Enhancing Natural Language Understanding using Meaning Representation and Deep Learning

Anupiya Nugaliyadde

This thesis is presented for the Degree of Doctor of Philosophy
Murdoch University

August 2019

Declaration

I declare that this thesis is my own account of my research and contains as its main content work which has not previously been submitted for a degree at any tertiary education institute.

Anupiya Nugaliyadde

Abstract

Natural Language Understanding (NLU) is one of the complex tasks in artificial intelligence. Machine learning was introduced to address the complex and dynamic nature of natural language. Deep learning gained popularity within the NLU community due to its capability of learning features directly from data, as well as learning from the dynamic nature of natural language. Furthermore, deep learning has shown to be able to learn the hidden feature(s) automatically and outperform most of the other machine learning approaches for NLU. Deep learning models require natural language inputs to be converted to vectors (word embedding). Word2Vec and GloVe are word embeddings which are designed to capture the analogy context-based statistics and provide lexical relations on words. Using the context-based statistical approach does not capture the prior knowledge required to understand language combined with words. Although a deep learning model receives word embedding, language understanding requires Reasoning, Attention and Memory (RAM). RAM are key factors in understanding language. Current deep learning models focus either on reasoning, attention or memory. In order to properly understand a language however, all three factors of RAM should be considered. Also, a language normally has a long sequence. This long sequence creates dependencies which are required in order to understand a language. However, current deep learning models, which are developed to hold longer sequences, either forget or get affected by the vanishing or exploding gradient descent. In this thesis, these three main areas are of focus. A word embedding technique, which integrates analogy context-based statistical and semantic relationships, as well as extracts from a knowledge base to hold enhanced meaning representation, is introduced. Also, a Long Short-Term Reinforced Memory (LSTRM) network is introduced. This addresses RAM and is validated by testing on question answering data sets which require RAM. Finally, a Long Term Memory Network (LTM) is introduced to address language modelling. Good language modelling requires learning from long sequences. Therefore, this thesis demonstrates that integrating semantic knowledge and a knowledge base generates enhanced meaning and deep learning models that are capable of achieving RAM and long-term dependencies so as to improve the capability of NLU.

ACKNOWLEDGEMENTS

It is with immense gratitude that I acknowledge the assistance, theoretical and moral support of my Principal Supervisor, Associate Professor Kevin Wong, who paved the way to compile this dissertation of mine, in fulfilling the requirements for a Doctor of Philosophy in Information Technology at Murdoch University. I am equally indebted to my Co-Supervisors, Associate Professor Ferdous Sohel and Dr Hong Xie for the invaluable knowledge shared and encouragement given to upgrade my academic forte to undertake this challenging study on "Enhancing Natural Language Understanding using Meaning Representation and Deep Learning". I am grateful to Dr Mohd Fairuz Shiratuddin, Advisory Chair, for the friendly academic environment provided so that I could utilise my time productively. Above all, I wish to express my utmost gratitude to all my academic supervisors for extending their time and energy in assisting the advancement of my academic career, which enabled me to serve the science of Information Technology.

I am indebted to Murdoch University, for granting me a full-time candidature for the Murdoch International Postgraduate Scholarship (MIPS), to undertake a PhD in Information Technology at the beginning of the 2016 academic year.

My student life at the Department of Information Technology, Murdoch, like all my dear friends, has been filled with great memories, in spite of the time demanding and stressful academic work. Special thanks to my friends for the wonderful memories and time spent over the last 42 months at Murdoch.

I am grateful to my parents, my darling wife, sisters, brothers-in-law, uncles and aunts for the moral support given when I felt an urge for consolation. I extend my heartfelt praise to my dearest family for their patience, sacrifice and understanding; for being together to help me through this challenging time.

Ariya, my darling daughter - the glittering star of my life, I dedicate my academic work to you, for always being with me and for sacrificing the comforts during your toddler years.

List of Publication Related to the Thesis

Journal Articles

J1. A. Nugaliyadde, K.W. Wong, F. Sohel, H. Xie, “Long Short-Term Reinforced Memory Network, Question Answering”, Transactions of the Association for Computer Linguistics. (TACL)
(Submitted)

Conference Proceedings

P1. A. Nugaliyadde, K.W. Wong, F. Sohel and H. Xie, “Language Modelling through Long-Term Memory Networks” presented at The International Joint Conference on Neural Networks, Hungary, 2019.

P2. A. Nugaliyadde, K. W. Wong, F. Sohel, and H. Xie, "Reinforced Memory Networks for Question Answering," presented at The 24th International Conference On Neural Information Processing Guangzhou, China, 2017, pp. 482-490.

P3. A. Nugaliyadde, K. W. Wong, F. Sohel, and H. Xie, "Multi-level Search of a Knowledgebase for Semantic Parsing," in International Workshop on Multi-disciplinary Trends in Artificial Intelligence, 2017, pp. 44-53.

P4. A. Nugaliyadde, K. W. Wong, F. Sohel, and H. Xie, "Enhancing semantic word representations by embedding deep word relationships," in Proceedings of the 2019 11th International Conference on Computer and Automation Engineering, 2019, pp. 82-87.

Summary of Contributions in the Thesis

Chapter	Contribution	Publications
Introduction		
Chapter 2: Background	This Chapter presents a comprehensive literature review regarding Natural Language Understanding (NLU) and a theoretical background to understanding the problems addressed in Chapters 3 to 5.	**J1, P1, P2, P3, P4**
Chapter 3: Enhancing Semantic Word Representations by Embedding and Semantic Parsing using Deep Word Relationships	1. Proposed the use of multi-level search for semantic parsing using Conceptnet 2. Integrating Word2Vec and Conceptnet to create a deeper word embedding	**P3, P4**
Chapter 4: Reasoning, Attention and Memory for Question Answering	Proposed a deep learning model which is capable of addressing reasoning, attention and memory	**J1, P2**
Chapter 5: Long Term Memory Network for Language Modelling	Proposed a deep learning model to hold a longer memory for NLU	**P1**
Conclusion		

Contents

List of Figures

List of Tables

List of Abbreviation

NLU	Natural Language Understanding
LSTM	Long Short Term Memory Network
RNN	Recurrent Neural Network
LTM	Long Term Memory Network
LSTRM	Long Short Term Reinforced Memory Network
RAM	Reasoning, Attention and Memory
RL-LSTM	Reinforcement Learning Long Short Term Memory Network
NLP	Natural Language Processing
QA	Question Answering
AI	Artificial Intelligence
DMN	Dynamic Memory Network
RRN	Recurrent Reasoning Network

Chapter 1

Introduction

Natural Language Processing (NLP) is the method of processing text data. The NLP methods are manifested in various applications, e.g. Chabot's, sentiment analysis, and text summarization. However, natural language understanding is a subcategory on NLP. Natural Language Understanding (NLU) is focused on making a computer comprehend language. NLU is focused on understanding complex language. NLU uses computational techniques to learn and understand human language [1]. This began around the 1950s. However, the complex structure and dynamic nature of natural language make NLU, an Artificial Intelligence, a difficult issue (AI hard) [2]. NLU has been prominently improved in the past decade [1]. Improvements in NLU have mainly been due to machine learning algorithms [3]. Machine learning gained interest because it was capable of learning through the available data.

Earlier, NLU used statistical methods. These methods showed highly accurate results [4]. However, statistical methods were unable to make proper generalization since the patterns were too large and complex to be mapped and coded. Machine Learning (ML) was thus introduced to overcome this issue. ML algorithms such as Support Vector Machines (SVM) and Conditional Random Fields (CRF) have shown promising results. These ML algorithms have failed in proper generalization and identification of complex patterns. On the other hand, Artificial Neural Networks (ANN) have been shown to produce better results than the SVM and CRF algorithms. ANN was unable to identify complex patterns which were visible in NLU. A model had to be used to identify the complex hidden patterns.

Inspired by humans understanding natural language, natural language understanding models focus on deep neural network/ deep learning models. The deep neural networks have a similar structure to neurons in the brains. Deep neural networks have a large number of neurons combined together, to act similar to a set of neurons. Deep learning techniques are one of the most widely used algorithms in ML due to its ability to identify complex hidden patterns and the availability of data. The increase of computational power and a large amount of available

data enable deep learning to be applied for NLU [5]. The capability of the features to learn automatically from the available data has also made deep learning a popular technique for NLU [5]. This dramatic improvement in NLU through deep learning has enabled new deep learning models to be introduced. However, before processing the text, it should be converted either into a logical or numerical format. There are two main approaches in converting the text suitable for processing via deep learning. One approach is to convert the text into the semantic parser, which is to create a parsed text, while the other approach is to convert the text into a numerical format.

Computational language understanding requires the conversion of text to computationally understandable structures [6]. Semantic parsing converts a natural language utterance into a logical form, which can be used to produce denotations [4]. The semantic parsed text holds the logical meaning of a given text [7]. In order to avoid complexity, the most common approach for text parsing is to parse sentence by sentence [8]. Most recent research on semantic parsing focuses on machine learning [3] [9], which requires supervision and training. Various techniques have been introduced to improve semantic parsing by reducing supervision.

The other approach is to convert the text into numeric data. These deep learning models require text data to be converted as numerical data in order to process and learn the features [10]. The conversion of text to numerical data requires the conversion to hold all the information presented in the context [10] [11]. Textual information is converted into vectors. These vectors are created for the words in a given context and are generally called word embeddings/ word representation. As holding a logical representation or numerical data on text is not sufficient to address NLU, the data needs to be processed/ analysed [12] [13].

Deep learning models are designed in order to address certain tasks, from image analysis [14] to learning sequential data patterns [15]. During the past decade, many deep learning models have been introduced for NLU, e.g. Dynamic Memory Network [13], Recurrent Relational Network [16] and Active Long Term memory Network [17]. However, in order to achieve NLU, a deep learning model requires to address Reasoning, Attention and Memory (RAM) [12]. A deep neural network should be capable of learning to memorise context, give attention to important factors in the context and reason through a given context, in order to properly understand the context. Therefore, ultra-specific deep learning models, which achieved either reasoning, attention or memory, e.g. reasoning network [18], attention-based network [19], and MEMEN (Memory based network) [20] have been introduced. Furthermore, language is sequential and these sequences are long. The sequential information in natural language holds the relationship with each of the words [21]. NLU depends highly on the sequence of text and prior information is required in gathering knowledge of the content. Memory networks are used

to hold sequential memory. In order to hold a sequence in memory, many types of memory networks are developed. The Long Short-Term Memory Network (LSTM) is created to carry the past inputs through the use of cell state and logic gates [15]. This prevents the LSTM from facing the exploding and vanishing gradient problem, similar to the Recurrent Neural Network (RNN). The LSTM will allow access to the older inputs so as to improve the output. The LSTM will also reduce the computational operational time and learn from the sequence. Some models were introduced to combine sequentially placed LSTMs to address long term memory [22].

1.1 Problems in Natural Language Understanding

Although NLU has gained momentum in the past decade, there are still gaps in the research. This thesis focuses on three issues still existing in many NLU applications.

1. Semantic Parsing and Word Embedding

Current NLU approaches focus on semantic knowledge (context knowledge), which is not sufficient to understand language. Semantic parsing uses the context to create a logical form for a given sentence and word embedding uses the context to create a vector representation for a given word. Word representations are inputs for the learning models in NLU tasks. However, to understand language, knowing only the context is not sufficient. Reading between the lines is a key component of NLU. Embedding word relationships, which are not represented in the context, enhance the word representation.

2. Reasoning, Attention and Memory (RAM)

Context-based question answering (QA) tasks require Reasoning, Attention, and Memory (together RAM) and QA tasks are one of the common methods normally used to validate NLU capability. Deep neural networks are capable of understanding non-linear hidden patterns in text. Recent research has used these capabilities of deep neural networks to support the learning of RAM. Currently, most developed deep neural networks do not focus on all three factors in RAM in one model. Deep neural networks would either focus on reasoning, attention, or memory (but not three) for the QA tasks.

3. Handling Longer Sequences

RNN, LSTM and other memory networks which contain memory are popularly used to learn the patterns in sequential data. Sequential data have long sequences that hold relationships. RNN can handle long sequences but suffer from the vanishing and exploding gradient problems. While the LSTM and other memory networks address this problem, they are not capable of handling

long sequences (50 or more data points long sequence patterns). Language modelling requiring learning from longer sequences are affected by the need for more relevant information in memory.

1.2 Aims and Objectives

A key factor in NLU is holding context-based knowledge and general knowledge in order to understand any given language. Capturing context-based knowledge and general knowledge and using that knowledge to improve NLU by enhancing meaning representation and reasoning, attention and longer sequential memory.

A word and sentence representation are required by the machine learning models as an input for machine learning models to properly understand the text. Given the word and sentence representations, machine learning models should use reasoning, attention and memory to understand the text. However, in order to properly understand language long sequences are required by a machine learning model. In order to show a proper flow, and to address the problems outlined above, the following are the aims and objectives investigated in this thesis.

1. Improving sentence and word representation by enhancing meaning representation

In order to understand a language, context and prior knowledge of the context is required. Therefore, in order to enhance sentence and word representations, an integration of context-based knowledge and meanings extracted from a knowledge base is investigated and introduced.

2. Addressing RAM in one model

RAM is a key factor in understanding language. A model should be able to learn to reason from the context, give attention and hold the context in memory. A model must hold the capability of learning to achieve all factors in RAM, so as to properly understand a given language. Therefore, one model to achieve reasoning, attention and memory is investigated and introduced.

3. Better handling of longer sequences

Language has long sequences which hold dependencies in the sequence. Thus, learning the sequence and holding the sequence in memory is a key aspect in language understanding. Therefore, it is necessary to investigate and develop a model which is able to hold a longer memory in order to understand language.

1.3 Contribution

1. Integrating semantic knowledge and a knowledge base to create an enhanced meaning representation for sentence parsing and word embedding

   This study presents a model to cover the gap in providing enhanced meaning representations, which combine semantic knowledge and knowledge graph extracted knowledge for a given word to create word representations. The resulting word representation contains more relationships and knowledge than semantic models. A higher weight is given to the semantic word representation to avoid an over-generalization of the final word representation.

2. Addressing reasoning, attention and memory for natural language understanding with an improved deep learning model

   This study will address the problem of having a model that can address RAM at the same time for QA tasks. The contribution is on proposing a novel deep neural network, named as LSTRM (Long Short-Term Reinforced Memory), which explores all three factors in RAM for context-based question answering tasks. In LSTRM, an adaptation of the Long Short-Term Memory Network (LSTM) is used to create a memory. Reinforcement Learning (RL) is used to enhance attention and reasoning. LSTRM is capable of reasoning and dynamic question-fact interaction (attention) on the memory it holds.

3. A learning model which has the capability to better learn longer sequences for NLU

   In NLU, using the relevant information in a long sequence is important. As such, this part of the study proposes a Long Term Memory network (LTM), which can tackle the exploding and vanishing gradient problems and handle long sequences without forgetting. The LTM

is designed to scale the data in the memory and give a higher weight to the input in the sequence.

1.4 Thesis Organization

The thesis consists of 6 Chapters and is organized as follows.

Chapter 1 provides a description of the current state of Natural Language Understanding (NLU) and the problems it faces. This Chapter describes the motivation for the proposed approaches in order to address the problems in NLU. The Chapter concludes by pointing out the contributions made in this thesis towards NLU.

Chapter 2 covers existing NLU techniques and provides an overview of the gaps in the research in this field.

Chapter 3 contributes to the enhancement of the meanings for word and sentence representation. In this Chapter, an approach to integrate both the knowledge acquired from semantics (context-based knowledge) and knowledge acquired from a knowledge base is presented. This Chapter demonstrates a semantic parser which extracts knowledge from Conceptnet in two stages, in order to generate enhanced meaning representations to support semantic parsing. In this Chapter, an enhanced word embedding, which integrates semantics and knowledgebase (ConceptNet) knowledge is introduced. The integration between the semantics and knowledge base generates an embedding which enhances the meaning representation. Chapter 3 focuses on enhancing the meaning representation to develop a semantic parsing and word embedding.

Chapter 4 contributes to RAM for question answering. In this Chapter, the Long Short-Term Reinforced Memory Network (LSTRM) is introduced. This Chapter discusses how LSTRM uses concepts of memory networks and reinforcement learning to address RAM. This chapter demonstrates the performance of QA tasks which require RAM.

Chapter 5 focuses on long term memory. This chapter introduces the Long Term Memory (LTM) network. It describes the LTM architecture and demonstrates its performance on language modelling. Through language modelling, the LTM network shows it is capable of learning long sequences.

Chapter 6 gives the conclusion and summary of the thesis. It also discusses the future work that would be conducted as a result of this research.

Chapter 2

Background

This chapter reviews the literature on NLU using deep learning to provide a foundation for the thesis. Section 2.1 discusses what NLU is, its branches and applications. Section 2.2 discusses the use of deep learning in NLU and its models. Section 2.3 discusses the problems in the current approaches in NLU. Finally, Section 2.4 summarizes Chapter 2.

2.1 Natural Language Understanding

Natural language is the language which humans use to communicate with each other. Natural language can be expressed in both spoken and text forms. Natural Language Understanding (NLU) is the practice of using computational models and techniques to learn and understand human language [1]. NLU is a subfield in Artificial Intelligence (AI) because it provides the capability of learning and understanding natural language similar to humans [2]. In order to address NLU through computational models, NLU is divided into 5 main steps. These sections are directly based on the Natural Language Processing (NLP) approaches. The main steps of NLU are lexical analysis, syntactic analysis, semantic analysis, discourse integration and pragmatic analysis [3] [23].

Lexical analysis is conducted in order to analyse the structure of words, phrases or a collection of words in any language. This is used to identify and separate text into meaningful sections. The syntactic analysis uses grammar in sentences, which arranges words in order to show the word relationships. The semantic analysis draws the contextual-based meaning of the sentences through the use of syntactic structure. Discourse integration analyses sentences with the use of past sentences. The knowledge of past sentences is carried forward to analyse the new sentence. Finally, pragmatic analysis is based on re-interpreting the context in order to understand the actual meaning of the sentences [3] . NLU models are based on addressing these main steps. However, the overall main contribution to NLU is through syntactic analysis, semantic analysis, discourse integration and pragmatic analysis. Pragmatic analysis requires

cognitive science and phycology [24]. In this thesis, only syntactic analysis, semantic analysis and discourse integration are considered for NLU, because they are the main contributors of NLU, as compared to the various steps described above. These steps can be validated through different applications in NLU.

2.2 Applications in NLU

There are many applications that use NLU [3]. Some of these applications are used in order to validate the steps in NLU. The main applications can be divided into two main categories, namely pre-processing for: NLU and NLU models respectively. The main pre-processing applications include: Parts of Speech (POS) tagging [24], Name Entity Recognition (NER) [25], semantic parsing [26] and word embedding [10]. The pre-processed information would be transferred to NLU models in order to understand the language. NLU models are Machine Translation (MT) [27], dialog systems [28], Question Answering (QA) systems [12] and language modelling [29]. The pre-processing models are based on lexical analysis, syntactic analysis and semantic analysis. Here, the NLU models are based on semantic analysis and discourse integration. Also, pragmatic analysis will not be discussed as it is beyond the scope of the thesis.

POS tagging involves tagging a token to words in sentences based on its context (adjectives, nouns or verbs). POS tagging is mostly done through rule bases. Tagging sentences support NLU, as it provides the NLU model with more details to process the sentence [24]. NER models tag words which are names and entities [25]. This is used in order to identify which words require attention, compared to the other words in a given sentence or paragraph. Tagging identifies key elements in a context but does not give a proper representation of the context. Semantic parsing is converting a given sentence into a logical format [26]. The logical format represents the meaning of the sentence. The logical format created through semantic parsing is used in NLU models in order to understand sentences clearly. Word embedding is the conversion of words into numerical representation (a vector) [10]. Word embedding holds knowledge about the word and its relationships. Therefore, word embedding holds meaning representations of the words. The most common word embeddings use semantic relationships to create the vector. Therefore, word embeddings hold semantic knowledge about the words in a given sentence.

The pre-processed output is passed on to the NLU models for the processing and generating of meaning or understanding. MT translates a text/ speech from one language to another [27].

The best MT model should be able to translate from one language to another without affecting the grammatical structure of the language. Another NLU model, dialogue systems, is a model which can carry out a conversation with a human by using natural language [28]. Therefore, dialogue systems are developed to understand the conversations that humans carry out and thereby respond accordingly. However, QA requires a model to understand a context and question in order to derive the answer [12]. Therefore, QA systems have a high dependency on understanding a given text. Finally, language modelling is the task of predicting the last word in a given sequence of words [29]. In order to predict the last word, the NLU model should capture the sequence and understand the context. Without holding the sequence and understanding the sequence, the NLU model is not capable of predicting the last given word in the sequence.

This thesis focuses on 4 main applications in NLU. Under pre-processing, semantic parsing and word embedding are focused upon. Under NLU models, QA and language modelling are the focal points. Semantic parsing and word embedding are pre-processing models which derive meaning using the context. Unlike other pre-processing models, rather than filtering words, these pre-processing models are context and meaning dependent. These models can directly influence language understanding. Therefore, semantic parsing and word embedding's meaningful representations for NLU is a key factor. QA is a challenging and complex task which requires complex NLU [12]. QA models require understanding through many aspects of NLU. Furthermore, language modelling requires a level of sequential understanding in order to predict the next word in the sequence. Language modelling should learn from the sequence and context [30]. Therefore, a good language model should be able to learn from the long sequence and understand it in order to predict the last word in the sequence. In order to address these NLU applications, models are developed using various techniques.

2.3 Models Enhancing NLU

Each of the 4 applications are discussed separately in this section.

2.3.1 Semantic parsing

Semantic parsing is the conversion of a given text into a logical format with meaning representation. Semantic parsing is developed in three different approaches: Logic-based, which converts the text into a first-order logic; Graph-based, in which the nodes are entities and Programming Language Conversion, which converts the text directly into a programming

language [31]. Similar to every AI technique, the early semantic parsing models first used a rule-based approach, which later changed to a statistical approach and finally, a supervised approach [31]. A different approach of supervision uses the knowledge base to create the semantic parser [8]. The use of the knowledge base allows the semantic parser to use knowledge, which is not available in the context, to identify the relationships and create an in-depth logical relationship between the words to create the semantic parse result [32]. Using context alone is insufficient to identify relationships between the words because the relationships are only based on the available context [33]. Therefore, a model based on the relationships between words to create a logical representation is more robust with the use of a knowledge base, as compared to using only the context.

2.3.2 Word embedding

Word embedding is the conversion of words in a context into the numerical format. Earlier models such as one-hot-vector used vectors to create identify words in a context. Although this vector representation holds value for a given word, it does not represent any other information of a given word. As such, such vectors are insufficient for NLU models to understand the context [34]. Therefore, the capturing of context-based information and the embedding of it in a vector for processing is crucial for NLU models [13]. The more information the word embedding process holds, the better it can support any NLU model [11]. Therefore, it is important that a word embedding model captures all the knowledge on a given word in order to support the NLU models.

These pre-processing models are key elements in NLU models which process the language and generate an output. The pre-processed input is then passed on to be processed, and the NLU model is thus created.

2.3.3 Question Answering

QA models require a good understanding of language. A QA model should be capable of going through a given context and understanding the context and question in order to answer it. Early adaptations of QA models used rule-based approaches and statistical models which focused on following a given probabilistic model or rule set in order to answer a given question [35]. However, with the introduction of large data sets, the rule-based and statistical models were not capable of performing well due to the dynamic and complex nature of natural language.

Therefore, machine learning models which learnt from examples were capable of capturing the dynamic nature of the language [1]. With the increase in computational power, deep learning has been used in natural language. These deep learning models have shown to outperform the other approaches used for NLU [36]. Therefore, many deep learning algorithms have been introduced to achieve QA [37] [16]. These deep learning models are structured in order to achieve specific tasks which support QA. Deep learning models in QA have shown to address most of the problems faced in understanding context and questions to generate an answer. Moreover, QA is not the only NLU application that has been highly improved through the use of deep learning.

2.3.4 Language modelling

Language modelling is the prediction of the last word(s) in a given sequence [30]. This sequence can be a sentence, a paragraph, a Chapter or a book. In order to predict the last words in a sequence, a deep learning model should capture the sequence and understand the context. Therefore, language modelling requires deep learning models to hold the sequences. Memory networks are the models that are capable of holding sequences and using them for processing. The most popular networks that have been used for language modelling are the Recurrent Neural Network (RNN) [30] and the Long Short-Term Memory Network (LSTM) [15]. LSTM and RNN use the past outputs as current inputs. Therefore, the current output is influenced by past output. Therefore, the sequential information is carried on through time. Because language is sequential and the relationships between words are key to understanding language, carrying forward past information has been a crucial element to achieving NLU.

2.4 Problems in NLU

Although NLU has shown tremendous improvements throughout the past decade due to the introduction of a large linguistic dataset, an increase in computational power and successful machine learning models [1], there are still many gaps left to be addressed. In this section, three main gaps in NLU (through the aspects of NLU applications) are pointed out.

2.4.1 The Problem of using only Context-based Knowledge for NLU

In order to understand language, most of the current models use only the available context [10]. However, only using the available context would not suffice. Similar to humans, in order to understand language, it requires the holding of knowledge which is not available in the context. Therefore, additional information, which is not available in the context, must be added on in order to address NLU [33]. Holding contextual knowledge can be used in order to understand a context from a broad perspective. Current pre-processing approaches process using only the available context and pass it onto the NLU models. Thus, these NLU models are only capable of learning from the available context and are consequently limited to the given context. As compared to context-based models, pre-processing models can be influenced to generate better results through additional information [8]. In particular, sentence-based pre-processing models (semantic parsing) and word-based pre-processing models (word embedding) can directly influence the results, by holding additional information, as compared to only holding context-based information. Therefore, in order to achieve proper NLU, it is essential to use pre-processing models to hold enhanced meaning representation (knowledge which is extracted apart from the contextual knowledge).

2.4.2 The Problem of Reasoning, Attention and Memory

Enhanced knowledge of sentences or words through pre-processing is not sufficient to address NLU. Reasoning, Attention and Memory (RAM) are the three factors that the NLU model should be capable of addressing in order to understand language [12]. An NLU model can be validated on RAM based on the model's capability of handling QA [13] [16]. In order to answer a given question from a context, an NLU model should be able to use its memory, focus its attention on important facts, reason through the context and question in order to generate the answer. Therefore, an NLU model should be able to address all aspects of RAM in order to properly answer a given question. However, current NLU models are capable of addressing only one or two factors in RAM. In particular, the Recurrent Relational Network focuses mainly on attention and reasoning [16], while the Dynamic Memory Network focuses mainly on attention and memory [13]. Models which focus on one or two factors in RAM will thus compromise on the other factor(s). Therefore, there is a need for a model which is capable of addressing all the factors in RAM.

2.4.3 The Problem of Long Term Memory

Language is sequential. This sequence is long-ranging; from sentences, paragraphs, as well as Chapters to books. These sequences hold dependencies within each of them. Therefore, in order to understand the language from the sequences, a model should be capable of holding long sequences [21]. The most common models which are capable of holding sequential data are the RNN and LSTM (commonly known as memory networks). Most of the other memory networks are based on the concepts of the LSTM. However, the LSTM controls its memory and is not capable of holding long sequences. The LSTM uses the forget gate to forget the past sequence if the network decides that it is irrelevant. Therefore, the LSTM limits its memory [15] and does not hold long sequences. Given a long sequence, the same condition is applied for many memory networks. However, the RNN does not limit its memory. Instead, when presented with long sequences, the RNN suffers from the vanishing or exploding gradient problem [38]. Therefore, the RNN fails at training when presented with long sequences. Long term memory can be validated through language modelling, in which a model should be able to learn the long sequence in order to predict the last word in that sequence. Current memory networks do not perform well in language modelling since they either forget the past sequence or get affected by the vanishing or exploding gradient. Therefore, a model which can learn long sequences, without being affected by the vanishing or exploding gradient, is required.

Chapter 3

Improving Semantic Word Representations by Embedding and Semantic Parsing using Enhanced Word Relationships

In this Chapter, the purpose is to enhance meaning representation at the sentence and word levels. At the sentence level, the objective is to improve the way semantic parsing is used, while at the word level, word embedding is key to enhance the words embedded by adding meanings. The main focus is the impact on NLU by using the knowledge base and integrating semantic information and the knowledge graph to improve NLU. In order to support NLU, knowledge from the context, as well as pre-conserved knowledge from the knowledge base, are integrated. This demonstrates the use of enhanced word relationships in order to address semantic parsing and word embedding. This Chapter is divided into two main sections. For the first half (from Sections 3.1 to 3.4), it focuses on the study of sentence-level meaning enhancement, whilst the second half (from Sections 3.5 to 3.8) focuses on enhancing word-level meaning representations.

This chapter is organized as follows. Section 3.1 discusses the knowledge base used for semantic parsing. Section 3.2 describes the models that are introduced for semantic parsing. Section 3.3 proposes two models for semantic parsing using knowledge bases. Section 3.4 compares the proposed models to the other knowledge-based semantic parsing models. Section 3.5 describes semantic word embedding and the need for enhanced word relationships. Section 3.6 describes the word embedding model integrating a semantic word vector and a knowledge graph. Section 3.7 describes the novel word embedding model. Section 3.8 describes the results compared to the other word embedding models. Section 3.9 concludes with the introduced semantic parsing models and the word embedding model.

3.1 Semantic Parsing Approaches

Computational language understanding requires the conversion of text to computationally understandable structures [2]. Semantic parsing converts natural language utterance into a logical form, which can be used to produce denotations [26]. These denotations are used for many natural language understanding tasks such as semantic role labelling, Word Sense Disambiguation (WSD), query processing, Information Extraction (IE) and Question Answering (QA) [39]. Most recent research on semantic parsing focuses on machine learning [40, 41], which requires supervision and training. Various techniques have been introduced to improve semantic parsing by reducing supervision [26]. In order to achieve this, it uses a knowledge base, ConceptNet [42], to fully remove supervision. ConceptNet is a knowledge-base contains words and its relationships. ConceptNet can be used to extract related words to given words through the various relationship connections [42].

Semantic parsing requires a combination of natural language with logic in order to construct the best results [41]. In order to construct a set of a proper logical predicate, a knowledge base is required. A knowledgebase is used to paraphrase and semantically parse a sentence [41]. Knowledge bases have the potential to achieve complex semantic parsing. ConceptNet 3 is utilised to manually extract concepts for the semantic parsing of sentences [8]. Even though ConceptNet contains an imperative set of concepts for semantic parsing and uses these concepts to remove supervision, two novel Approaches are further introduced. These two Approaches are explored as one method focuses more on concepts, whereas in the other, basic filtering take place depending on the relationships. The proposed techniques use ConceptNet 5 [42] as it is the updated version from ConceptNet 3. Furthermore, concepts are extracted from ConceptNet using two levels: i) the concepts are first extracted via the input sentence words (Level 1); ii) the extracted concepts are passed on to ConceptNet 5 and the concepts are then extracted for the second time (Level 2). These two levels of search create a multi-level search. The extracted concepts are matched with each other to generate a semantic parsed output. Also, the two Approaches are different as Approach 2 filters concepts at Level 1 (based on relationships) and do the concept extraction at Level 2. However, for Approach 1, the concepts are extracted and finalised in one Level. The aim is to use the concepts to create a logical format. As an example, for the search, “when was the United Nations founded?”, the proposed Approach focused on producing:

$$(!\mathrm{fb}:\mathrm{organization.date\_founded}\ \mathrm{fb}:\mathrm{en.united\_nations}) \tag{3.1}$$

The idea's novelty is based on using a multi-level search on ConceptNet, with automatic extractions of concepts from ConceptNet 5. The proposed Approaches hold information-rich semantic parsing results.

3.2 Semantic Parsing using Knowledgebase

Many semantic parsing approaches have been introduced in recent years. Most research has focused on machine learning approaches. Machine learning approaches such as deep neural networks [26] and SVM [32] have been used. Learning approaches for semantic parsing requires the learning of a pattern to extract information in order to create the semantic parser. Even though learning methods are capable of learning complex patterns and extracting information for a semantic parser, they are not capable of generating unavailable information in the text. NLU requires information available in the context, and knowledge which is not available but related to the context. The related information provides an insight into the context. Therefore, there should be a method of extracting unavailable information. A knowledgebase has the capability of providing the unavailable information through the input text.

Wordnet [43] is a knowledge base which holds lexical information. The lexical information held in Wordnet has the capability of providing the unavailable information [44]. Therefore, information-rich results are generated through the use of a knowledge base. ConceptNet 5 [42] is a complex knowledge base which holds a number of concepts for each word. ConceptNet 5 holds pre-existing lexical and semantic features for words and phrases. This also provides unseen information that cannot be extracted from the context. These concepts from ConceptNet hold more information which could be used to enhance semantic parsing [8], while dependency parsing is used to get the semantic relationships between the words. Conceptual information is gathered using the ConceptNet ontology. The concepts from the text are sent to ConceptNet and the semantics are then retrieved. This focuses on identifying concepts from the inputs and passing the concept to ConceptNet to find the relationships. ConceptNet is used in order to give the machine a better understanding of the sentence.

Through paraphrasing [41], ConceptNet allows the parser to identify more complex concepts to support semantic parsing. Through the use of paraphrasing, machine learning approaches require less training and supervision in semantic parsing as it provides insight by utilising more information to support the semantic parser. [26]. Therefore, the Approach aims to achieve a semantic parser which uses ConceptNet to remove supervision and training.

3.3 Proposed Knowledge-based Semantic Parsing

This research focuses on creating a semantic parser with the use of ConceptNet 5 using a multi-level search. The basic steps of the method are shown in Figure 3.1. The left panel shows the steps, while the right panel shows how an example is parsed. The first step is Part Of Speech (POS) tagging of the inputs. Using the tagged sentences, the phrases will be identified. The phrases and words are then searched in ConceptNet 5. The extracted concepts are validated and used for semantic parsing.

ConceptNet 5 is used to map words with the knowledge to support semantic parsing. Using the pre-existing knowledge of ConceptNet 5 for each word and word phrase, a knowledge structure (as shown in Figure 3.2) is generated. Furthermore, as shown in Figure 3.2, initially, the words and word phrases are searched from ConceptNet 5, which creates level 1 search results. The level 1 results are then searched in ConceptNet 5 in order to generate level 2 results. The results in level 2 are then used for the parser.

The methodology can be divided into two phases: generating the lexical representation and semantic representation generation. POS tagging is used to identify and process the information so as to create the lexical representation. The lexical representation is thereafter used to create

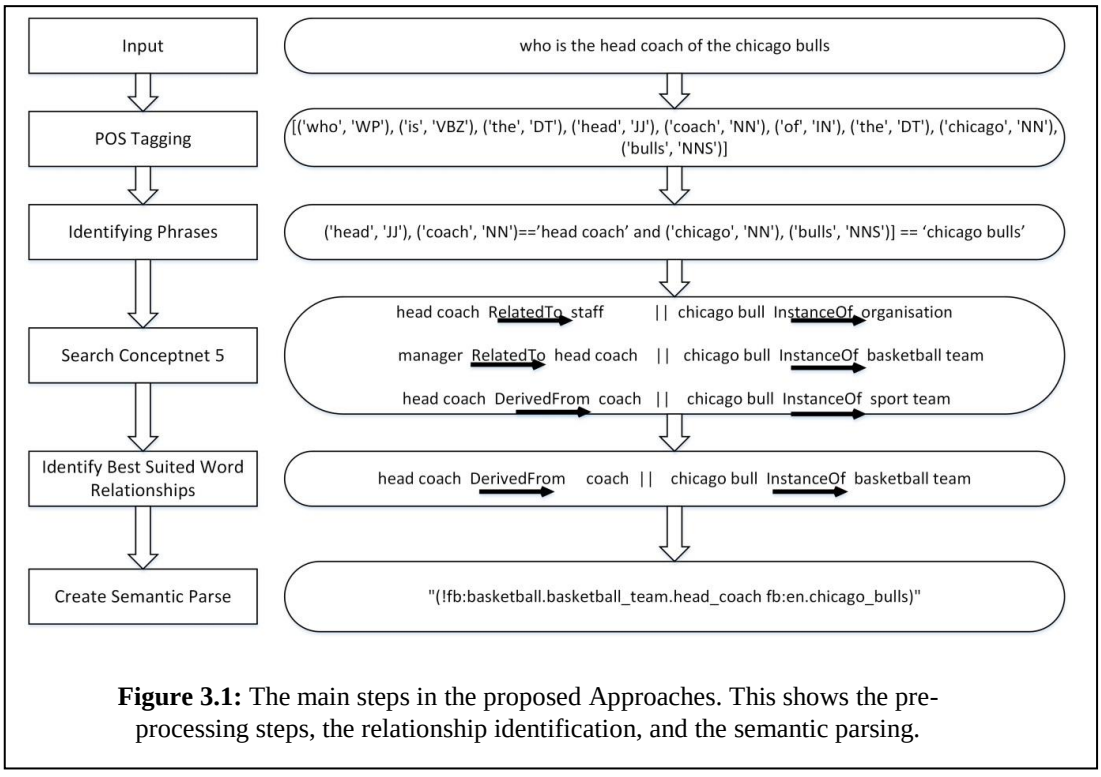


Figure 3.1: The main steps in the proposed Approaches. This shows the pre-processing steps, the relationship identification, and the semantic parsing.

the semantic representation through ConceptNet 5. The two Approaches presented in this Chapter differ in the final phase of generating semantic representation.

3.3.1 Lexical Phase

The main task in the lexical phase is POS, which is shown in Figure 3.1. This step is important as the tags for the words are used to identify the phrases from the words. A given query is tagged using the NLTK POS tagger since this method is a simple tagging method without the use of any learning method. The tagged query is used in order to create phrases. The initial step is used in order to combine words which should be searched together. In ConceptNet 5, as an example, if the words, "prostate" and "cancer" were searched separately for "what causes prostate cancer?", it would generate irrelevant results for the given query. When tagged using the NLTK POS tagger, it will generate:

$$[('what','WP'),('causes','VBZ'),('prostate','NN'),('cancer','NN')] \quad (3.2)$$

The tags are used to create the phrase. The following seven rules are used, which are a subset of the rules used in [8] for creating the multi-word expressions/ phrases that will be searched from ConceptNet 5.

Rule 1: Names which fall together, such as "Barack Obama", are combined together to create a phrase.

NP = NN + NN

Rule 2: Nouns which fall together, such as "farmer's market", are combined together to create a phrase.

NP = NN 's + NN

Rule 3: Nouns which follow the pattern of a noun followed by the plural of another noun ['NN', 'NNS'], such as "city officials", are combined together to create a phrase.

NP = NN + NNS

Rule 4: Names such as "Alice in Wonderland", which follow the pattern of ['NNP', 'IN', 'NNP'] are combined together to create a phrase.

NP = NNP + IN + NNP

Rule 5: A numeric value before a noun or set of nouns is combined together to create a phrase. Example: "2003 cricket world cup" should be considered a phrase.

NP = NUM + NN+NN+NN

Rule 6: A noun followed by a numeric value and a noun, similar to "Apollo 12 mission" is considered a phrase.

NP = NN + NUM + NN

Rule 7: A worded numeric value followed by a noun or nouns, as shown in the example "second grade", is combined as a phrase.

NP = NUM + NN

After the phrases are generated, stop words such as "a, an, is, at, be, by, for, is, in, were and was" are removed from the query. These stop words have little effect on the semantically parsed results. Since the stop words are removed after the phrases are identified, the stop words in the phrases are not affected. This is with reference to the stop words in word phrases such as "Alice in Wonderland". This is the identifying phrase (Figure 3.1), which combines the words to create noun phrases.

The generated phrases and words are searched automatically in ConceptNet 5. In order to search ConceptNet 5, the words and phrases are passed to the ConceptNet 5 API. This generates related words and their relationships (concepts) to the searched word/ phrase. ConceptNet 5 is searched on two levels. The levels are described as shown in Figure 3.2 for the example of "cat can run". The two levels created in the methods are based on the two stages of information extraction. The initial concept extraction for the information creates the first level, while the next concept extraction creates the second level. These Approaches do not use any handwritten rules in extracting the concepts as well as for semantically parsing the query. Two approaches are used to improve semantic parsing.

The step, "search in ConceptNet 5" (as shown in Figure 3.1), is achieved using these two Approaches. They are described as follows. The next sections explain identifying best-suited word relationships.

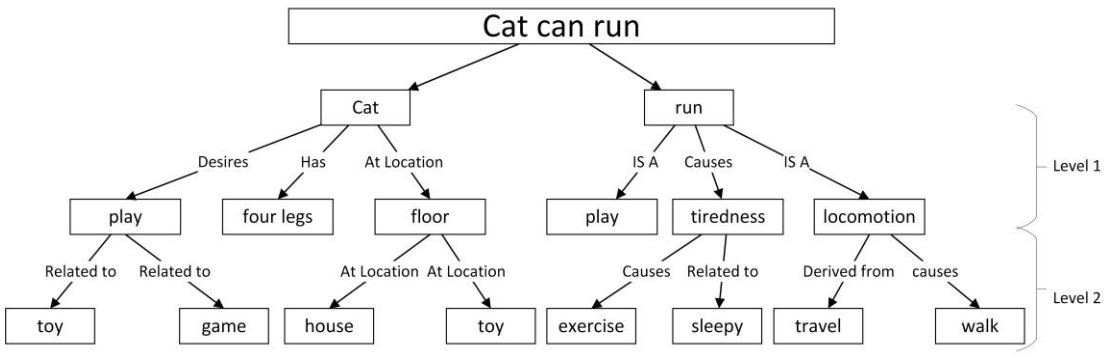


Figure 3.2: Example search of the Level structure generated. This shows a sample set of data extracted from ConceptNet 5, moving from Level to Level.

3.3.2 Approach 1

Figure 3.3 shows the steps in Approach 1. The search mentioned in Figure 3.3 is based on the extraction of concepts from ConceptNet 5. All the concepts extracted at Level 1 are searched again in ConceptNet 5. The concepts drawn from Level 2 are used to semantically parse the query. The relationships and related words are mapped with each other in order to generate the logical format. The logical format is based on the lambda format. The lambda expression predicate is created using these extracted concepts. The lambda expression variable is extracted from the query itself. The related words at Level 2 and their relationships are further used to enhance the semantic logical format to be generated. The relationships are identified by matching the words in the concepts extracted. As shown in Figure 3.2, "cat" and "run" both have "play" in common. With that information, it can be stated that the words have a semantic similarity.

In this Approach, the words from the query are not eliminated although they do not show any relationships at Level 1 of the ConceptNet 5 search. There are a larger set of concepts in Conceptnet 5 compared to the earlier versions. These extracted concepts can be used to improve on the semantic parsing, as well as further elaborate a sentence.

Figure 3.3: Approach 1 Block Diagram, shows the data flow from one step to the other.

3.3.3 Approach 2

As shown in Figure 3.4, the initial search for Approach 2 is similar to Approach 1, where Level 1 search results are extracted from ConceptNet 5 for each word or phrase in the query. The results generated from Level 1 for each word and phrase are then compared. The relationships with each word and phrase are identified, and the rest of the words are dropped. The identified related words and phrases are then passed to ConceptNet 5 to further extract the concepts. The generated concepts in Level 2 are used in order to create the semantic parser. The extracted

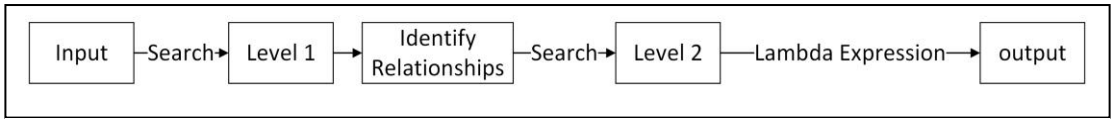


Figure 3.4: Approach 2 Block Diagram, shows the dataflow from one step to the other.

concepts are used in developing the parser using the words and their relationships in Level 2. The filter applied to the Level 1 results reduces the Level 2 search in ConceptNet 5. This reduces a substantial amount of search in ConceptNet 5 at Level 2, thereby reducing the computational intensity in generating the final semantic parse logical format.

3.3.4 Semantic Phase

The proposed Approaches use ConceptNet 5 without any filter, similar to [8]; therefore, it is information-rich. ConceptNet 5 provides words and their relationships to develop the logical format for semantic parsing. The logical format is created using the lambda format. The predicate of the lambda expression is used for the concepts extraction from ConceptNet 5 and pass the variable from the query. This can be explained further through the use of the example query, "what is serge made out of?", where the target lambda expression is:

$$(!fb : fashion.textile.fiber\ fb : en.serge) \tag{3.3}$$

The predicate "fashion.textile.fiber" is generated using the concepts drawn using ConceptNet 5 and "serge" is directly extracted from the query. Hence, this can be considered an improvisation of the calculations for semantic parsing. This also provides more information which is not revealed in the given query. Therefore, that information is not missed out when parsing the query.

3.4 Comparing Semantic Parsing Models and Analysis

The proposed Approach was tested on the Stanford Sentiment and Freebase datasets. These datasets comprise of 11 855 and 917 open domain queries, respectively. The queries are similar to queries which are passed to a question answering system or search engine. The two Approaches were tested using all the data in the Stanford Sentiment dataset, which includes both the training and testing datasets.

The queries in the datasets were converted into a lambda format. The lambda format supports a direct search from a corpus. The lambda format generates a logical format. Given the example, "when was starry night painted?", it generates the related words and their relationships from ConceptNet 5. The proposed Approaches will generate:

when $\xrightarrow{\text{RelatedTo}}$ time

paint Related To art
starry night InstanceOf artwork

based on the above relationships and generated words (3.4), which shows the logical format:

(!fb : visual_art.artwork.date_completed fb : en.starry night) (3.4)

For each query, the proposed Approach generated more than one logical expression, depending on the relationships. The semantic parsed results are not limited to be one-on-one, but instead, generate as many results as possible for each query.

The results generated in the experiments are shown in Table 3.1. In Approach 1, the results generated in Levels 1 and 2 are used separately for semantic parsing. This enables us to identify and observe the improvement in providing more concepts in order to achieve semantic parsing. The following equations were used:

True Positive = TP
False Positive = FP
False Negative = FN
Total Number of Queries = TNQ
Precision = TP/ (TP+ FP)
Recall = TP/ (TP+FN)
Accuracy = TP/ TNQ

Overall, the results in Approach 2 achieved a high recall rate of 94.39% due to the high level of concepts available. The concepts were not omitted; therefore, rich information is used for semantic parsing. Validating the concepts from the first level to the next reduces the false negative results. An improvement of 1.33% of the recall rate is visible from Approach 1 to Approach 2. The reduction of false negative can be stated to be important for tasks similar to question and answering query optimization, as it reduces the missing important information.

The results generated in Approach 1 show that when moving from Level 1 to 2, the recall improves from 91.96 to 93.29%. This validates the hypothesis that when provided with more concepts, the results will improve.

Although a high recall is shown, precision is reduced due to the high number of false positive results created in Approach 1. The issue is addressed by applying a filtration method in Approach 2. Following that, as shown in the results, recall and precision improve. Therefore, the use of a filter after the Level 1 search seems to generate better results.

Approach 2 can be compared with Approach 1 in the aspect of computational intensity. Approach 2 reduces the searches in Level 2 due to the relationship filter. Both Approaches have a computational complexity of $O(nc)$, where n is the processing complexity and c is the

number of iterations. However, it is evident that Approach 2 has less computational complexity as compared to Approach 1. The value of c in Approach 2 will be lower than in Approach 1 since the number of iterations is less. Since Approach 2 uses filters, this reduces the number of false negative results. As such, Approach 2 shows superior performance in both results and computational costs.

The comparison model uses ConceptNet 3 extracts the concepts manually for the selected 300 utterances of the Stanford sentiment dataset [8]. This is surpassed by automating the extraction of the concepts using ConceptNet 5. Furthermore, the two levels in ConceptNet 5 were searched to identify the semantic relationships and create the logical format for 11 855 queries. A semantic parsing framework, which requires both ConceptNet and WorldNet, is shown [7]. Here, ConceptNet is used in order to enrich WordNet's relationships using the method. As the proposed method only relies on ConceptNet 5, the approach holds fewer dependencies compared to the above method. Furthermore, both the words and relationships were extracted from ConceptNet 5 for semantic parsing. Therefore, using the relationships, the logical representation can be created with a stronger relationship than the other semantic parsing methods. The proposed method also shows the potential of improving the ability of the logical representation to hold additional information.

Table 3.1: Tested results on the Stanford Sentiment dataset using Conceptnet 5

	Approach 1		**Approach 2**
	Level 1	**Level 2**	**Level 2**
Precision	81.77%	82.37%	85.68 %
Recall	91.96%	93.29%	94.39%
Accuracy	90.95%	95.48%	97.04%

The results generated in Approach 2 were compared with [8] the results for semantic parsing. This is the closest match to the method. For the research, 917 queries were used. In contrast, the Stanford Sentiment dataset only used 300 manually selected queries. Furthermore, the proposed Approach extracted concepts automatically, in which the model compared against manually extracted the concepts from ConceptNet. ConceptNet 3 holds separate editions for English, Brazilian and Portuguese with the Global Mind [45]. ConceptNet 5 contains all the languages of ConceptNet simultaneously and a collection of large knowledge bases [45].

As of date, even via brute search, there is no semantic parsing technique using ConceptNet 5, although there are a few techniques in ConceptNet 3. For the sake of comparative analysis, the proposed Approaches were tested on ConceptNet 3 as well. The results are shown in Table

3.2. As shown in Table 3.3, the proposed Approaches outperformed the existing method of the knowledge base approach [8]. It should be further noted that 3 times more queries are used than the compared method. The method had 11 855 queries, while the compared approach used only 300 manually selected queries. Again, consistent with the results for ConceptNet 5, Approach 2 produced better results than Approach 1.

Table 3.2: Results comparison using Conceptnet 3, for the Stanford Sentiment dataset

	Poria's model [8]	**Approach 1**	**Approach 2**
Precision	92.21%	93.82%	94.91%
Number of queries tested	300	11 855	11 855

Furthermore, it can be confidently stated that unlike methods which require training [46], the methods can be used without any form of training. The proposed method was further tested with ConceptNet 5 on the freebase dataset and the WebQ, for a comparison with the current state-of-the-art method on freebase and WebQ Question-Answer pairs [26]. The compared model generates accuracies of 71.3% and 32.9%, respectively. For the Free917 dataset, the accuracies of 77.1% for Approach 1 and 79.2% for Approach 2 are generated. As for the WebQ dataset, the proposed approaches showed accuracies of 36.5% and 38.2% respectively.

Table 3.3: Accuracies for Free917 and WebQ datasets, with the comparison with Berant's model [26], Approach 1 and Approach 2.

	Free917	**WebQ**
Berant's [26]	71.3%	32.9%
Approach 1	77.1%	36.5%
Approach 2	79.2%	38.2%

ConceptNet 5 holds a large number of relationships compared with Conceptnet 3 for a given word. A comparison between the ConceptNet 3 results and ConceptNet 5 results cannot be fairly judged since Approach 1, and Approach 2 solely relies on the relationships words hold. Higher the number of relationships and the quality of the relationships words hold directly impact on the results.

3.5 Semantic Word Embedding and Need of Enhanced Word Relationships

Most Natural Language Understanding (NLU) models use word representations for their inputs [47]. Word representations are used for question answering [13] [39], machine translation [27], dialog systems [48], text understanding [49] and named entity recognition [50].

The commonly used word representations are those created using semantic representation and the lexical information of the local and global contexts [47]. A semantic word representation can demonstrate, "King to queen is to man to woman" (king-queen=man-woman) [51]. GloVe [10] and Word2Vec [11] are the more commonly used word embeddings, due to their capability of holding a semantic relationship. GloVe uses a global word co-occurrence count, which uses the statistic of the corpus and the semantics of the words in the corpus to create a vector for the words [10]. Word2Vec, a predictive model, uses the context words to predict the target word. Word2Vec uses a feedforward network [11] for the prediction. Both GloVe and Word2Vec use a large text corpus to create the context-derived word representations.

NLU relies on understanding the context by using information not expressed in the context, which normally requires reading between the lines. Communication requires more information than the current context, which is believed to be known by the listener [52]. Having the ability to retain more information in a learning model is shown to improve the performance over models that do not retain information [53]. Conceptnet is a general knowledge graph comprising of words and their relationships [33]. It contains the knowledge that can be provided for better word representations. However, there is a limitation in containing the semantic knowledge, which is also important in word embedding.

The next sub-sections of the Chapter present a model which combines semantic knowledge and knowledge graph extracted knowledge for a given word to create word representations. The resulting word representation contains more relationships and knowledge than semantic models. A higher weight is given to the semantic word representation in order to avoid an over-generalization of the final word representation. The proposed model is evaluated using SimLex-999 [54]. Furthermore, to illustrate the capability of differentiating association and similarity, word embedding is represented in 3D graphs in this Chapter.

The following sub-sections contribute the following: i) providing a method to generate a word representation that combines semantic knowledge and knowledge graph information, and ii) evaluating and demonstrating the ability of such a word representation in capturing both similarity and association between words.

3.6 Word Representation Models

Word representations have moved from lexical-based to semantic-based to provide information-rich word representations. Early word representations, like Latent Semantic Analysis (LSA) [55], hold the statistical representations on the corpus. LSA is a low-dimensional word representation based on the term-document frequency. These models fail at co-occurrences with general words (e.g. and, a, the, etc.).

Later, Positive Pointwise Mutual Information (PPMI) [56] and Hellinger's Principal Component Analysis (PCA) [57] were used to generate vector representations for words. The word representation was generalised and could be used for many natural language processing tasks. However, these cannot be used for language understanding [58]. The semantic-based vector representations have shown the potential to improve the understanding of learning models [13]. The semantic word representations produce dimensions of meaning and hold the distributed representation for the words. Continuous Bag-of-Words (CboW) and skip-gram models show linguistic patterns as linear relationships between the word vectors. This linear relationship is used to demonstrate the related words in a word representation. Learning-based natural language processing applications that use this representation would not receive the full meaning representation of the words, since it does not use the co-occurrence statistics of the corpus.

GloVe addresses the problems faced by the skip-gram model, which does not focus on the statistics of the corpus. In addition, GloVe has the capability of achieving analogy tasks. Therefore, based on the context, GloVe supports a higher level of meaning representation [10]. Word2Vec also uses the co-occurrence statistics from the corpus [34] and the semantics of the corpus. Word2Vec and GloVe are context-dependent. Therefore, depending on the context, the vector representation has drastic changes. The word representations and relationships are based only on the textual context. The word representation does not hold information that is not available in the context.

Conceptnet's knowledge graph is used to create word representations [59] and language understanding [53]. Furthermore, to support language understanding, Conceptnet is used for word representations, using PPMI and expanded retrofitting [60]. The PPMI creates the term-term matrix from the corpus text. The context of a term would be the terms that appear nearby in the text by using sparse matrices. The sparse matrices are used to derive word embedding. The expanded retrofitting, using ConceptNet, uses multilingual connections to learn English

words from a language translation. However, the word representations do not consider context-based statistics and analogy. Therefore, it loses the information embedded in a textual context.

Semantics is a key element to provide an understanding of the textual context. However, understanding the context does not rely only on the textual context [61]. NLU requires the capability of understanding analogy and corpus statistic. However, to gain further knowledge, an understanding of relationships that are not visible in the context, but known by the listener, is required. Conceptnet was developed with the intention of providing general knowledge for natural language understanding tasks [33]. Conceptnet provides related words and relationships connected to a given word. These connected words and relationships provide hidden knowledge of the textual context. However, Conceptnet does not provide complete information such as corpus statistics, so as to present a complete picture to understand language. Therefore, it is the aim of this Chapter to propose better word embedding.

3.7 Enhanced Relationship based Word Embedding Model

This section discusses the process of word embedding, which takes into account the semantic relationships and relationships which are not visible in the textual context. GloVe and Word2Vec use word embedding based on semantic relationships. However, word embedding is context-dependent. To avoid context dependencies and to allow a general representation, Conceptnet and Word2Vec are combined. This combination makes it possible to express similarity and association for word embedding, which is not available in Word2Vec and GloVe separately.

The proposed word embedding gives the word w a final vector representation of v_f, which encompasses a complete relationship of w with other words. v_f combines a semantic and context-independent representation for a given word. Therefore, this embedding is capable of representing the embedded meaning even though it is not visible in the textual context.

The embedding creates a 300-row vector for each word representation. The 300-row vector was chosen as it presented the best results from the experiments that were conducted (when comparing 50, 100, 300 and 500 vector sizes). The embedding consists of the two main types: semantic word embedding and generalised word embedding.

The semantic and knowledge graph embedding are combined to create the one-word embedding, representing both semantic and general word representations. These two embeddings are necessary in order to clearly understand words in a given context. Figure 3.4 presents a summary of the steps that will be described as follows.

3.7.1 Semantic Word Embedding

Semantic word embedding is used to embed the meaning expressed through the textual context. Semantic word embedding is generated through the Word2Vec (3.5). It embeds the word (w) using Word2Vec to create an initial word embedding, which holds the word similarity. The vector (v) holds the word similarity depending on the context that the word appears in.

$$v = Word2Vec(w) \tag{3.5}$$

However, the semantic approach only addresses the similarity between words and fails to handle the association between them. The association between words is not visible through the context. To address associations, as well as similarities that are not visible in context, Conceptnet [59] is used. The set of n related words W_{c_n} , to the word (w) are extracted from ConceptNet 5. Equation 3.6 demonstrates the results drawn from Conceptnet for the word w and generates related words W_{c_n}.

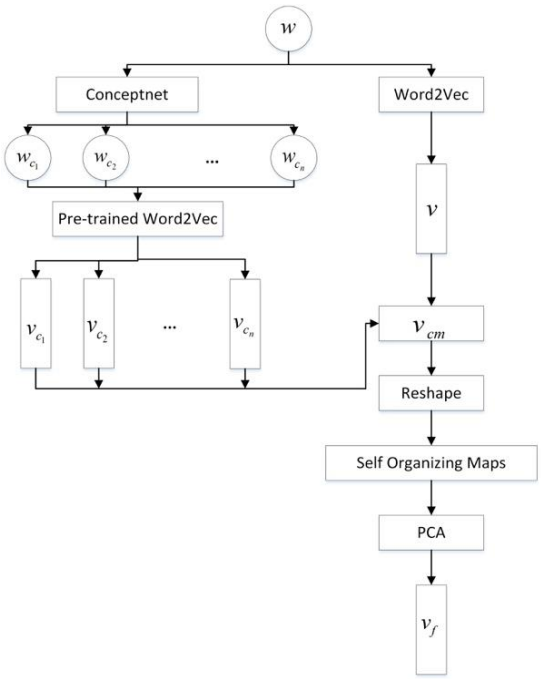


Figure 3.5: Overall methodology and data flow for the word embedding. The word $\boldsymbol{w}$ is passed on to ConceptNet and Word2Vec. Word2Vec creates the word embedding $\boldsymbol{v}$. The words extracted from ConceptNet $\boldsymbol{W_{c_n}}$ are passed on to a pre-trained Word2Vec to generate $\boldsymbol{v_{c_n}}$ and combine with $\boldsymbol{v}$ using $\sum_{i=1}^{n} \boldsymbol{v_{c_i}} . \boldsymbol{v}$, to generate $\boldsymbol{v_{cm}}$. $\boldsymbol{v_{cm}}$ is reshaped to a 2D matrix. The 2D matrix is passed to a SOM and the generated matrix is then passed on to PCA for dimensional reduction and transformed to the final vector $\boldsymbol{v_f}$.

$$W_{c_n} = Conceptnet(w) \tag{3.6}$$

W_{Cn} consists of n related words that are extracted from ConceptNet for the given word w, since each word w from ConceptNet would hold 20 related words on average, i.e. n=20. The extracted words w_{c_n} are converted into vectors v_{c_n} using a pre-trained Word2Vec. The Word2Vec pre-training was done on the Wikipedia dataset, and this created a 300-row vector for each word. Each of the extracted words is used to create its own vector (3.6). The model creates v_{c_n}, which is a set of n vectors for each word w and each one of these n vectors consists of 300 elements, as presented in Word2Vec. The vectors v_{c_1} to v_{c_n} are then added together.

$$v_{c_n} = Word2Vec(w_{c_n}) \tag{3.7}$$

3.7.2 Embedding Similarity and Association

Adding the extracted vectors $\sum_{i=1}^{n} v_{c_i}$ for the words w_{c_n} creates a generalised 300-row vector representation to the w. The combined vector representation is v_{c_n}. When extracting words from ConceptNet, it would contain similar words and associated words with w. Thus, adding the vector representations together would generate a generalised vector representation, which holds association and similarity relationships with w. Therefore, embedding the associated words and

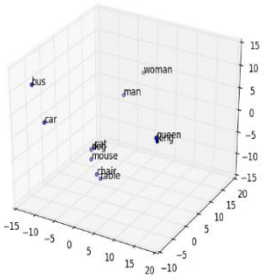


Figure 3.6: Proposed word embedding 3D representation for the words – "dog, mouse, chair, table, car, bus, man, queen, woman, king".

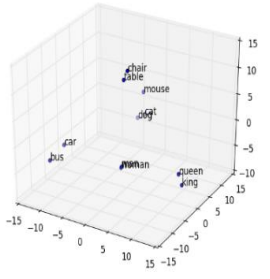


Figure 3.7: Word2Vec word embedding 3D representation for the words – "dog, mouse, chair, table, car, bus, man, queen, woman, king".

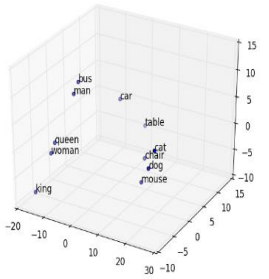


Figure 3.8: ConceptNet word embedding 3D representation for the words – "dog, mouse, chair, table, car, bus, man, queen, woman, king".

similar words through the use of ConceptNet generates a word embedding, which is generalised. However, a higher weight should be given to the semantic word representation directly from the word. For this, a Word2Vec representation v is created for the word w directly as well. For Equation 3.8, v is separately multiplied with the word embedding created through ConceptNet. The vector representation (v_{cm}) holds both similarity representation (through the Word2Vec and ConceptNet) and the association representation (through ConceptNet). Therefore, v_{cm} holds relationships that extend the semantic relationships that are extracted and generated from the available context. Therefore, v_{cm} for w would hold the association and similarity relationships.

$$v_{cm} = \sum_{i=1}^{n} v_{c_i} . v \tag{3.8}$$

The v_{c_m} is scaled in order to have the word embedding in a reasonable distribution. v_{cm} is further enhanced to generate the final vector representation (v_f) using Self-Organizing Maps (SOM) [62] that cluster words for the nearest neighbour. The nearest neighbour is identified using the Euclidean distance between the words. The SOM stage is mainly used to optimize the distribution by using SOM. The word embedding could produce a closer relationship within the neighbours. The v_{cm} 300-row vector is reshaped to a 2D metrics to pass it to the SOM. v_{cm} is considered a centre and the four nearest neighbours are based on the Euclidean distance that is passed through the SOM. In order to map v_{cm} to the Kohnen's layer, the vector v_{cm} is reshaped into a 2D matrix. The Kohnen's layer would run for 500 iterations, with a learning rate (a) of 0.005. This would produce a new 2D representation for each word. The 2D representation from the SOM will be transformed into a vector v_f using PCA. Through the use of PCA, the dimension reduction is applied to the 2D metric to create the final v_f. v_f is calculated for each word representation w_{c_n} as shown in (3.9).

$$v_f = \sum_{i=1}^{n} [v_{cm} + a(v_{c_i} + v_{cm})] \tag{3.9}$$

v_f creates a clear distinction between similar words, as compared to associated words.

3.8 Word Embedding Results Comparison

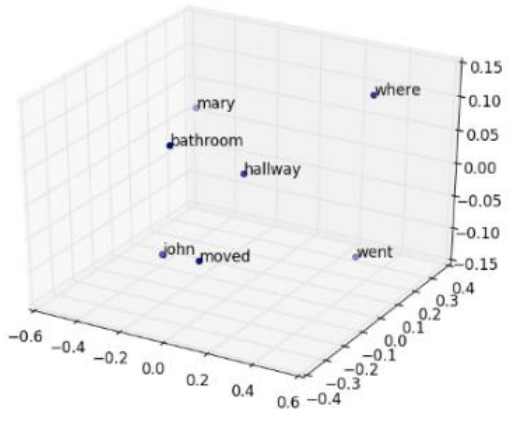


Figure 3.9: The proposed word representation for the context and the question – "Mary moved to the bathroom. John went to the hallway. Where is Mary?"

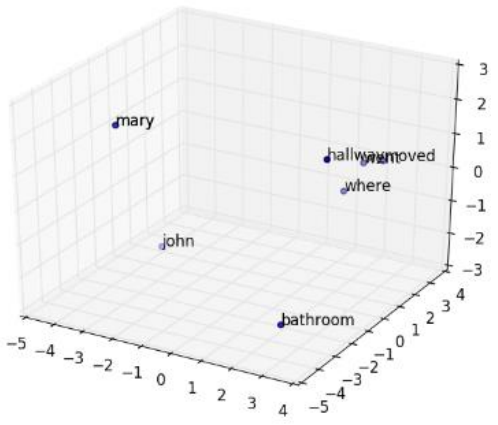


Figure 3.10: Word2Vec representation for the context and question – "Mary moved to the bathroom. John went to the hallway. Where is Mary?"

Table 3.4: Performance on SimLexc-999. The proposed word embedding is created using Wikipedia data. The proposed word embedding and [11] use ≈ 1000m, [63] uses ≈ 990m and [64] uses≈ 852m tokens.

Models	**Spearman's Correlation**
Word2Vec [11]	0.414
Deep Neural Networks with multitask learning[64]	0.268
Semantic similarity of words using concept networks [63]	0.76
Human Performance [16]	0.78
Proposed word embedding	**0.886**

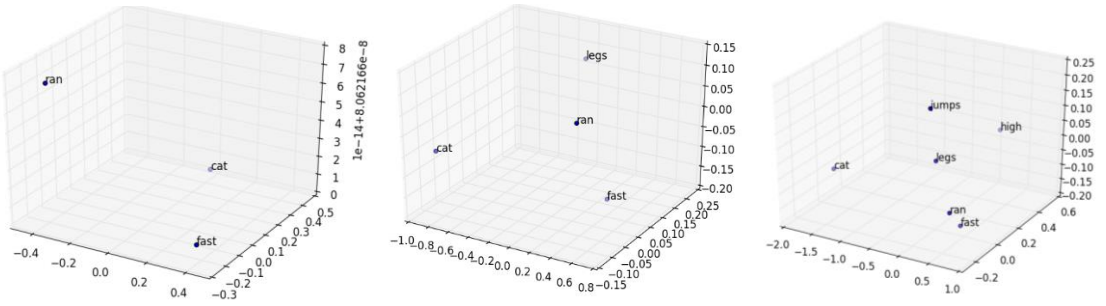
SimLex-999 is a resource used to evaluate models which generate meanings of words and concepts [54]. SimLex-999 captures the similarity between words rather than the relatedness and association. In order to achieve a high score, the word embedding should be able to capture similarity independently to the relatedness and association. Therefore, SimLex-999 has shown

to be a challenging evaluation model, as compared to WordSim-353 [65]. The proposed word embedding demonstrates the capability of differentiating similarity from association and relatedness, which is reflected in Table 2.4. The context used for word embedding in order to test on SimLex-999 is created based on Common Crawl data[1].The proposed word embedding is evaluated using SimLex-999 [54]. The SimLex-999 dataset to Spearman's ρ correlation between the words is calculated and presented in Table 2.4. Table 2.4 compares the proposed model with [11] , [63] and [64]. The Spearman's correlation shows that the proposed model is achieving higher Spearman's correlations as compared to the semantic word representations. This also shows that the proposed model can differentiate between similarity and association, as the proposed model uses ConceptNet to support generalization and Word2Vec to provide a semantic similarity. The normal 2D word embedding is extended [10] into a 3D word representation using the existing distant measure. PCA does a dimensional reduction to the 300-row vector by converting it to a vector of 3. The proposed word embedding in a 3D representation is shown in Figure 3.6, for "dog, mouse, chair, table, car, bus, man, queen, woman, king". The proposed 3D word representations place humans on one plane, vehicles on another plane and objects are placed in another plane with animals, but closer to the plane representing humans. Furthermore, the similarities between man to woman is less than queen to king. The proposed word embedding also captures more distinguishable representation than in Word2Vec (see Figure 3.7). Figure 3.8 shows the general vector representation created through ConceptNet. This shows a clear representation and that it is capable of distinguishing between the association and similarity of the words [54].

The 3D comparison between the Word2Vec (Figure 3.7) and ConceptNet-based word embedding (Figure 3.8) is shown. The relationships between "king to queen" and "man to woman" are similar, but the similarities between them are different (as shown in Figure 3.6) when compared to the word representation using Word2Vec (as shown in Figure 3.7) or using ConceptNet (as shown in Figure 3.8). The vector representation created using only ConceptNet is not context-dependent and shows a general representation which does not capture the similarity between words. This also shows that similarity, association or relatedness cannot be captured by pre-determined relationships which the words hold with each other.

[1] https://commoncrawl.org/

To demonstrate the application of word representation for sentences, a sample context from the bAbI dataset is used [12]. “Mary moved to the bathroom. John went to the hallway. Where is Mary?” is represented in 3D word representation using Word2Vec (as shown in Figure 3.9) and the proposed word representation is also shown in Figure 3.9. Observing the proposed models and 3D representations in Figure 3.9, “Mary” and “bathroom” are closer in the vector space. Figure 3.10 demonstrates that Word2Vec places “John” closer to the bathroom compared to “Mary”. Therefore, the proposed word embedding demonstrates that it supports one factor-based question answering tasks, as seen in e.g., the sentences, “*Cat ran fast. Cat has legs. Cat jumps high.*” The movement of words in the 3D space, when more sentences are added, shows the dynamic nature of the proposed word representation (Figure 3.11).



(a) One sentence **(b) Two sentences** **(c) Three sentences**

Figure 3.11: The word representation is dynamic, as seen in “Cat ran fast. Cat has legs. Cat jumps high”. Sentences are added one sentence at a time. There is the capability of achieving similarity and association based on the semantics of the sentences. Word2Vec captures the semantic representation, while Conceptnet generalizes vector representations in order to enhance it. Also, from (a) to (c), the vector representation changes with the new context. In the final representation (c), the separation of words in each axis is according to the relationship.

3.9 Summary

One of the three major objectives of this thesis, as highlighted in Chapter 3, is the investigation of the uses of enhanced knowledge, in the area of NLU. The use of both context-based knowledge and knowledge extracted from a knowledge base supports NLU. Also, as demonstrated, knowledge extracted from the knowledge base does support NLU. As such, the two methods, language parsing and word embedding, with the use of enhanced knowledge, work together to aid in NLU.

Firstly, two fully automatic approaches are proposed and presented for semantic parsing using rich information extracted from ConceptNet 5. Approach 1 uses the results generated through Level 2 for semantic parsing of the queries. Approach 2 filters the concepts generated from Level 1 and passes it onto Level 2. Both approaches produce information-rich results since a multi-level concept extraction is done via ConceptNet 5. The experiment results for Approach 1 showed that when more concepts are available, the precision, recall and accuracy improve. The results generated from Approaches 1 and 2 have recall values of 93.29% and 94.39%, as well as precision values of 82.37% and 85.37% respectively for the Stanford Sentiment dataset. Furthermore, when tested on Free917 and WebQ, the results for each approach were 77.1% and 79.2%, as well as 36.5% and 38.2% respectively. The results of Approach 2 indicated that with the use of a filter, the results also improved. The experiments also showed that with the use of more concepts, semantic parsing could be further improved. These show the capability of enhancing semantic parsing through the complex concepts extracted from ConceptNet.

Finally, a word embedding that uses context-based statistics, analogy and related meaning of words are proposed. This word embedding holds both context-based information via Word2Vec and related words via ConceptNet. The word representation places a higher weight to the context-based information to create the word representation, thus preventing overgeneralization. The word representation is evaluated using SimLex-999, which achieved a Spearman's correlation of 0.886. Furthermore, the proposed word representations are displayed in 3D to show the capability of distinguishing the associated words to similar words, as based on the context. The proposed word embedding is similar to the human performance of word similarity and association, by achieving a Spearman's correlation of 0.886; given that a human can achieve 0.78.

Chapter 4

Long Short-Term Reinforced Memory Network for Question Answering

In Chapter 3, models which integrate knowledge bases and semantic knowledge to extract the meanings of words and sentences are introduced. The knowledge to parse sentences and create embedding has enabled some models to outperform others which useless knowledge. Therefore, Chapter 3 highlights the importance of holding more knowledge to better understand natural language. However, holding more information is not sufficient to understand a language. Models should have the capability of learning and understanding text. Therefore, a model which is capable of using enhanced knowledge to address language understanding is required.

This Chapter focuses on the area of Question Answering (QA) for natural language understanding. In order to address QA, Reasoning, Attention and Memory (RAM) is required. This chapter introduces the Long Short-Term Reinforced Memory (LSTRM) Network to address RAM. The key aspects in understanding and generating the answer to a question using the context require a QA model to reason, hold attention and contain memory. LSTRM uses the concepts of the Long Short-Term Memory Network (LSTM) and Reinforcement Learning (RL) to address RAM. The earlier model, R-LSTM [39], is further enhanced in this Chapter. In order to fairly compare against other QA models, the embedding introduced in Chapter 3 is not used in the experimentation.

This chapter is organized as follows. Section 4.1 discusses question answering for natural language understanding. Section 4.2 discusses the overview of the models for reasoning, attention and memory. Section 4.3 proposes the long short-term reinforced memory network. Section 4.4 introduces the datasets for question answering. Section 4.5 demonstrates LSTRM's results and discussion. Finally, concluding remarks are given in Section 4.6.

4.1 Question Answering for Natural Language Understanding

Question Answering (QA) tasks require Reasoning, Attention and Memory (RAM) [12]. Recent models try to achieve RAM [37] [1]. However, a single model is not introduced, which is capable of achieving RAM. Reasoning requires a deep learning model to be capable of drawing novel inference from previously acquired knowledge [19]. A deep learning model requires the attention to remove unwanted information from the text and hold only important information [13]. Memory is required to hold the information for the deep learning model to process [12]. In order to answer questions using a given context, a deep learning model is required to achieve RAM. Deep neural networks are capable of learning non-linear hidden patterns [5]. This capability gives deep neural networks the potential to achieve many different types of learning tasks [12] [13] [14]. Hence, it would either focus on Reasoning [16], Attention [66] or Memory [12]. In the case of QA tasks, memory is a key component because they require the holding of information in memory, in order to identify the answer for the given question [5] [13] [66] [67].

Most QA-based deep learning approaches focus on memory-based deep neural networks [12]. These networks transfer information from the input at a time $(t-1)$ to the current input at time t. The memory in deep learning is the base for QA [39] [13]. In order to answer the questions, the information in the context is used. As the information sequence is long, it requires the deep network to hold the information in its memory, so as to allow it to find the answer in the sequence. Therefore, QA-based deep learning uses memory networks as a base [68]. The Long Short-Term Memory Network (LSTM) is one of the prominent memory-based deep neural networks that is used for QA tasks [15]. However, not all information in the sequence may be related to the answer. Removing unnecessary information from memory is beneficial in order to easily find the answer to the question from the context. The gates in deep neural networks filter unnecessary information [69].

The use of gates in the memory networks removes unnecessary information and gives precedence to the important information [13] [37] [12]. The precedence given to the important information will thereby lead to the correct answer. The gates are used to give more attention to the important information in the context, thus allowing the deep neural networks to find the answer easily [19] [66]. Therefore, attention and memory have co-relationships. However, holding a large memory and holding attention on it is a more complex task. Therefore, it can be seen that memory and attention do not have a clear separation line. Hence, most of the research focuses on attention, using minimal memory to support attention [66]. However, with the QA

tasks, attention alone is not sufficient. Reasoning based on the available context is required to properly answer a question from the context.

Reasoning is required in order to understand a question and infer the answer from the context. Recent work on deep neural networks has a focus on the reasoning for QA tasks [68]. Memory needs to be applied to reason [18]. Therefore, attention and reasoning are based on memory. Deep Reinforcement Learning (RL) is designed to achieve a given goal using the Markov Decision Process (MDP) [70]. RL is based on the science of decision making, allowing an algorithm to achieve a clearly stated goal [70]. RL has achieved state-of-the-art performance in goal-based tasks [71] [72]. Most common and current RL applications are based on manoeuvring through a given environment so as to achieve a goal [72]. In order to achieve a goal, the best policy is learnt by RL [73]. Learning the best policy to achieve a goal requires reasoning and decision-making over each available policy. Therefore, RL is capable of using deep MDP to achieve reasoning. Deep MDP is capable of reasoning through each state to achieve the goal state, using the best policy. A well-defined environment, states and a goal are key to success in most model-based RL [71]. It is a challenge to define an environment and states to apply RL in QA tasks. Since Markovian models rely only on the previous state to decide the next state, therefore, it produces results based on previous inputs rather than memory, which thus affects RAM [74].

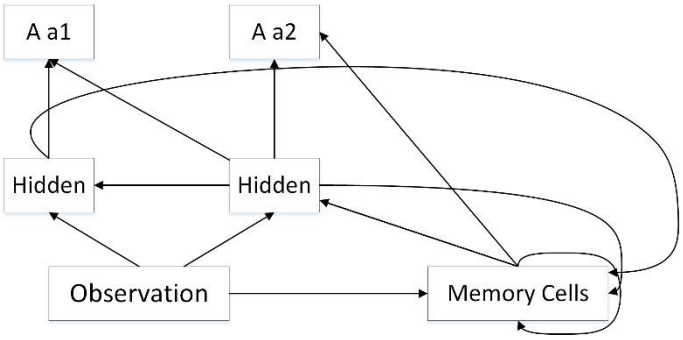


Figure 4.1: The pictorial representation of RL-LSTM, extracted from Bakker [75]. The arrows are fully connected weights. RL-LSTM was developed to handle non-Markovian tasks such as T-maze and pole balancing. Memory cells would remember past information and pass it to RL in order to generate the final output.

Early integration of memory and RL by Bakker [75] introduced model-free RL, combined with the LSTM (RL-LSTM) for non-Markovian tasks, e.g., T-maze and pole balancing tasks (Figure 4.1). The use of the LSTM handles long-term dependencies required to achieve T-maze and pole balancing tasks. RL-LSTM shows the possibilities of using memory to reason, e.g.,

deciding whether to go left or right from a T-joint in the maze [75]. The LSTM has complemented RL in achieving non-Markovian tasks (tasks which have hidden states), by adding the factor of memory, while RL handles the reasoning on the locations required to achieve T-maze and variations of pole balancing tasks. RL-LSTM cannot be used for QA tasks because the LSTM is used only to hold the past states and its rewards to predict the next reward, underpinning the memory required for reasoning and attention. RL-LSTM uses the LSTM to predict the next state based on the current state and reward, which supports the MDP decision on a higher rewarding state. RL-LSTM relies on an external environment and feedback to train the LSTM depending on the current state. This direct integration in RL-LSTM uses memory to support non-Markovian tasks. The memory from the LSTM remembers the past states of the RL bot, allowing the bot to move forward. The direct integration, although highly efficient for non-Markovian tasks, does not suit tasks which require RAM for QA tasks. Although RL is capable of reasoning and attention, while the LSTM is capable of providing memory and attention, direct integration of the models with separate functions does not support RAM.

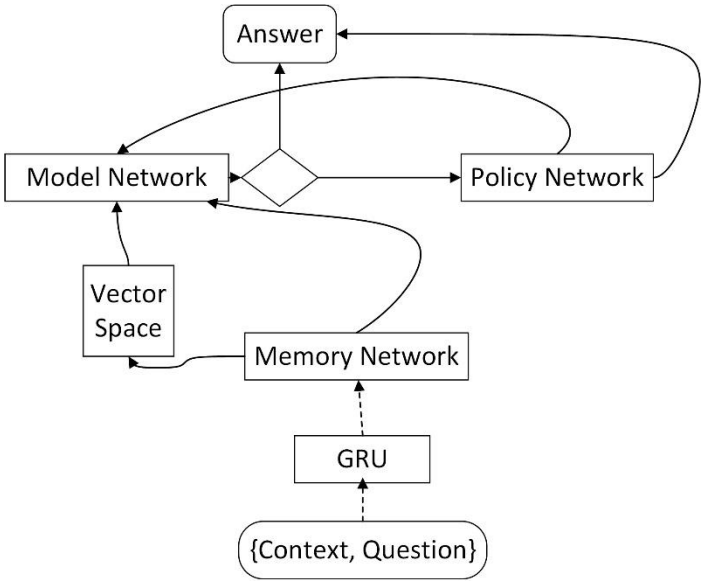


Figure 4.2: This presents the LSTRM network. The arrows indicate the data flow from one unit to another. {Context, Question} is the input and Answer is the output of the LSTRM network.

This chapter proposes the Long Short-Term Reinforced Memory (LSTRM) Network (Figure 4.2) which achieves RAM for QA tasks. LSTRM focuses on achieving RAM for context-based QA. LSTRM uses the concepts adapted from the LSTM to create the memory and combines it with concepts of RL for reasoning and attention for the context-based QA tasks. The

experiments show the possibility of RL acting on the memory to achieve an abstract level of RAM on context-based QA. This shows that it is possible to use the concept of RL to validate and correct the results generated through the memory-based network in order to produce accurate results. RL would validate and control the outputs of the memory. This can be used to achieve attention and reasoning to answer questions from a given context. LSTRM is tested on the following datasets: bAbI 1K [12], TriviaQA [76] and Quasar [77]. In order to achieve high accuracy in the tested datasets, LSTRM must be able to consider all aspects of RAM. LSTRM achieves optimal results using only half of the provided training data, which demonstrates its adaptability.

4.2 Overview of Models for RAM

QA requires complex natural language understanding capabilities [29] of word sense disambiguation, inference resolution [78], reasoning and handling dynamic context resolution [79] [80]. RAM are key components to a learning model for QA [81] [37] [12]. Memory-based deep neural networks, Dynamic Memory Network (DMN) [13], Memory Network (MemNet) [37] and Recurrent Reasoning Network (RRN) [68] have shown to achieve sections of RAM for the bAbI 1K dataset. This dataset has 20 different types of tasks and to achieve a task; the test accuracy must be above 95%. These networks show improvements in reasoning and attention. However, they do not achieve reasoning for dynamic question-fact interaction. DMN and RRN achieved attention and reasoning respectively but failed to achieve it for all the bAbI tasks. RRN failed at basic induction which requires reasoning. DMN achieved basic induction but failed in position reasoning and pathfinding. RNN's main focus is on reasoning, while DMN's main focus is on attention. Consequently, RRN failed in attention-based tasks, while DMN failed in reasoning-based tasks. Therefore, a proper focus on all aspects of RAM is the key to develop an overall network that could achieve all of the bAbI tasks. However, memory-based deep neural networks have failed in reasoning. These networks also require a large number of training data. The LSTM has been tested in these tasks and has failed to achieve similar results to RRN and DMN. This is due to the prominence in remembering more information than memory-based deep neural networks [39]. The LSTM comprises of three gates (input gate, forget gate and output gate) for the control of memory and output [1] [69]. The LSTM's input gate retains the most information passed. Therefore, the LSTM holds a strong memory, as compared to RRN and DMN. The overload of information reduces the potential to achieve RAM. The LSTM has more

information for reasoning and attention than any other memory-based deep neural network, but it does not properly utilize its memory. The LSTM's memory holds more information to support RAM, providing more context held in memory for reasoning and attention [39].

Attention-based networks are also used to achieve RAM. The Mnemonic reader is an end-to-end architecture and a combination of using BiLSTMs to query sensitive encoding, iterative aligning and multi-hop answer pointer for reading comprehension on TriviaQA [82] [76]. The multi-hop answer pointer moves from one sentence to another to identify the correct answer. It tries to identify the exact answer by traversing through the context. The query sensitive encoding aligns the attention on the query and encodes the context according to it. Iterative aligning changes the pointer to support the generation of the correct answer. This focuses on iterative attention sequencing for the correct answer by traversing through the context. The network does not focus on reasoning to generate the correct answer, which reduces the requirement of iterations of attention sequences.

The use of evidence aggregation for QA tasks has also shown promising results [83]. It has achieved an F1-score of 49.6% for the Quasar-T dataset [77]. The use of evidence in QA is focused on reasoning and re-ranking the answers according to relevant evidence, which links the question to the answers. This creates many candidate answers and re-ranks the answers using Bidirectional LSTM (Bi-LSTM). After the question is identified, the context is revisited multiple times. These deep networks have been successful in reasoning, but failed in achieving RAM, as compared to human capabilities.

RL is not intended for RAM or QA in many goal-based tasks but has exceeded human-level performance [71] [72]. RL has the capability of performing reasoning and attention to achieve the clearly stated goal [70] [84]. RL has the potential of achieving an abstract level of RAM in order to achieve a given goal. Most RLs are based on MDP. MDP focuses on optimizing actions on the environment to get the highest reward until the goal state is achieved [73]. MDP's actions can be considered as the reasoning to achieve a goal. RL's success depends on the environment, states and the goal state definitions [71]. Environment, states and a goal are well-defined in text-based games which supports RL [74]. This has made significant improvements when the environment and goals are specified. Defining an environment and the states are challenging in QA tasks. RL is capable of achieving reasoning and attention; given a well-defined environment, but memory has been a drawback in QA [74].

Bakker's RL-LSTM [75] was developed for non-Markovian tasks (the environment and its states are not visible to the model), e.g., T-maze and pole balancing (Figure 4.1). These tasks are RL tasks, which shows that memory dependencies do improve a task's performance.

Recurrent Neural Networks (RNN) and LSTMs are trained to approximate the value function for the RL. RL-LSTM shows the possibilities of reasoning and using memory to predict the future state. The LSTM has complemented RL on achieving the non-Markovian tasks by using the LSTM to use the past state and reward information to predict the next state. This ability cannot be used for QA. RL-LSTM uses the LSTM in order to remember past information so as to support the RL to learn from more than the previous state. LSTM would remember the past states and provide the information to the RL in order to make the predictions. RL-based prediction of rewards and states cannot be used on deciding an answer to a question. Using LSTM to remember the past states does not support the QA tasks because the QA cannot be as specialized as a non-Markovian task. RL's MDP approach cannot be used as a base for the QA task. Therefore, RL-LSTM cannot be used for QA.

RL and end-to-end LSTM can be trained for dialogue systems [28]. The LSTM is either trained using domain experts or through the use of RL. This shows that supervised learning is complementary to RL. Furthermore, the policy trained through RL accelerated the RL learning rate. This shows the possibility of using RL to improve and accelerate the training process. Through accelerated training, accuracy can be increased. The imitation of the real response through RL supports the acceleration of the training.

Table 4.1: Comparison of RAM with current deep neural networks. X represents the models achieved the task and _ indicates the model has not achieved the task.

Technique	Reasoning	Attention	Memory
DMN	_	X	X
RRN	X	_	_
Reinforced Reader Ranker	X	X	_
MemNet	_	X	X
Evidence Aggregation for Answer Re-Ranking	_	X	X
LSTRM	X	X	X

Table 4.1 shows the types of deep neural networks for QA and compares their capabilities of achieving RAM. Each type of memory-based network has either achieved reasoning or attention based on memory. It shows the gap in the memory networks, which are capable of achieving RAM. Figure 4.2 shows the LSTRM's architecture with the dataflow. The overview of LSTRM is distinguishable from the other comparable model, RL-LSTM. Figures 4.2 and 4.3 also show that LSTRM has a different structure than simply combining RL and the LSTM.

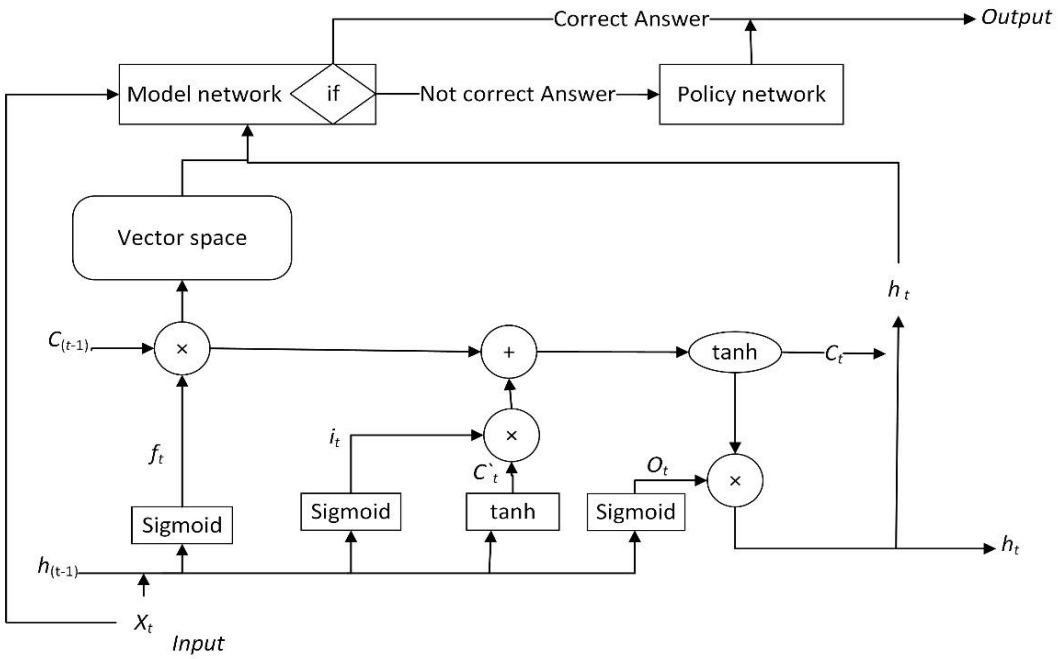


Figure 4.3: An overview of the proposed LSTRM network model with the dataflow. X_t is the current input. A vector space is created from the input gate f_t . The vector space is used as the environment for RL. h_t output is validated by the model network using the vector space environment. If the model network decides this answer is correct, the output is passed out. Otherwise, the policy network, which is based on MDP, will correct the output.

4.3 Proposed Long Short-Term Reinforced Memory Network

4.3.1 Overview

The aim of LSTRM is to utilize a memory of deep neural networks, policy network and model networks in RL to achieve RAM for QA tasks. LSTRM uses concepts adapted from the LSTM for memory and concepts from RL's policy networks and model networks for reasoning and attention (Figure 4.3). The model and policy networks are used to apply RL concepts in LSTRM. This has shown the capabilities of achieving an abstract level of RAM for context-based QA. Context x_t (as shown in Figure 4.3) is passed to the memory and the input gate f_t would remove the unwanted words [15]. The filtered-out words are used to create the environment (vector space) for the model and policy networks. The vector space is created using the filtered words (Figure 4.4). The memory of the networks predicts the answer to the question derived from the

context. The model network (Figure 4.5) learns to replicate the final result on the environment. If the model network determines that the answer given by the network's memory is incorrect, the policy network would act on the vector space and identify the goal state (the answer) for the given question. The answers are traversed through the environment (vector space) of words to find the most relevant answer. The response is generated through an MDP in the policy network [85]. However, this could also be achieved using the Monti Carlo tree search [71] [72] (using the vector space as the environment), but the performance lags slightly behind deep model-based policy network.

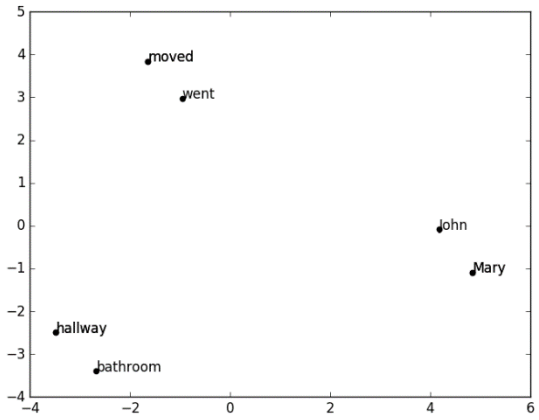


Figure 4.4: The states representation of "Mary moved to the bathroom. John went to the hallway." The words are the states, and the vector space is the environment. The RL would change from the states in the vector space to achieve the goal state.

4.3.2 Pre-processing for LSTRM

LSTRM is trained with sets which contain context, questions and answers. It learns to predict the answer to the question from the context. Words are encoded as input for LSTRM. Word encoding can be achieved using recurrent neural networks. The input method for LSTRM is similar to the one used by [13]. In order to maintain the completeness of this Chapter, the input is described below.

In order to create the input for LSTRM, the sentences are concatenated as a long word list. The pre-trained GloVe model is used to embed the words [10]. The words are embedded using a Gated Recurrent Neural Network, (GRU) [86]. At time t, there is an input of x_t, and the hidden state is h_t. The GRU can be defined as:

$$z_t = \alpha(W^Z x_t + U^Z h_{t-1} + b^Z) \quad (4.1)$$

$$r_t = \alpha(W^r x_t + U^r h_{t-1} + b^r) \quad (4.2)$$

$$\widehat{h_t} = \tanh(W x_t + r_t U\, h_{t-1} + b^h) \quad (4.3)$$

$$h_t = z_t . h_{t-1} + (1 - z_t) . \widehat{h_t} \quad (4.4)$$

where "." is the element-wise product, W^Z, W^r, $W \in R^{nH \times nI}$ and U^z, U^r, $U \in R^{nH \times nI}$. Dimensions n are hyperparameters. z_t is the input vector and the GRU gates would do convert the vector as shown in (4.1) to (4.4) to generate the final vector h_t which is passed on to the LSTRM.

Pre-trained Word2Vec [10] (GloVe and Word2Vec was pre-trained for the same Wikipedia datasets) embedding was also used to embed the input for LSTRM. Both pre-trained models generated similar results for LSTRM.

4.3.3 LSTRM's RAM Architecture

The embedded input is transferred to the memory section of the network. The f_t gate removes only the unwanted information (4.5). The selected information is used for the following two objectives:

1. To pass on to the next time *t+1* through the memory combined with the cell state C_{t-1}.
2. To create a vector space for the model and policy networks.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \quad (4.5)$$

f_t is combined with the previous cell state (C_{t-1}) in order to add past information to generate the output. LSTRM handles sequential data, which is important for QA and natural language understanding tasks. The cell state (C_t) carries on the previous output and joins it together with the current input. This allows the information to pass on through each time stamp.

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \quad (4.6)$$

$$C'_t = tanh(W_c[h_{t-1}, x_t] + b_c) \quad (4.7)$$

$$C_t = tanh(f_t C_{t-1} + i_t . C'_t) \quad (4.8)$$

The memory section's output (h_t) is created in order to pass it on to the next time stamp and model network. It is created by combining C_t and O_t from Equation 4.9.

$$O_t = \sigma_o(W_o[h_{t-1}, x_t] + b_o) \tag{4.9}$$

$$h_t = O_t.\tanh(C_t) \tag{4.10}$$

$$Vec_t = f_t C'_t \tag{4.11}$$

The vector space Vec_t (Figure 4.4) used by the model and policy networks is created by Equation 4.11. Vec_t is used to create the environment for the model network to learn (Figure 4.5). While training, the model network receives the correct answer, as well as the answer generated from the memory section of the network. The model network then learns to validate the answer using the vector space. If the model network recognizes the memory network's result as correct, it will pass the answer as the output. Otherwise, it would pass the predicted answer to the policy network to predict the actual answer. The model and policy networks treat the words as states (s). The policy network decides on which actions (a) should be taken to achieve the final goal (g) (answer). The policy network also decides on which action is taken to achieve the best reward (r). r is 1 only if the policy network achieves the correct answer. The discount factor γ is set at 0.001. Therefore, the policy network targets to maximize r. The policy would create the best action plan to achieve the correct answer g. The model network learns by observing a the Vec_t and s transitions in the Vec_t. This observation is used to create the model in the model network. Equation 4.12 is used in order to learn the dynamic nature of Vec_t. The model network learns to minimize the value generated by Equation 4.11, in order to accurately identify g, as well as to support the learning model to learn faster.

$$f(s, a) = \sum_t \left|\left| f(s_i, a_i) - s'_i \right|\right|^2 \tag{4.12}$$

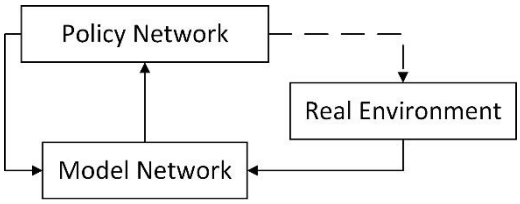


Figure 4.5: The model network interacts and learns from the real environment (vector space). RL interacts with the model and generates the results to the real world. The direct arrows indicate continuous data transfer and the dashed arrow indicates occasional interactions. The policy network interacts with the real world when generating output.

The policy network (deep neural network) decides on what the next a should be for the given Vec_t. Its aim is to maximize r by reaching g. The policy network learns both from the

correct answers generated by the memory, and the wrong answers by learning to correct the answer based on the responses of the environment. However, the policy network cannot go through all the states because the discount factor would reduce the value of r with the number of states moved. Equation 4.13 shows the policy network's functionality; the target is to achieve the highest r. The policy that achieves the highest r is considered the optimal policy $\dot{\pi}$ in achieving a given g.

$$\dot{\pi} = \arg max_{\pi} [\sum_{t \geq 0} \gamma^t \ r_t | \pi] \tag{4.13}$$

When training, the policy and model networks, they are capable of achieving QA, by validating and correcting the memory network results.

4.4 Dataset

The proposed LSTRM was tested on the bAbI 1K [12], TriviaQA [76] and QUASAR [77] datasets. LSTRM achieved state-of-the-art results for all three datasets: an accuracy of 99.52% and 71.2% for bAbI 1k and TriviaQA respectively, and an F1 score of 81.2% for QUASAR.

The amount of training data was increased by adding new, randomly selected data until all the training data was used to train the LSTRM network. LSTRM was tested ten times for each training dataset selection and the results were obtained by averaging the results for each training dataset. The test and training datasets are separated, in which the test data is not visible while training.

4.4.1 bAbI 1K Dataset

This dataset has 20 different tasks. Each task has sets of context, question and answer. There are 1K training sets and 1K testing sets for each task. Each task has a separate goal that the agent is required to achieve. Humans can generally achieve these tasks. The tasks are noiseless [12]. However, for each task to be stated as achieved, the agent must get an accuracy of 95% or above.

These tasks range from simple to complex supporting factors, argument relations, yes or no questions, counting questions, direction-based questions and indefinite questions. Each task is challenging and complex. These tasks require a high level of context-based QA for RAM. Therefore, if a given model can achieve all tasks, it can be stated that the model is a strong model

for achieving RAM on the context-based QA tasks. RRN managed to achieve an overall accuracy of 97.02% [12].

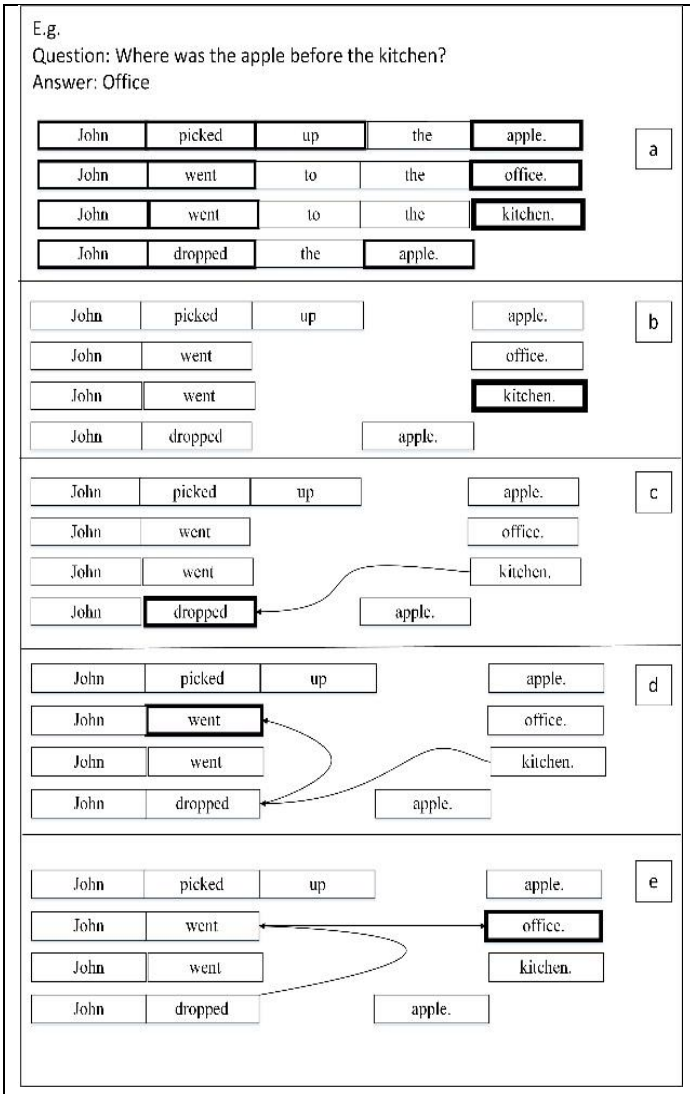


Figure 4.6: The possible steps of LSTRM in generating the final answer for the question. The thicker boundaries indicate the results each step generated.
a. shows the results that are kept in memory in the memory network (this has forgotten the unimportant words).
b. shows the final result (kitchen) generated by the memory network. Since the answer is not correct, the model network would pass it on to the policy network.
c. The policy network would move from state (word) to state (word).
d. The policy network moves the predicted answer to other states (words) until it reaches the correct answer.
e. The policy network moves to the final answer and this answer is passed as the final output of LSTRM.

4.4.2 TriviaQA Dataset

This dataset is aimed at reading and comprehension, holding triplets of 650K questions, answers and evidence. 95K pairs are authorized by trivia enthusiasts [76]. The information required to answer a question is mostly scattered over multiple sentences. Because the dataset is diverse, it is divided into two sections: web and Wiki. Two baseline algorithms are used to test it: a feature-based classifier [87] [88], achieving Exact Match (EM) of 23% and a neural network BiDAF [66], achieving EM of 40.32%, F1 of 45.91% for the Wiki and EM of 40.74% and F1 47.05% for the web that performs well in SQuAD [89].

4.4.3 Quasar Dataset

Quasar has two large-scale datasets which have two different objectives:

1. QUASAR-S (contains software-related questions, e.g., words similar to CAN may mean “Control Area Network”, rather than the general meaning of “doable”)
2. QUASAR-T (contains trivia questions drawn from the Reddit collection)

These questions can be divided into a single token, short or long answers [77].

For QUASAR-S, the best performing baseline is the BiRNN language model, which achieves 33.6% accuracy. The GA Reader model [90] achieves 48.3% accuracy on the set of instances for which the answer is in the context [77]. For QUASAR-T, both neural network models significantly outperform the heuristic models, with BiDAF getting the highest F1 score of 28.5%. However, this is significantly below the human performance of 60.4% and 60.6% for each dataset [77].

4.5 LSTRM's Results and Discussion

In order to compare and validate the results, all models were set-up with the same input module (same input embedding). The same training and validating data are used for all deep learning models. All models were trained from scratch in order to be used for proper comparison. LSTRM is trained on the full training dataset; however, the LSTRM converges to the highest accuracy with half the training data and does not increase, drastically. Therefore, LSTRM converges faster than other deep neural networks.

4.5.1 Generating Results from LSTRM

An example of how the LSTRM generates the answers are shown in Figure 4.6. The sample is taken from the bAbI dataset. An assumption is made that memory does not generate the correct answer; therefore, the policy network and the model network would interact and generate the final answer.

TABLE 4.2: bAbI 1 k results for each tasks compared with LSTRM, RRN and DMN. RAM shows which category the tasks fall under: R= Reason, A= Attention and M= Memory.

Tasks	**RAM**	**Accuracy (%)**				
		1000 sets for Training data			**500 sets for Training data**	**450 sets for Training data**
		DMN	**RRN**	**LSTRM**	**LSTRM**	**LSTRM**
Single Support Factor	M	100	100	100	100	100
Two Supporting Facts	M	98.2	99.7	99.5	99.5	99
Three Supporting Facts	M	95.2	96.5	99.8	99.8	97.5
Two Argument Relations	R	100	100	99.7	99.5	98.2
Three Argument Relations	R	99.3	99.6	99.6	99.6	99.1
Yes/No Questions	R	100	100	100	100	100
Counting	A	96.9	100	99.6	99.6	98.2
Lists/Sets	A	96.5	100	99.6	99.6	98.6
Simple Negation	R	100	100	100	100	99.5
Indefinite Knowledge	R	97.5	100	99.9	99.9	99.3
Basic Co reference	R	99.9	100	100	100	100
Conjunction	R	100	100	100	100	100
Compound Co-reference	AM	99.8	100	100	100	99.6
Time Reasoning	R	100	99.9	100	100	99.5
Basic Deduction	R	100	100	100	100	100
Basic Induction	R	99.4	45.1	97.8	97.5	96.2
Positional Reasoning	R	59.6	100	98.5	98.5	97.1
Size Reasoning	R	95.3	99.7	98.5	98.5	98
Path Finding	R	34.5	99.9	98.5	98.5	97.1
Agent's Motivations	RA	100	100	100	100	100
Mean Accuracy		93.6	97.02	**99.55**	**99.52**	**98.85**

4.5.2 Results for 1K bAbI Dataset

Reinforced Memory Network (R-MN) has shown an accuracy of 99.02% in achieving all tasks, while RRN has shown an accuracy of 97.02% in achieving only 19 tasks (only 19 tasks achieve an accuracy above 95%). As a comparison, LSTRM obtains a mean accuracy of 99.55% while achieving all tasks and has an accuracy of over 97.5% or above for each and every task (Table 4.2). As shown in Table 4.2, RRN has significantly underperformed in basic induction tasks, due to its lack of required attention, on the induction tasks.

R-MN's accuracy is 99.02% for the bAbI 1K tasks [39]. R-MN uses the input in the datasets without any filter. Therefore, the input contains more noises as compared to LSTRM.

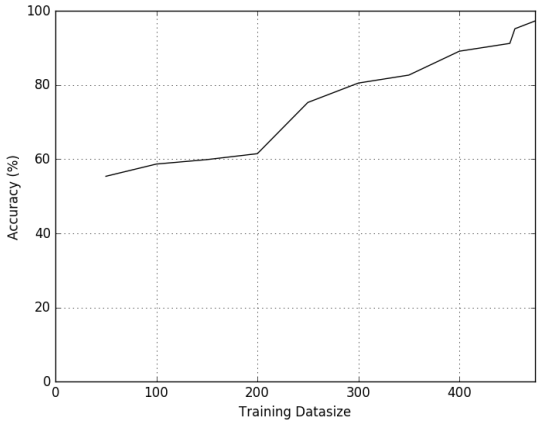


Figure 4.8: LSTRM outperforms RRN and DMN at training data size 450

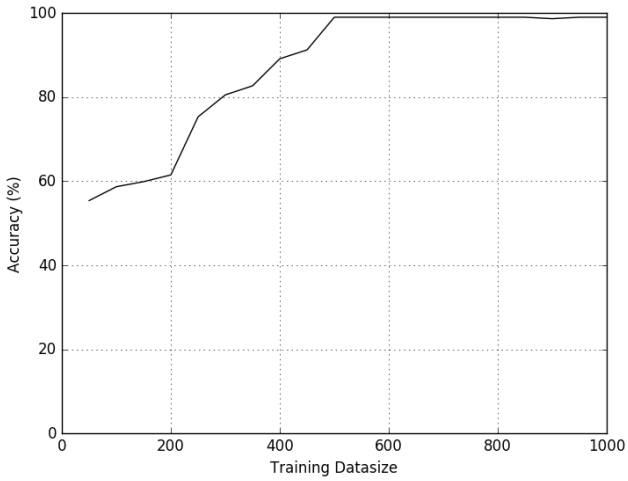


Figure 4.7: Overall accuracy of the bAbI 1K dataset introducing LSTRM, respective to the training data size

Basic induction was not achieved by RRN (Table 4.2). RRN is based on reasoning and has shown the capability of achieving deep reasoning tasks. RRN fails in basic induction tasks due to the lack of memory or attention. DMN [13] generates an overall mean accuracy of 93.6% and achieves 18 tasks (Table 4.2). DMN has failed in position reasoning and pathfinding (Table 4.2), which require deep reasoning capabilities. DMN focuses on attention but not on reasoning. This clearly shows that LSTRM achieves better results, as compared to DMN, and RRN.

LSTRM shows that using the hybrid approach of the RL and Memory networks has the capability of achieving RAM for context-based QA tasks. RRN uses each sentence as a separate node in its relational graph. LSTRM does not require the whole 1K data to achieve its overall accuracy of 99.52% (Figure 4.7). Therefore, LSTRM requires less training data to achieve 99.52% accuracy, which is an indication of LSTRM's capability of faster adaptability.

Furthermore, as Figure 4.8 shows, LSTRM is capable of achieving (97.3% accuracy) results similar to that of RRN, with only 475 training datasets (97.3% accuracy, which is slightly higher than RRN's 97.02%). Table 4.2 also shows the accuracy of each task, when given only 475 datasets, to achieve similar results to RRN.

4.5.3 Results for TriviaQA Dataset

The authors have set BiDAF [66] which achieved the Exact Match (EM) at 40.32% for the Wiki dataset and 40.74% for the web dataset. The dataset also revealed the human performance for both datasets of 79.92% for the Wiki and 75.4% for the web datasets, which is very high compared to the benchmarks that the authors have set [76] (Table 4.3).

Memen [20] was tested against the TriviaQA dataset, slightly improving on the EM and F1 scores compared to the benchmark set by the authors. Memen uses a combination of four

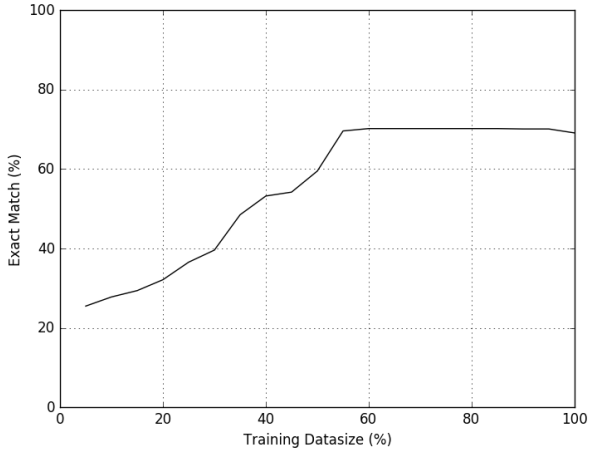


Figure 4.9: LSTRM's EM values with increasing Trivia_QA training dataset sizes

BiLSTMs to create its attentional vectors; these vectors require a high level of computational power. Furthermore, Memen requires name entity tagging and point of speech tagging. Therefore, Memen requires more complex pre-requests to achieve its results. It also focuses more on attention than reasoning and memory in RAM.

The mnemonic reader [82] has shown an improvement over Memen by achieving an EM of 46.65% for the web and 46.95% for the Wiki datasets (Table 4.3). The bidirectional approach from question to context and vice versa has supported the results. Therefore, by revisiting the context with the knowledge of the question, the mnemonic reader shows either a smaller focus on the memory of the network or a lesser memory capacity.

Table 4.3 shows the LSTRM's performance compared with the other methods focused on TriviaQA. LSTRM shows its capability of achieving all factors in RAM. Furthermore, Figure 4.9 shows the training performance of LSTRM and that it achieves an EM score of 79.1% by using 60% of its training data. This is also a clear indication of LSTRM's faster adaptability. Figure 4.9 also shows that after achieving the highest EM, with an increase of the training dataset size, its performance does not drop but stays at the same EM value instead. This shows that LSTRM does not overtrain and reduce its testing EM score.

TABLE 4.3: Comparison of TriviaQA's EM and F1 scores between LSTRM and BIDAF, MEMEN Mnemonic Reader and humans. '-' indicates that tests were not carried out.

Method	Domain	Distance Supervision			
		Development Dataset		Test Dataset	
		EM (%)	F1 (%)	EM (%)	F1 (%)
BiDAF	Wiki	40.26	45.74	40.32	45.91
	web	41.08	47.40	40.74	47.05
MEMEN	Wiki	_	_	43.16	46.90
	web	_	_	44.25	48.34
Mnemonic Reader	Wiki	_	_	46.94	52.85
	web	_	_	46.65	52.88
LSTRM	Wiki	78.2	81.51	79.1	82.8
	web	73.4	78.1	74.92	77.39
human	Wiki	_	_	79.7	_
	web	_	_	75.4	_

4.5.4 Results for Quasars Dataset

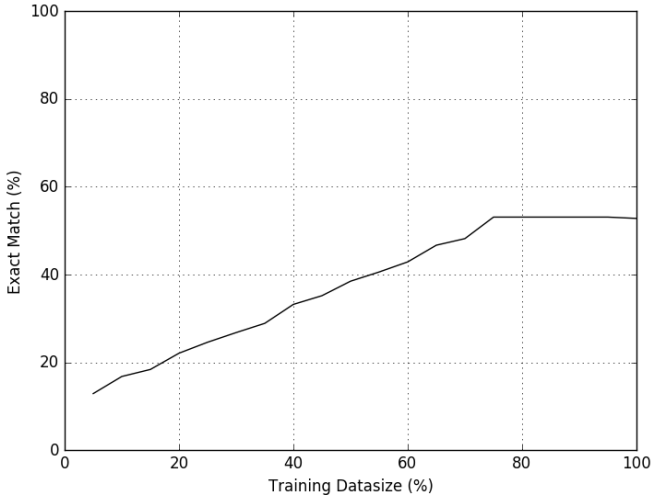


Figure 4.10: EM results for the QUASARS dataset with reference to the training dataset size

The most recent dataset, Quasar, provided another set of tests. As shown in Table 4.4, BiDAF was tested on this dataset with much lower performance compared to humans [77]. The human performance figures were obtained by testing both experts (closed book) and non-experts (open book) on the Quasars dataset. These tasks are challenging for both experts and non-experts alike. Quasar-T human results are less than EM 55%. Therefore, even for humans, these tasks are challenging. Because this dataset is relatively new, only a few methods have been tested on it. R^3 [67], the open domain QA network, is tested but only on the Quasar-T dataset. It achieved EM 34.2%. It used the RL approach to support its ranking. The rankings show support for the results by identifying the best-suited sentences and generate the results using the rankings.

An evidence-based answer re-ranking module focuses on the sections of the context which require attention to identify the answer [91]. The re-ranker model focuses on attention in order to answer the questions. Using evidence available in the context that is related to the question, the answer is re-ranked similar to that of R^3. This relies more on the evidence in the context to answer the questions. The evidence is used to draw attention towards the context segments until the model pinpoints the final answer through attention. The Full Re-Ranker model focuses on attention, as compared to reasoning and memory, which is undermined in some QA tasks.

TABLE 4.4: Quasars EM and F1 scores for overall EM and F1 scores compared to BiDAF, expert human and non-expert human and LSTRM.

Method	**Overall Accuracy**			
	EM		**F1**	
	Validation accuracy (%)	**Test Accuracy (%)**	**Validation accuracy (%)**	**Test Accuracy (%)**
BiDAF	0.257	0.259	0.289	0.285
LSTRM	**0.523**	**0.531**	**0.591**	**0.599**
Human Expert	0.547	_	0.604	_
Human Non-Expert	0.515	_	0.606	_

LSTRM was tested with the Quasars dataset as shown in Table 4.4 and Table 4.5. LSTRM outperformed BiDAF, R3 and Full Re-ranker and has achieved near-human performance. Rather than focusing on re-ranking sentences, LSTRM extracts the direct answer through its memory, reasoning and attention sequence.

This shows that tackling RAM to achieving an F1 score of 59.9% for many datasets, without changing the model, is an achievable task. Figure 4.10 shows the faster adaptability of achieving an EM score of 53.1%, using only 75% of the training dataset. Therefore, LSTRM has a faster learning capability than that of currently available deep learning models (Figures 4.7 - 4.10) for all the tested datasets.

TABLE 4.5: Quasars-T and Quasars-S EM and f1 scores against Reinforced ranker-reader, full ranker, human performance and LSTRM. '_' indicates the models have not been tested on the dataset category.

Method	**Quasar-T**		**Quasar-S**	
	EM (%)	**F1 (%)**	**EM (%)**	**F1 (%)**
Reinforced Ranker-Reader (R3)	34.2	40.9	_	_
Full Re-Ranker	42.3	49.6	_	_
LSTRM	**51.3**	**55.1**	**58.1**	**59.9**
Human Performance	51.5	60.6	_	_

4.6 Summary

LSTRM is proposed to achieve RAM which is required for context-based QA. This has shown that LSTRM is capable of handling abstract RAM tasks for context-based QA. LSTRM uses the memory gate's construct for memory, RL for reasoning and attention for QA tasks. Reasoning, attention and memory are required for QA. LSTRM shows that it has the capacity of reasoning and paying attention to the network's memory for QA. LSTRM was tested on 20 tasks from the bAbI 1K, TriviaQA and Quasar datasets. All the bAbI 1K tasks achieved an overall mean accuracy of 99.54%. LSTRM achieved an EM score of 79.1% for the Wiki dataset and an EM score of 74.92% for the web dataset of TriviaQA. LSTRM achieved an EM score of 53.1% for the Quasar dataset. LSTRM also shows that it is highly adaptable through the bAbI dataset, by using less training data, as compared to the other models, in order to achieve all 20 tasks. Therefore, an adaptation of memory networks and RL have shown, through LSTRM, that it is capable of achieving RAM for context-based QA.

Chapter 5

Language Modeling through Long-Term Memory Network

In Chapter 4, a model which addresses reasoning, attention and memory for natural language understanding is introduced. Chapter 4 demonstrated the importance of reasoning, attention and memory for natural language understanding. However, reasoning and attention rely on a long memory. Languages have longer sequences. As such, a better memory network should hold longer sequences.

This Chapter focuses on the area of language modelling (predicting the next word in a sequence of words) for natural language understanding. Language modelling requires a longer memory. Therefore, this chapter introduces the Long-Term Memory (LTM) network to address language modelling. The key aspects of predicting the next word is that of understanding the given sequence. The LTM holds the whole given sequence in order to address language modelling. Furthermore, in order to fairly compare the existing models that address long term memory, general procedures are followed in experimentations and the models, which are introduced in Chapter 3, are not used.

This chapter is organized as follows. Section 5.1 discusses long term memory. Section 5.2 discusses the problems in long term memory. Section 5.3 proposes a long term memory network. Section 5.4 introduces the experiments conducted on the long term memory network. Sections 5.5 and 5.6 present the long term memory network results and discussion. Finally, some concluding remarks are given in Section 5.7.

5.1 Long-Term Memory

Natural language understanding requires the processing of sequential data. Natural language is time-dependent, and past information can influence current and future output. Therefore, models which are capable of processing sequential data are required. Memory determines a model's

capability of recalling from past information. Sequential deep learning models have shown to achieve state-of-the-art results in natural language understanding tasks such as question answering [39], machine translation [27] [92] and language modelling [21] [93] [94].

Memory networks have a recurrent behaviour which uses outputs to influence the current output [93] [15] [37]. With an increase in sequence length, the effect on the current input is reduced, and after a certain number of steps, the effect on the current input becomes invisible. In order to understand a language, the model used is required to learn from past knowledge. Relevant information needed to understand language is spread throughout the sequence. Therefore, long term memory is required for natural language understanding [21] [93].

Recurrent Neural Networks (RNNs) are capable of handling long sequences but suffer from the exploding and vanishing gradient descent [21] [95]. In order to overcome the issue, the Long Short-Term Memory Networks (LSTMs) [15], Simple Recurrent Network [21] and Memory Network [37] clip the gradient. These models still suffer from the problem of the vanishing gradient when the sequences are long. The gradients of non-linear functions are close to zero and the gradient is backpropagated through time while multiplied. When the eigenvalues are small, the gradient will converge to zero rapidly. Therefore, these models are capable of only handling short term dependencies.

LSTM, GRU and SRN, as proposed by Mikolov [21], use gates to control the vanishing gradient problem. The gates control the memory sequence and prevent the overflow of data. The forget gate in the LSTM is a crucial element which forgets the past sequence [69]. The gates control or forget the previous sequences which influence the current input. Therefore, these memory networks do not handle long-term sequences.

Holding longer sequences in memory is important in the proper understanding of a language and it is also necessary for many long-term dependency tasks [96]. In order to remember long sequences, as well as to prevent the learning model from suffering from the vanishing gradient problem, the Long Term Memory Network (LTM) is introduced in this Chapter. The LTM does not forget past sequences. The LTM incorporates past outputs and current inputs. It generalizes past sequences and gives higher emphasis on the new inputs in order to support natural language understanding. The LTM was tested for long term memory dependency-based language modelling tasks. The LTM is tested on the Penn Treebank and Text8 datasets and it outperformed the current state-of-the-art memory network models.

5.2 Problems in Long-Term Memory

Long term memory dependencies require the learning from patterns. Memory networks are used in order to learn long-term dependencies [39]. Memory networks including RNN and LSTM are used for many natural language tasks such as question answering, speech to text, language modelling and time series analysis [39] [97] [98] [12] [99]. These memory networks have shown to achieve state-of-the-art results in benchmark datasets. However, RNN, LSTM and other memory networks perform differently from each other, and each has its own merits.

RNN is capable of handling infinite continuous sequences [100]. It takes input and passes the value continuously. The output is then looped back and combined with the input [38]. However, long-term dependencies learning fails due to the exploding and vanishing gradient problem [101]. This is due to the direct influence of past information to the current input x_t (5.1). The internal state S_t for a current input of RNN can be defined as:

$$S = activate\ (U_{x_t} + W_{S_{t-1}}) \tag{5.1}$$

where activation can be any activation function (e.g. tanh, Relu), U the weight for the current input, W being the weight for the past input state S_{t-1} [100]. Therefore, the overall output would be affected by past outputs. When $W_{S_{t-1}}$ is added to the weight of the current input U_{x_t}, the past state S_{t-1} directly affects the current state S_t , as shown in Equation 5.1.

The LSTM was introduced in order to handle the vanishing and exploding gradient problem [15]. The forget gate was later added to the original LSTM. This is capable of preventing the internal state from growing indefinitely and handling the network break [69]. The forget gate resets the cell state when it decides on forgetting the past sequence. The cell state holds past inputs with the network or resets the cell state to forget past information held in the network. The LSTM has shown to be a stable model, which is not affected by the vanishing and exploding gradient problem [101]. However, the LSTM is only capable of handling short term dependencies [100].

Traditional memory networks (RNN and LSTM) have shown to handle natural language understanding tasks [102]. RNN is capable of handling continuous data streams which are entered into the network, as in speech recognition [103] and language modelling [21] [94] [30]. The LSTM has shown to perform more complex tasks such as question answering [39] [12]. Traditional memory networks and specified memory networks (Dynamic Memory Network [13] and Reinforced Memory Network [39]) benefit from the learning from longer dependencies in

order to understand language. Longer dependencies are captured by adding more hidden layers. The hidden layers would also contribute towards the vanishing and exploding gradient. Therefore, forgetting past sequences is one main approach used in memory networks [93] [69]. This affects long-term dependency.

The vanishing and exploding gradient are one of the most problematic issues in memory networks through backpropagation [5]. The memory network is trained by deriving the gradients of the network weights using backpropagation and the chain rule. Consider a long sequence which has more than 30 words as the input, "I was born in France. I moved to the UK when I was 5 years old ... I speak fluent French". Using language models, the last word of the paragraph "French" requires learning through a long dependency from the first word "France". Passing the paragraph through an RNN can cause the vanishing and exploding gradient problem [38]. This problem occurs while the RNN is training. Gradients from the deeper layers have to go through matrix multiplications using the chain rule, and if the previous layers have small values, it declines exponentially [38]. These gradient values are insignificant to the model to learn from; this is a vanishing gradient problem. If the gradient is large, it gets larger and explodes, which negatively affects the model's training; this is the exploding gradient problem.

Clipping the gradients places a predefined threshold value which changes the gradient length and makes an attempt to control the vanishing and exploding gradient problem of RNN [38]. Gradient clipping affects the convergence of the gradient. The LSTM and other memory networks avoid the vanishing and exploding gradient by using gates which control passing past outputs to the current input [104]. Clipping also requires a target to be defined at every time step, which increases the complexity [38]. Memory networks, including the LSTM, forget past outputs which the network deems as irrelevant. Attention-based memory networks [105] avoid the vanishing and exploding gradient by focusing on only a few factors which are relevant to the tasks. As these methods are used to avoiding the vanishing and exploding gradient, it prevents the prolonging of the memory of the network. According to the example, if either the sequence is long, or the model does not identify any relevancy in "France", it is removed from the memory. If the model does not know "France", it would directly influence the model's prediction of the last word "French".

The long-term memory network should have the capability of holding all past sequences and not be affected by the vanishing or exploding gradients.

5.3 Long-Term Memory Network

The proposed model has two main objectives: 1) to handle longer sequences, and 2) to overcome the vanishing gradient. The proposed LTM is structured such that it is capable of holding and generalizing old sequences (Figure 5.1) and gives emphasis on recent information. Figure 5.1 shows a single cell LTM (which holds long term memory) which generalizes past sequences.

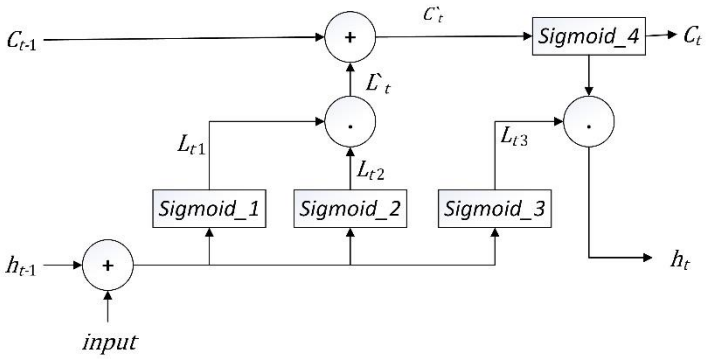


Figure 5.1: Long-Term Memory cell. The arrows show the data flow from within the cell. "." indicates the dot product between the two vectors and + indicates the sum of the two vectors.

Retaining longer memory sequences is a crucial requirement in natural language understanding since past sequences affect current inputs [36]. Furthermore, the LTM gives an emphasis weight to the current input. The LTM holds three states:

a.) input state: handles the current input and passes it on to the output.

b.) cell state: carries past information through each time step.

c.) output state: handles the current output and passes the output to the cell state.

The LTM's functionality relies on the gate structure within it. The LTM cell contains four gates, with the first three gates impacting on the input, while the last gate controls and generalizes the cell state. However, the LTM's cell state does not reset itself in the same way as the function of the LSTM forget gates [69]. Therefore, the LTM is capable of holding longer sequences in memory. The following sections provide the details of the architecture.

5.3.1 Input state

The input is combined with the previous output and passed on to Sigmoid 1, as shown in Equation 5.2. In Equation 5.2, σ indicates the sigmoid functions and $W_1$is the weight for the

gate. L_{t1} is the by-product, which generates an effect on the LTM cell which depends on the current input and previous output.

$$L_{t1} = \sigma(W_1(h_{t-1} + input)) \tag{5.2}$$

Similarly, Equation 5.3 shows a similar functionality but with a different weight W_2, which gives a higher impact on the current input, although scaled through the sigmoid functions (Sigmoids 1 and 2). These two equations, (5.2) and (5.3), support long term memory by emphasizing on the current input and adds on to the past input. W_2 is the weight for the gate represented by Equation 5.3.

$$L_{t2} = \sigma(W_2(h_{t-1} + input)) \tag{5.3}$$

In order to emphasize the current input's effect on the output, L_{t1} and L_{t2} are passed through a dot operation to create L'_t. L'_t is created, as shown in Equation 5.4.

$$L'_t = L_{t1}.L_{t2} \tag{5.4}$$

L'_t amplifies the effect of the current input and past output. L'_t is passed on to the cell state, which would be carried along to future sequences. L'_t shows the current input's effect on the output.

5.3.2 Cell state

The cell state in the LTM is similar to the LSTM's cell state [15], which carries forward past outputs to the present cell. Natural language understanding requires both past outputs and current inputs. The current input is emphasized over the past outputs. Thus, L'_t has a higher value, from the combination with the current input, and is passed on to the cell state, as shown in Equation 5.5. Therefore, the output would have a higher effect on the current input. As shown in Equation 5.5, C'_t, the current cell state, combines the current input $L'_t$and the past output C_{t-1}.

$$C'_t = L'_t + C_{t-1} \tag{5.5}$$

The final cell state C_t, as shown in Equation 5.6, is calculated using the C'_t and passes through Sigmoid 4. Through this, the LTM scales the cell state C_t. The cell state carries on a scaled value to the final output state. W_4 is the weight for Equation 5.6.

$$C_t = \sigma(W_4 C'_t) \tag{5.6}$$

5.3.3 Output State

Equation 5.7 shows the direct influence on the output of a given LTM cell. L_{t3} directly influences the output by passing on the current input. W_3 is the weight for Equation 5.7.

$$L_{t3} = \sigma(W_3(h_{t-1} + input)) \tag{5.7}$$

The cell state C_t and L_{t3} are joined together and combined through the dot operation. C_t and L_{t3} create the final output h_t. Equation 5.8 shows the final output creation. h_t has a higher impact through the current input as well as the past outputs. Therefore, the impact from both the past and current inputs are combined, as shown in Equation 5.8.

$$h_t = C_t . \mathrm{L}_{t3} \tag{5.8}$$

The output h_t and C_t is passed on to the next time step, which is shown in Figure 5.2. The LTM is used as a cell, and the cell passes h_t and C_t. This also shows how the cells pass the past outputs on and combine with the current inputs.

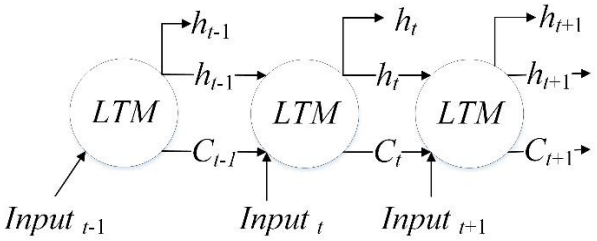


Figure 5.2: Long-Term Memory cells connected. The figure also illustrates the data passed on in the cell state and how the output is passed on from one LTM cell to another.

5.4 Experimentation on Long-Term Memory Network

In order to demonstrate long term dependency learning, the LTM is tested on language modelling. Three types of experiments are conducted to evaluate the LTM using the Penn treebank and Text8 datasets. The Penn treebank dataset contains 2499 stories of the Wall Street Journal. These stories are in the raw text format. The Text8 dataset contains over 240000 Wikipedia articles. Articles from both datasets contain long relationship dependencies between words. The LTM is evaluated on the two datasets against current state-of-the-art models. Finally,

the LTM is evaluated against itself by changing the number of cells, in order to find out which cell size generates the best results.

The LTM was first evaluated on the Pennbank dataset [106]. Similar to the model [107], it consists of pre-processing the data and the training size of 930K tokens, as well as validating the size of 74K tokens and the testing size of 82K tokens. The dataset has a vocabulary of 10K words. In order to match with the current state-of-the-art model experiments, 300 LTM cells are used.

The second dataset Text8 [21] has 44K vocabulary from Wikipedia. This dataset has 15.3m training tokens, 848K validation tokens and 855K test tokens. The settings are similar to [108]. Words which occur ten times or lower are placed as an unknown token. 500 LTM cells are used in this experiment.

In order to evaluate the model on its performance, the cell number is gradually increased and tested for both the Pennbank and Text8 datasets. The experiment conditions are the same as the above experiments, except for the number of layers. All learning models on the Penn Treebank dataset follow [107] closely, while the experiments on Text8 follows [21]; this includes the inputs with hyper-parameters.

5.5 Results for Language Modelling on Long-Term Memory Network

The LTM's long term memory is tested on the Penn Treebank and Text8 datasets. The results are validated using perplexity shown in Equation 5.9. Perplexity is the inverse probability of the test set, normalized by the number of words. The lower the perplexity, the better the model.

$$Perplexity(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \dots w_{i-1})}} \tag{5.9}$$

The first experiment was based on the Penn treebank dataset. The results are shown in Table 5.1. The LTM was tested against traditional memory and recurrent networks and current state-of-the-art models (Delta-RNN). RNN, which had the lowest performance over the tested models, with 300 hidden layers, achieved a test perplexity of 129. This demonstrates that RNN is not capable of handling long-term dependencies. Although the LSTM has outperformed the RNN, ultra-specific models (which handle long term memory) outperformed the generalized models on long term memory. The LTM achieved a test perplexity of 83 with 300 units, which is 20 points above the current state-of-the-art results. LTM is also tested on the Text8 dataset and

compare with RNN, LSTM, SCRN, and MemN2N (Table 5.2). Furthermore, the LTM achieved state-of-the-art results at ten hidden layers (Table 5.3).

The LTM was also tested with the Text8 dataset with 500 hidden layers. The LTM was compared against traditional memory networks and current state-of-the-art models (MemNet) (Table 5.2). The LTM outperformed all state-of-the-art models by using only ten hidden units (Table 5.3). The ultra-specified long term dependency-based memory networks have shown to outperform the generic memory networks.

The LTM was tested on the Text8 and Penn treebank datasets by increasing its hidden layers, in order to identify the best performing number of hidden layers. Table 5.3 shows the validation and testing of perplexity for Text8 and Penn Treebank while increasing the hidden layers. Table 5.3 also shows that the LTM achieved state-of-the-art results with only ten hidden layers, in which other networks require 300 hidden layers or more to achieve the same state-of-the-art results. The results are further improved by increasing the number of hidden layers. The LTM achieved the best results for the Penn treebank dataset with 650 hidden layers. Furthermore, the LTM achieved its best results for Text8 with 600 hidden layers.

Table 5.1: PENN Treebank Validate and Test Perplexity

Model	**Penn Treebank**		
	# hidden layers	**Validate perplexity**	**Test perplexity**
RNN	300	133	129
LSTM	300	123	119
SCRN	300	120	115
Delta-RNN	300	-	102.7
LTM	300	85	83

Table 5.2: TEXT8 Validate and Test Perplexity

Model	**Text 8**		
	# hidden layers	**Validate perplexity**	**Test perplexity**
RNN	500	-	184
LSTM	500	122	154
SCRN	500	161	161
MemN2N	500	118	147
LTM	500	85	82

5.6 Discussion on Long-Term Memory Network

The structure of the LTM, as shown in Figure 5.1, is designed in order to hold the inputs that passed through the LTM cell, as well as to scale the output. The use of the sigmoid functions is a crucial aspect of maintaining a scaled output. Equation 6 is used to create a cell state and output. The use of the sigmoid function in Equation 6 scales the cell state in order to prevent the exploding or vanishing gradient problem. Since the cell state is scaled and passed on from one-time stamp to the other, the cell state value would not explode or vanish, thereby preventing the occurrence of the vanishing or exploding gradient. The vanishing and exploding gradient are the main reasons for a memory network to forget or underperform. In order to prevent the exploding and vanishing of the gradient, the LSTM introduces the forget gate [15]. Using the forget gate, the LSTM can handle longer sequences and forget the sequence when irrelevant sequences are presented to the LSTM. However, past sequences, although not substantially relevant, have an effect on long term natural language understanding tasks. The LSTM has a downfall in long term memory. On the other hand, the LTM scales the outputs and holds it in its memory. Therefore, even the long dependencies would affect the final output of the LTM.

Table 5.3: Increasing the number of hidden layers for the Validation and Test Perplexities for Text8 and Penn Treebank Datasets

# hidden layers	**Text8**		**Penn Treebank**	
	Validate perplexity	**Test perplexity**	**Validate perplexity**	**Test perplexity**
10	103	100	100	99
50	101	99	98	97
100	99	98	95	92
150	97	95	90	89
200	95	93	88	86
250	93	90	87	85
300	90	89	85	83
350	89	87	82	80
400	87	86	79	78
450	86	84	77	76
500	85	82	74	72
550	81	80	72	70
600	79	77	69	67
650	79	77	68	67
700	79	77	68	67

The LTM leaves a high impact on the new inputs (5.4). The LTM combines L_{t1} and L_{t2} in order to pass a higher impact from the current input to the output (as shown in (5.4)). Therefore,

the LTM gives a higher priority to the new inputs, which is more relevant to the current output. Equation 8 shows the effect on the final output, which combines both the processed input and cell state, which carries past sequential information.

Language modelling is one evaluation method to analyse the long-term dependencies of the LTM. The Penn treebank and Text8 datasets require longer learning capabilities. Language modelling requires a clear understanding of the entire text, rather than a window of the text. Holding an entire article, in order to predict and understand the text, is easier for the model. Through scaling, the LTM holds all the information that passes through it. Therefore, the LTM is capable of understanding a clear picture of the entire article. Attention-based memory networks [9] identify the most relevant information and predicts based on the information the attention has captured. Attention-based memory networks are capable of handling shorter sequences. It fails to hold long sequences as the attention diverts when given longer sequences. The LTM does not focus on memory and holds all past inputs. Unlike attention-based networks that would forget most irrelevant information which might be relevant later on the sequence, the LTM would hold all the information passed through the model.

Tables 5.1 and 5.2 compare the LTM with other state-of-the-art models and traditional memory networks. This shows that the LTM is capable of handling longer sequences and produces state-of-the-art results. LTM's longer memory plays a crucial role in language modelling tasks. Table 5.3 shows that increasing LTM cells would further enhance the results and produce a lower perplexity score. The LTM has shown to hold longer sequences and be unaffected by the vanishing and exploding gradient.

Similar to the LSTM, the LTM avoids the vanishing or exploding gradient descent by using the gates. The LTM uses the gates to enhance the input passed to the network. The LTM handles long-term dependencies by the use of sigmoid functions to scale the new inputs and carry on past outputs at the gates. The LTM handles long sequences through the scaling. The example of "I was born in France. I moved to the UK when I was 5 years old ... I speak fluent French" predicting the last word, "French", is thus attainable since the model holds the entire sequence. Holding the entire sequence in the memory has supported the model to predict "French" as the last word. The LTM carries forward the entire sequence, allowing the models to use the entirety of the sequence to predict the final word. The most important requirement for the model is to be able to predict the last word. The LTM is capable of handling the vanishing and exploding gradient, as well as handling long-term dependencies.

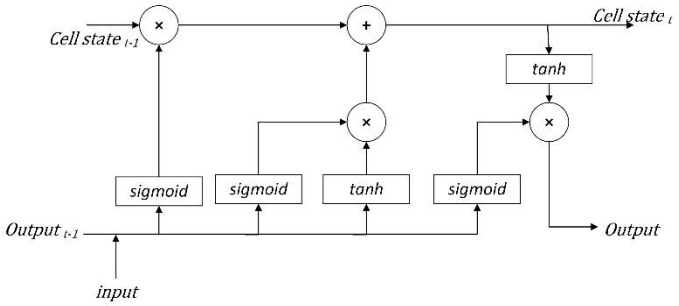


Figure 5.3: General cell of a Long Short-Term Memory network. The figure illustrates a general Long Short-Term Memory Network cell taken from the time (time stamp $input_t$).

Figure 5.3 shows the LSTM cell, which holds three gates (forget gate, input gate and output gate). The LSTM holds a combination of sigmoid and tanh activation functions, while the LTM relies only on the sigmoid function. Comparing Figure 5.1 with Figure 5.3, there is a core difference between the LSTM and LTM. The LTM uses generalization through the sigmoid activation functions that hold a longer sequence, without forgetting past information. This generalization supports the LTM with its stability, without deteriorating the functionality of the LTM with time or new data. However, the LSTM forgets longer sequences through the forget gates in order to maintain the network's stability. Hence, the LSTM sacrifices long-term dependencies for network stability.

5.7 Summary

This Chapter presents a long term memory network which is capable of handling long-term dependencies. The LTM is capable of handling long sequences without being affected by the vanishing or exploding gradient. The LTM has shown to outperform traditional LSTM and RNN, as well as the memory specific networks in language modelling. The LTM was tested on both Penn treebank and Text8 datasets, in which the LTM outperformed all state-of-the-art memory networks using minimal hidden units. Increasing the number of hidden units have shown that the LTM does not get affected by the vanishing and exploding gradient. Instead, by adding more hidden units to the LTM, it achieved lower perplexity scores and stabilised.

Chapter 6

Conclusion

In this section, the thesis is summarised with the findings for each contribution.

1. Improving sentence and word representation by enhancing meaning representation

This was investigated through the use of enhanced knowledge in the area of NLU. The use of both context-based and knowledge-based knowledge is used to support NLU. It was demonstrated that the two methods on sentence parsing and word embedding, with the use of enhanced meaning representation, achieved better results as compared to the other models.

In sentence parsing, two fully automatic approaches are proposed and presented for semantic parsing using rich information extracted from ConceptNet 5. Approach 1 uses the results generated through Level 2 for semantic parsing of the queries. Approach 2 filters the concepts generated from Level 1 and passes them onto Level 2. Both approaches produce information-rich results since a multi-level concept extraction is done from ConceptNet 5. The experimental results for Approach 1 showed that when more concepts are available, the precision, recall and accuracy will subsequently improve. For the Stanford Sentiment dataset, the results generated from Approach 1 and Approach 2 have recall values of 93.29%, 94.39% and precision values of 82.37%, 85.37% respectively. Furthermore, when tested on Free917 and WebQ, it achieved 77.1%, 79.2% (recall values) and 36.5%, 38.2% (precision values) respectively for each approach. The results of Approach 2 indicated that with the use of a filter, the results improved. The experiments showed that with the use of more concepts, the semantic parsing could be further improved. These show the capability of enhancing semantic parsing through the complex concepts extracted from ConceptNet.

A word embedding that uses context-based statistics, analogy and related meaning of words to create word representations is proposed. This word embedding holds both context-based information via Word2Vec and related words via Conceptnet. The word representation places a higher weight to the context-based information in order to create the word representation, thus

preventing over-generalization. The word representation is evaluated using SimLex-999, which achieved a Spearman's correlation of 0.886. Furthermore, the proposed word representations are displayed in 3D to show the capability of distinguishing the associated words to similar words based on the context. The proposed word embedding is similar to the human performance of word similarity and association, as it achieved a Spearman's correlation of 0.886; given that a human can achieve 0.78.

2. Addressing RAM in one Model

LSTRM is proposed to achieve RAM which is required for context-based QA. This has shown that LSTRM is capable of handling abstract RAM tasks for context-based QA. LSTRM uses the memory's gate construct for memory, RL for reasoning and attention for QA tasks. Reasoning, attention and memory are required for QA. LSTRM shows that it has the capacity of reasoning and attention on the network's memory for QA. LSTRM was tested on bAbI 1K dataset's 20 tasks, TriviaQA and Quasar datasets. All the bAbI 1K tasks were achieved with an overall mean accuracy of 99.54%. LSTRM achieved an EM score of 79.1% for the Wiki dataset, an EM score of 74.92% for the web dataset of TriviaQA and an EM score of 53.1% for the Quasar dataset. LSTRM also shows that it is highly adaptable through the bAbI dataset, as it used less training data, as compared to the other models, to achieve all 20 tasks. Therefore, an adaptation of memory networks and RL have shown, through LSTRM, that it is capable of achieving RAM for context-based QA.

3. Better handling of longer sequences

This thesis presents a long term memory network which is capable of handling long-term dependencies. The LTM network is capable of handling long sequences without being affected by the vanishing or exploding gradient. The LTM network has shown to outperform traditional LSTM and RNN, as well as the memory specific networks in language modelling. The LTM network was tested on both the Penn treebank and Text8 datasets, in which the LTM has outperformed all state-of-the-art memory networks using minimal hidden units. Increasing the number of hidden units have shown that the LTM does not get affected by the vanishing and exploding gradient. By adding more hidden units, the LTM has achieved lower perplexity scores and stabilised.

Through the introduction of these models, this thesis points out the importance of using enhanced meaning representation for language understanding through the use of deep learning models, which are capable of reasoning, attention and learning long sequential memory. Therefore, in this thesis, it has been shown that by enhancing meaning representations and through reasoning, attention and long sequential memory, language understanding can be addressed.

6.1 Future work

In this section, a few areas that future research on NLU could focus on is highlighted. With advancements in computational power, largely available public data and machine learning algorithms specific to achieving given tasks [1], the models introduced in this thesis can be further improved upon.

1. The word embedding introduced in Chapter 3 is optimized using Self-Organizing Maps (SOM). When using SOM, the final word vector is optimized. However, there are other techniques available to further optimize the word embedding so as to get a better word representation. Therefore, it is important to investigate other optimization algorithms. Proper use of optimization algorithm can improve the word embedding. The main focus of optimization can be done via a global optimization approach. Through the use of better optimization algorithms, word embedding can be further improved to better illustrate the meanings of the words with clear representations. The optimization can also be used to reduce the vector size. The currently proposed vector holds 300 rows per word. However, through the optimization of word embedding, the vector can be reduced. This can be highly beneficial for memory models which use the proposed embedding.

2. Currently, LSTRM is an offline learning model. LSTRM can be tested after it is trained with the data. Although LSTRM requires less data for training, it does not learn from the mistakes made while the model is being tested. Most NLU applications receive data sequentially. Therefore, in order to be applicable, a model should be able to learn from the current input and further optimize the results in the next step. As such, the model should dynamically adapt to the new patterns and data presented to the model. Through enhancing the RL model in LSTRM, the online mode can be investigated. The RL capability of learning can also be used to improve online learning. One investigative approach would be to focus on updating the model network

and the policy network to learn from new sequences online. Another approach is to directly influence the policy network by feeding the responses given through the environment to its results and subsequently updating the policy online. However, in these online learning approaches, while updates are being made to the network, it is important to ensure that the network stays connected and does not drop off.

References

[1] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science*, vol. 349, pp. 261-266, 2015.

[2] J. Allen, *Natural language understanding*: Pearson, 1995.

[3] E. Cambria and B. White, "Jumping NLP curves: a review of natural language processing research [review article]," *IEEE Computational Intelligence Magazine*, vol. 9, pp. 48-57, 2014.

[4] G. G. Chowdhury, "Natural language processing," *Annual review of information science and technology*, vol. 37, pp. 51-89, 2003.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436-444, 2015.

[6] C. Cummins and J. P. de Ruiter, "Computational approaches to the pragmatics problem," *Language and linguistics compass*, vol. 8, pp. 133-143, 2014.

[7] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "Joint learning of words and meaning representations for open-text semantic parsing," in *Artificial Intelligence and Statistics*, 2012, pp. 127-135.

[8] S. Poria, B. Agarwal, A. F. Gelbukh, A. Hussain, and N. Howard, "Dependency-Based Semantic Parsing for Concept-Level Text Analysis," in *CICLing (1)*, 2014, pp. 113-127.

[9] R. Mooney, "Learning for semantic parsing," *Computational Linguistics and Intelligent Text Processing*, pp. 311-324, 2007.

[10] 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)*, 2014, pp. 1532-1543.

[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111-3119.

[12] J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, *et al.*, "Towards ai-complete question answering: A set of prerequisite toy tasks," *arXiv preprint arXiv:1502.05698*, 2015.

[13] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, *et al.*, "Ask me anything: Dynamic memory networks for natural language processing," in *International Conference on Machine Learning*, 2016, pp. 1378-1387.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097-1105.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735-1780, 1997.

[16] R. B. Palm, U. Paquet, and O. Winther, "Recurrent Relational Networks for Complex Relational Reasoning," *arXiv preprint arXiv:1711.08028*, 2017.

[17] T. Furlanello, J. Zhao, A. M. Saxe, L. Itti, and B. S. Tjan, "Active long term memory networks," *arXiv preprint arXiv:1606.02355*, 2016.

[18] B. Peng, Z. Lu, H. Li, and K.-F. Wong, "Towards neural network-based reasoning," *arXiv preprint arXiv:1508.05508*, 2015.

[19] D. A. Hudson and C. D. Manning, "Compositional attention networks for machine reasoning," *arXiv preprint arXiv:1803.03067*, 2018.

[20] B. Pan, H. Li, Z. Zhao, B. Cao, D. Cai, and X. He, "MEMEN: Multi-layer Embedding with Memory Networks for Machine Comprehension," *arXiv preprint arXiv:1707.09098*, 2017.

[21] T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. A. Ranzato, "Learning longer memory in recurrent neural networks," *arXiv preprint arXiv:1412.7753*, 2014.
[22] F. Ma, R. Chitta, S. Kataria, J. Zhou, P. Ramesh, T. Sun, *et al.*, "Long-Term Memory Networks for Question Answering," *arXiv preprint arXiv:1707.01961*, 2017.
[23] C. D. Manning, C. D. Manning, and H. Schütze, *Foundations of statistical natural language processing*: MIT press, 1999.
[24] S. Attardo, *Humorous texts: A semantic and pragmatic analysis* vol. 6: Walter de Gruyter, 2010.
[25] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, pp. 3-26, 2007.
[26] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic Parsing on Freebase from Question-Answer Pairs."
[27] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.
[28] J. D. Williams and G. Zweig, "End-to-end LSTM-based dialog control optimized with supervised and reinforcement learning," *arXiv preprint arXiv:1606.01269*, 2016.
[29] J.-T. Chien and Y.-C. Ku, "Bayesian recurrent neural network for language modeling," *IEEE transactions on neural networks and learning systems*, vol. 27, pp. 361-374, 2016.
[30] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.
[31] A. Kamath and R. Das, "A Survey on Semantic Parsing," *arXiv preprint arXiv:1812.00978*, 2018.
[32] B. Agarwal, S. Poria, N. Mittal, A. Gelbukh, and A. Hussain, "Concept-level sentiment analysis with dependency-based semantic parsing: a novel approach," *Cognitive Computation*, vol. 7, pp. 487-499, 2015.
[33] H. Liu and P. Singh, "ConceptNet—a practical commonsense reasoning tool-kit," *BT technology journal*, vol. 22, pp. 211-226, 2004.
[34] Y. Goldberg and O. Levy, "word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method," *arXiv preprint arXiv:1402.3722*, 2014.
[35] D. Marcu and D. Wong, "A phrase-based, joint probability model for statistical machine translation," in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, 2002, pp. 133-139.
[36] E. Cambria and B. White, "Jumping NLP curves: A review of natural language processing research," *IEEE Computational intelligence magazine*, vol. 9, pp. 48-57, 2014.
[37] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *arXiv preprint arXiv:1410.3916*, 2014.
[38] 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.
[39] A. Nugaliyadde, K. W. Wong, F. Sohel, and H. Xie, "Reinforced memory network for question answering," in *International Conference on Neural Information Processing*, 2017, pp. 482-490.
[40] S. S. Pradhan, W. H. Ward, K. Hacioglu, J. H. Martin, and D. Jurafsky, "Shallow Semantic Parsing using Support Vector Machines," in *HLT-NAACL*, 2004, pp. 233-240.
[41] J. Berant and P. Liang, "Semantic Parsing via Paraphrasing," in *ACL (1)*, 2014, pp. 1415-1425.

[42] R. Speer and C. Havasi, "ConceptNet 5: A large semantic network for relational knowledge," in *The People's Web Meets NLP*, ed: Springer, 2013, pp. 161-176.
[43] G. A. Miller, "WordNet: a lexical database for English," *Communications of the ACM,* vol. 38, pp. 39-41, 1995.
[44] L. Shi and R. Mihalcea, "Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing," *Computational linguistics and intelligent text processing,* pp. 100-111, 2005.
[45] C. Havasi, R. Speer, and J. Alonso, "ConceptNet 3: a flexible, multilingual semantic network for common sense knowledge," in *Recent advances in natural language processing*, 2007, pp. 27-29.
[46] Y. Su and X. Yan, "Cross-domain Semantic Parsing via Paraphrasing," *arXiv preprint arXiv:1704.05974*, 2017.
[47] B. Dhingra, H. Liu, R. Salakhutdinov, and W. W. Cohen, "A comparative study of word embeddings for reading comprehension," *arXiv preprint arXiv:1703.00993,* 2017.
[48] T.-H. Wen, M. Gasic, N. Mrksic, P.-H. Su, D. Vandyke, and S. Young, "Semantically conditioned lstm-based natural language generation for spoken dialogue systems," *arXiv preprint arXiv:1508.01745,* 2015.
[49] X. Zhang and Y. LeCun, "Text understanding from scratch," *arXiv preprint arXiv:1502.01710,* 2015.
[50] J. P. Chiu and E. Nichols, "Named entity recognition with bidirectional LSTM-CNNs," *arXiv preprint arXiv:1511.08308,* 2015.
[51] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781,* 2013.
[52] C. Potts, D. Lassiter, R. Levy, and M. C. Frank, "Embedded implicatures as pragmatic inferences under compositional lexical uncertainty," *Journal of Semantics,* vol. 33, pp. 755-802, 2016.
[53] A. Nugaliyadde, K. W. Wong, F. Sohel, and H. Xie, "Multi-level Search of a Knowledgebase for Semantic Parsing," in *International Workshop on Multi-disciplinary Trends in Artificial Intelligence*, 2017, pp. 44-53.
[54] F. Hill, R. Reichart, and A. Korhonen, "Simlex-999: Evaluating semantic models with (genuine) similarity estimation," *Computational Linguistics,* vol. 41, pp. 665-695, 2015.
[55] T. K. Landauer, D. S. McNamara, S. Dennis, and W. Kintsch, *Handbook of latent semantic analysis*: Psychology Press, 2013.
[56] J. A. Bullinaria and J. P. Levy, "Extracting semantic representations from word co-occurrence statistics: A computational study," *Behavior research methods,* vol. 39, pp. 510-526, 2007.
[57] R. Lebret and R. Collobert, "Word emdeddings through hellinger PCA," *arXiv preprint arXiv:1312.5542,* 2013.
[58] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013, pp. 746-751.
[59] R. Speer, J. Chin, and C. Havasi, "ConceptNet 5.5: An Open Multilingual Graph of General Knowledge," in *AAAI*, 2017, pp. 4444-4451.
[60] R. Speer and J. Chin, "An ensemble method to produce high-quality word embeddings," *arXiv preprint arXiv:1604.01692,* 2016.
[61] G. N. Leech, *Principles of pragmatics*: Routledge, 2016.
[62] T. Kohonen, "The self-organizing map," *Neurocomputing,* vol. 21, pp. 1-6, 1998.

[63] G. Recski, E. Iklódi, K. Pajkossy, and A. Kornai, "Measuring semantic similarity of words using concept networks," in *Proceedings of the 1st Workshop on Representation Learning for NLP*, 2016, pp. 193-200.

[64] 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*, 2008, pp. 160-167.

[65] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, *et al.*, "Placing search in context: The concept revisited," in *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 406-414.

[66] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," *arXiv preprint arXiv:1611.01603*, 2016.

[67] S. Wang, M. Yu, X. Guo, Z. Wang, T. Klinger, W. Zhang, *et al.*, "R $^ 3$: Reinforced Reader-Ranker for Open-Domain Question Answering," *arXiv preprint arXiv:1709.00023*, 2017.

[68] R. Palm, U. Paquet, and O. Winther, "Recurrent relational networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 3368-3378.

[69] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," 1999.

[70] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning* vol. 135: MIT press Cambridge, 1998.

[71] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, *et al.*, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016, pp. 1928-1937.

[72] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484-489, 2016.

[73] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237-285, 1996.

[74] K. Narasimhan, T. Kulkarni, and R. Barzilay, "Language understanding for text-based games using deep reinforcement learning," *arXiv preprint arXiv:1506.08941*, 2015.

[75] B. Bakker, "Reinforcement Learning with Long Short-Term Memory," presented at the Neural Information Processing Systems, 2002.

[76] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, "TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension," *arXiv preprint arXiv:1705.03551*, 2017.

[77] B. Dhingra, K. Mazaitis, and W. W. Cohen, "Quasar: Datasets for Question Answering by Search and Reading," *arXiv preprint arXiv:1707.03904*, 2017.

[78] S. Shekarpour, K. M. Endris, A. Jaya Kumar, D. Lukovnikov, K. Singh, H. Thakkar, *et al.*, "Question answering on linked data: Challenges and future directions," in *Proceedings of the 25th International Conference Companion on World Wide Web*, 2016, pp. 693-698.

[79] X. Guo, T. Klinger, C. Rosenbaum, J. P. Bigus, M. Campbell, B. Kawas, *et al.*, "Learning to Query, Reason, and Answer Questions On Ambiguous Texts," presented at the 5th International Conference on Learning Representations, France, 2017.

[80] Z. Yu, J. Yu, C. Xiang, J. Fan, and D. Tao, "Beyond Bilinear: Generalized Multimodal Factorized High-Order Pooling for Visual Question Answering," *IEEE Transactions on Neural Networks and Learning Systems*, 2018.

[81] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, p. 471, 2016.

[82] M. Hu, Y. Peng, and X. Qiu, "Mnemonic Reader: Machine Comprehension with Iterative Aligning and Multi-hop Answer Pointing," 2017.

[83] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 2054-2063.

[84] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and Autonomous Control Using Reinforcement Learning: A Survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

[85] S. R. Branavan, H. Chen, L. S. Zettlemoyer, and R. Barzilay, "Reinforcement learning for mapping instructions to actions," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, 2009, pp. 82-90.

[86] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[87] H. Wang, M. Bansal, K. Gimpel, and D. McAllester, "Machine comprehension with syntax, frames, and semantics," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 2015, pp. 700-706.

[88] D. Chen, J. Bolton, and C. D. Manning, "A thorough examination of the cnn/daily mail reading comprehension task," *arXiv preprint arXiv:1606.02858*, 2016.

[89] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[90] B. Dhingra, H. Liu, Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Gated-attention readers for text comprehension," *arXiv preprint arXiv:1606.01549*, 2016.

[91] S. Wang, M. Yu, J. Jiang, W. Zhang, X. Guo, S. Chang, *et al.*, "Evidence Aggregation for Answer Re-Ranking in Open-Domain Question Answering," *arXiv preprint arXiv:1711.05116*, 2017.

[92] Z. Yang, W. Chen, F. Wang, and B. Xu, "Multi-sense based neural machine translation," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 3491-3497.

[93] A. G. Ororbia II, T. Mikolov, and D. Reitter, "Learning simpler language models with the differential state framework," *Neural computation*, vol. 29, pp. 3327-3352, 2017.

[94] M. D. Singh and M. Lee, "Temporal hierarchies in multilayer gated recurrent neural networks for language models," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 2152-2157.

[95] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, pp. 157-166, 1994.

[96] A. Nugaliyadde, K. W. Wong, F. Sohel, and H. Xie, "Enhancing semantic word representations by embedding deep word relationships," in *Proceedings of the 2019 11th International Conference on Computer and Automation Engineering*, 2019, pp. 82-87.

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

[98] S. Sukhbaatar, J. Weston, and R. Fergus, "End-to-end memory networks," in *Advances in neural information processing systems*, 2015, pp. 2440-2448.

[99] S. Boukoros, A. Nugaliyadde, A. Marnerides, C. Vassilakis, P. Koutsakis, and K. W. Wong, "Modeling server workloads for campus email traffic using recurrent neural

networks," in *International Conference on Neural Information Processing*, 2017, pp. 57-66.

[100] H. Salehinejad, "Learning over long time lags," *arXiv preprint arXiv:1602.04335*, 2016.

[101] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," ed: A field guide to dynamical recurrent neural networks. IEEE Press, 2001.

[102] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *ieee Computational intelligenCe magazine*, vol. 13, pp. 55-75, 2018.

[103] S.-H. Chen, S.-H. Hwang, and Y.-R. Wang, "An RNN-based prosodic information synthesizer for Mandarin text-to-speech," *IEEE transactions on speech and audio processing*, vol. 6, pp. 226-239, 1998.

[104] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014.

[105] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Advances in neural information processing systems*, 2015, pp. 577-585.

[106] A. Taylor, M. Marcus, and B. Santorini, "The Penn treebank: an overview," in *Treebanks*, ed: Springer, 2003, pp. 5-22.

[107] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, 2011, pp. 5528-5531.

[108] Z. Xie, S. I. Wang, J. Li, D. Lévy, A. Nie, D. Jurafsky, *et al.*, "Data noising as smoothing in neural network language models," *arXiv preprint arXiv:1703.02573*, 2017.