. The query summary is excluded, since based on the empirical experience including the query summary may affect the observation on the performance of the passage summary. In another word, it is clearer to see the divergence of the performance of the two methods without the query summary's influence on the final results.

We denote the output formation $\left[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{h}}_i\right]$ as O_1 and $\left[\boldsymbol{h}_i;\ \widetilde{\boldsymbol{h}}_i;\ \boldsymbol{h}_i \circ \widetilde{\boldsymbol{h}}_i\right]$ as O_2 in the scope of this appendix. The predictive results of the two methods under O_1 and O_2 conditions are listed in Table 5. It is clear that the second-order method outperforms the first-order significantly in both O_1 and O_2 output formation conditions. The

Table A.1: The EM and F1 scores of the two passage summary generation methods

Method	O_1		O_2	
	EM	F1	EM	F1
First-order	31.4	40.6	31.0	40.3
Second-order	61.8	72.9	62.3	73.4

absolute improvement is 30.4 for EM and 32.3 for F1 under O_1, and 31.3 for EM and 33.1 for F1 under O_2. This indicates that the passage summary generated by the second-order method capture much more information than that by the first-order method. The first-order method seems not able to produce meaningful information on the interaction of the passage and the query, which the second-order way is able to provide a reasonable amount of information to produce the reasonable performance.

Take a look at the trends of the loss curves in the training stage illustrated in Figure A.1. There are four pairs of comparison in this figure: (a) compares the loss curves of the first-order method under O_1 and O_2 conditions; (b) compares that of the second-order method under O_1 and O_2 conditions; (c) compares that of the first-order and second-order methods under the O_1 condition; (d) compares that of the first-order and second-order methods under the O_2 conditions.

Figure A.1(a) shows that under both conditions the two models encounter a plateau period of convergence at around training step 10. In condition O_1 the model stays in the plateau till around step 20, while condition O_2 seems to help the model quickly find the direction to continue its convergence. After the two models surmount their plateaus with different amounts of time, the losses continue to decrease till they stop at the points of the similar loss levels.

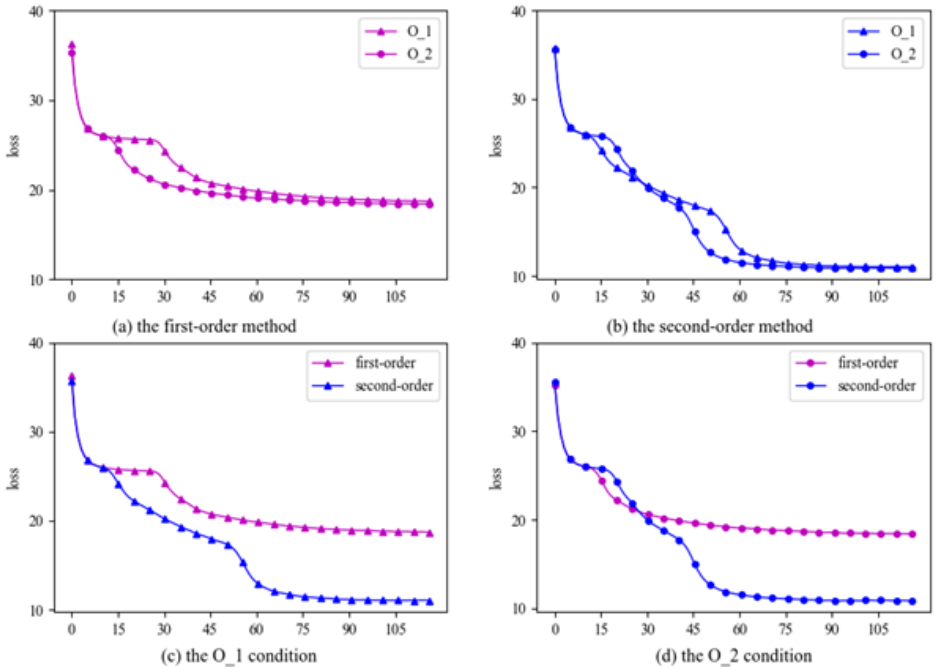


Figure A.1: Loss curves of the first-order and second-order methods under the O_1 and O_2 condition

Figure A.1(b) demonstrates that the the model with the second-order method shows the similar convergence patterns in the O_1 and O_2 conditions. Although during the converging process there is one duration that O_1 outperforms O_2 and another duration O_2 outperforms O_1, eventually both reach at the similar loss levels.

Figure A.1(a) and Figure A.1(b) display that no matter which passage summary generation method is adopted, the factor $\boldsymbol{h}_i \circ \widetilde{\boldsymbol{h}}_i$ in O_2 condition seems not be able to help the convergence of the model obviously.

Move to Figure A.1(c) and Figure A.1(d). The second-order method helps the model converge much more effectively than the first-order method in training in both O_1 and O_2 conditions. Both methods end up with significantly different loss levels where the second-order's is much lower than the first-order.

In summary, the passage summary generated by the second-order method contains more meaningful information that reflects the interaction of the question and passage, which enables to the model to converge more effectively than the first-order method.