Steps towards Question Answering with Deep Neural Memory Networks

Sajawel Ahmed

Master Thesis

Goethe University Frankfurt
Fraunhofer IAIS in cooperation with
PricewaterhouseCoopers AG WPG

Supervisors:
Prof. Dott.-Ing. Roberto V. Zicari
Frankfurt Big Data Laboratory
Dr. Jörg Kindermann
Knowledge Discovery Group

March 2017

Abstract

Since the rise of neural networks in science and industry much progress has been made in the field of Artificial Intelligence. In this thesis, steps towards automatic Question Answering (QA) on text passages are examined by applying deep neural networks with memory components to large-scale datasets for the greater aim of developing the human-like ability of reading some piece of text and answering arbitrary questions on it. We explore the neural models of *LSTM* and *Memory Networks* by using three major text datasets of *SQuAD* with *SNLI* for QA, and *MSRP* for Paraphrase Detection. We empirically show that the LSTM achieve high performance for the binary step towards QA and are in addition expandable to Paraphrase Detection with a performance close to the state-of-the-art. Besides, we demonstrate that in contrast to their success on the bAbI tasks Memory Networks are not suitable for QA on the realistic dataset of SQuAD. Finally, we discover the usability of the measure of cosine similarity on the internal hidden vectors of LSTM. The thesis is closed by presenting a semantic search algorithm for text pairs and its further development to use cases for the landscape of business activities.

Acknowledgements

This work would have not been possible without the help of all the people around me. First of all, I would like to thank Dr. J. Kindermann for supervising me with his kind and knowledgeable hints throughout the work for this thesis. He had always a calm and nice attitude with an immense degree of patience making it a pleasure to work with him. I would like to thank S. Giesselbach and further colleagues from the Fraunhofer Institute IAIS for their support and the friendly atmosphere they created at the institute.

Next, I would like express my deepest gratitude to Prof. R. Zicari from the Chair of Databases and Information Systems (Frankfurt Big Data Lab) who gave me overall useful directions during the course of the thesis. Especially his closing remarks about the final drafts were very meaningful.

Besides, I would like to mention the colleagues from PricewaterhouseCoopers. I offer my most sincere thanks to B. Lix and A. Hufenstuhl from Data & Analytics Services and H. Nagafi and J. Müller from the Experience Center for their business insights which were significant while defining the use cases and future perspectives.

Lastly, I thank all my dear friends and beloved ones, especially I. Meyer, and my siblings M. Ahmed and R. Ahmed, and foremost my father and my mother for supporting and motivating me all the time. Without them I would not be there where I am and without them I would not be that what I am. Thank you my father and my mother—this work is dedicated to you!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> *"The scholars and nations of the past which have ceased to exist, were constantly employed in writing books about various fields of science and wisdom, regarding those that were to come after them, and anticipating for a reward proportionate to their ability, and trusting that their endeavors would meet with acknowledgment, attention, and remembrance—content as they were even with a small degree of praise; small if compared with pains which they had undergone, and the difficulties which they had encountered, in revealing the secrets of science and its obscurities."*
>
> — Algorismi, *Liber Algebrae et Almucabola*

Nowadays, we are making huge progress in the field of Artificial Intelligence. Since the rise of artificial neural networks new astonishing frontiers are being continuously discovered. The development is fast to such an extent, that overall no technical limits are in sight. In the light of these developments, Question Answering (QA) is revisited in this thesis. Our overall goal is to depart from traditional approaches to QA where the solutions typically rely on a structured knowledge-base (KB). Examples for such systems are *IBM Watson* or *WolframAlpha*. In their backbone, all these systems typically use some structured KBs like a SQL Database, RDF-Graph or XML-Files. Their predominant achievement is "mainly" the translation of natural language questions into appropriate machine language queries without any usage of higher "semantics" and intelligence. What if we skip this intermediate step of translation and try to answer the human queries directly with machines? More specifically, can we identify a system which can be trained to read like humans and answer arbitrary questions on text passages? This might sound like a science-fiction movie. But certainly with this approach we will be closer the greater aim of Artificial Intelligence.

In this light, the exploratory thesis will examine the performance of neural networks in regard to the domain of QA on textual data. In particular, it will examine how well the semantics of a given text can be understood and utilized

by a deep learning system containing some form of memory components to answer questions posed in standard natural language by human users. The textual data will be taken from various Natural Language Processing (NLP) benchmarks, like the Stanford Question Answering Dataset [25] (SQuAD) which is a Wikipedia based QA dataset. Additional benchmarks like Stanford Natural Language Inference data [6] which asks to determine the relationship between two given texts, will also be considered for training the neural system.

1.1 Motivation (Recent Developments in Machine Learning)

Artificial neural networks are inspired by nature. Fueled by the technological advance in computational power through the use of highly parallel architectures like GPUs for general computing, alongside the availability of large recorded activity data (Big Data), the field of machine learning with neural networks is gaining increasing popularity in academia and industry for its ground breaking progress.

Recurrent Neural Networks like LSTM [13] (containing a local memory cell) and its further development of Memory Networks [36] (MemNN) (containing a global memory block) are the recent class of network architectures which have established a new state-of-the-art performance in many sequence processing applications, such as speech recognition, language parsing, and machine translation. These systems learn how to read from and write to their memory component to solve the need of long-dependency modeling, besides storing some intuitive knowledge in the learned weight parameters of the network like standard feed forward networks. We will use this recent class of neural memory network architectures for the domain of QA and aim to develop its usage further.

1.2 Research Questions (Exploratory Approach)

In this thesis, we examine the class of *QA on Passages* (Passage-QA) with deep neural networks containing some form of memory components and analyze the first steps towards it. Ultimately, we aim to find new frontiers for semantic relationships between text pairs when processed by neural networks. Due to the exploratory nature of this thesis, we have two prominent directions with major research questions (RQs).

Direction-1 Apply and examine the neural architecture of Match-LSTM to binary Passage-QA, a major step towards Passage-QA where the task is to classify

if a passage P contains an answer for a given question Q, both posed in natural language.

- Aim: Learn to develop the ability of reading comprehension on Wikipedia texts with SQuAD's list of (P, Q) pairs by training and extending the Match-LSTM classifier for Natural Language Inference to the related step towards Passage-QA. More precisely, learn to use the local memory component for solving the binary Passage-QA task.
- **RQ-1**: *How good is the performance of Match-LSTM when applied to binary Passage-QA task on Wikipedia articles?*

Direction-2 Apply and examine the performance of the neural architecture of MemNN to realistic Passage-QA task while using the (P, Q, A) triplets of SQuAD.

- Aim: Learn to develop the ability of reading comprehension on Wikipedia texts. More specifically, by using the (P, Q, A) triplets of SQuAD, learn how to use the global memory component for answering a question Q on a text passage P by locating the answer sequence A within P.
- **RQ-2**: *How good is the performance of MemNN when applied to Passage-QA task on Wikipedia articles?*

Further arising research questions These two RQs are the main starting point of the thesis. During the course of the experiments, additional insights were gained which in turn paved the way for further questions.

- **RQ-3**: *What are the effects of transfer learning to related classification tasks on text pairs, and can the extension of Paraphrase Detection be developed with Match-LSTM?*
- **RQ-4**: *Is it useful to apply the measure of cosine similarity to the internal hidden vector representations of Match-LSTM, and can use cases be designed on it?*

1.3 Involved Scientific Fields

This thesis contains a broad range of addressed topics and is located in various fields of science, like Machine Learning & Data Science, Computer Linguistics, and Neuroscience. Besides, it also has a business-oriented focus in its later part where theoretical findings are applied to possible use cases. Finally, the part of software engineering plays a major technical role for finding the actual answers to the research question.

1.4 Structure of Thesis

We provide here a structure of the thesis. Figure 1.1 gives a conceptual overview for this. The rest of this thesis is organized as follows:

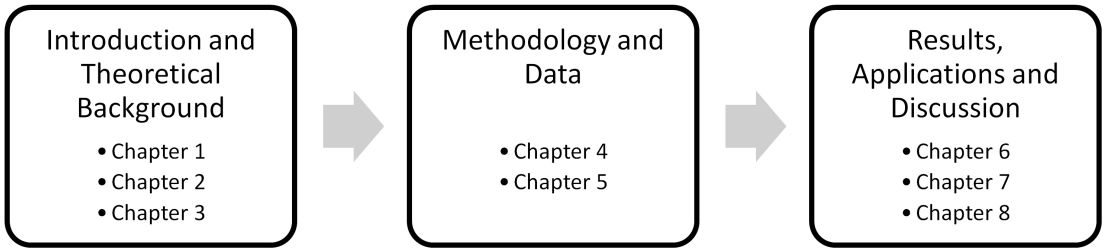


Figure 1.1: Overall structure of the thesis.

Chapter 2 gives the theoretical background for the domain of QA and the method of neural networks. For QA, the classic method is described and contrasted with current trends towards human-like QA on text passages. For neural networks, after explaining the biological background, a theoretical overview of the landscape of various neural architecture is given. Finally, the focus is shifted to the main class of interest for text processing task, leading to the illustration of Recurrent Neural Networks.

Chapter 3 provides a literature review of the state-of-the-art studies, which are closely related to our work and on which we are building on. First, those studies of QA are presented, which use a heterogeneous model of neural networks combined with classic methods using IR/KB. Next, the homogeneous models are presented, which entirely make use of neural networks, thereby becoming end-to-end trainable.

Chapter 4 outlines the methods and tools used to deal with the QA data. The two major pillars of this thesis, namely the models of *Match-LSTM* and *MemN2N* are defined formally, followed by the description of the technical aspects of software engineering and the working environment underlying this thesis.

Chapter 5 describes five publicly available large-scale QA datasets and some necessary preprocessing steps for applying the methods and tools to answer the research questions.

Chapter 6 presents the main experimental results of this thesis in three parts.

The first part starts with the results for Match-LSTM for binary Passage-QA in various setups, continues with the setup of joint-training thereby achieving the best results, and finishes off with a proof-of-concept by performing paraphrase detection on some pre-trained model. In the second part, the results for MemNN are presented when applied to a simplified version of SQuAD for Passage-QA. In the third and final part, a deeper analysis of the internal hidden representations of Match-LSTM is performed, thereby discovering the applicability of the measure of cosine similarity. The chapter closes by engineering a semantic search algorithm and proposing an alternative numerical evaluation method for word vector models.

Chapter 7 builds on the main results of the previous chapter and designs use cases and applications while keeping and eye on the landscape of business needs.

Chapter 8 closes the thesis by providing a summary of the achieved results and discussing potential future works.

Chapter 2

Theoretical Background

2.1 Question Answering (QA)

"Wieso, weshalb, warum? Wer nicht fragt, bleibt dumm."

— *Sesamstraße*

Everything is ultimately a form of Question Answering (QA)—all human problems can be cast into that format. Within computer science, QA is an interdisciplinary field of Text Mining, Information Retrieval (IR) and computational linguistics (NLP). Its aim is to *automatically* answer questions posed in natural language by human users. QA can take any dimension and semantic complexity. From simple questions about numbers, dates and names over grammatical co-references to more abstract semantic questions, like "What is the meaning of life?", QA has no limits by default. Because of its wide span, it is one of the most difficult tasks in the field of Text Mining. Parsing, Spam Detection in Emails, Sentiment Analysis and the likes of it have been solved successfully in recent years, but overall QA is still counted among the partly-unsolved tasks of Text Mining. Nevertheless, the advance of QA is closely related to the progress of AI. Since the rise of neural networks, we have some dramatic changes in this field. With the "toy-tasks" of Weston et al. [35] and with the publication of the SQuAD dataset of Rajpurkar et. al [25] some major steps towards structuring and categorizing QA tasks have been taken.

Classic Methods Traditionally, the task of QA has been tackled with Information Retrieval (IR) methods. Usually a structured knowledge base (e.g. SQL database, RDF graph database) has been utilized. The main working direction has been to transform the natural language question into a formal query which

can be dealt by a machine. Examples for such systems are IBM Watson and WolframAplha, which internally map natural language questions to machine queries. Since the *Deep Revolution* there has been a shift in this field. It departs from a technical and "feature handcrafting" approach to one which appears to be more promising for the future of AI.

Current trends towards human-like QA Since the rise of neural networks, we can observe a change of trends in the research community. Machine Reading Comprehension (RC) is becoming more successful. The RC task is to design systems which learn to read text and "comprehend" its content such that it can be utilized for further tasks—just like humans when they read an exam paper and answer arbitrary questions on it. Particularly, in the field of QA, we focus on the following set up. Given a text and a question both in pure, natural language, the goal is to find the answer out of the raw text. In other words, to locate the subsequence which contains the answer to the question. We call this class of tasks *Passage-QA*, i.e. *steps towards QA on text passages*, rather with the help of any (semi-) structured knowledge sources. This is the focus of our thesis.

2.2 Artificial Neural Networks

> *"The thing that hath been, it is that which shall be; and that which is done is that which shall be done: and there is no new thing under the sun."*
>
> — *Ecclesiastes*

Neural networks have been inspired by nature. They have a long lasting history. The mathematical problem of curve fitting for N data-points (i.e. linear inter- & extrapolation) can be seen as precursor of neural network applications. Hence, the practical usage of neural networks dates back to the early days of mathematics.

2.2.1 Biological Background

In biology and neuroscience, neural networks are known to be the core element of our nervous system and thus of our ever important human intelligence. Their behavior is yet to be fully understood, but what we know is that *neurons that fire together wire together* [11]. Neurons are brain cells with a complex structure as depicted in Figure 2.1(A) and (C). These cells communicate to each other through synapses and build sophisticated networks of neurons. A single neuron starts to forward its own set of signals to all the neighboring neurons once their incoming signals cross a certain threshold of activation. The important principle herein is that of *adaptive learning*. The neurons and their complex connections (i.e. the

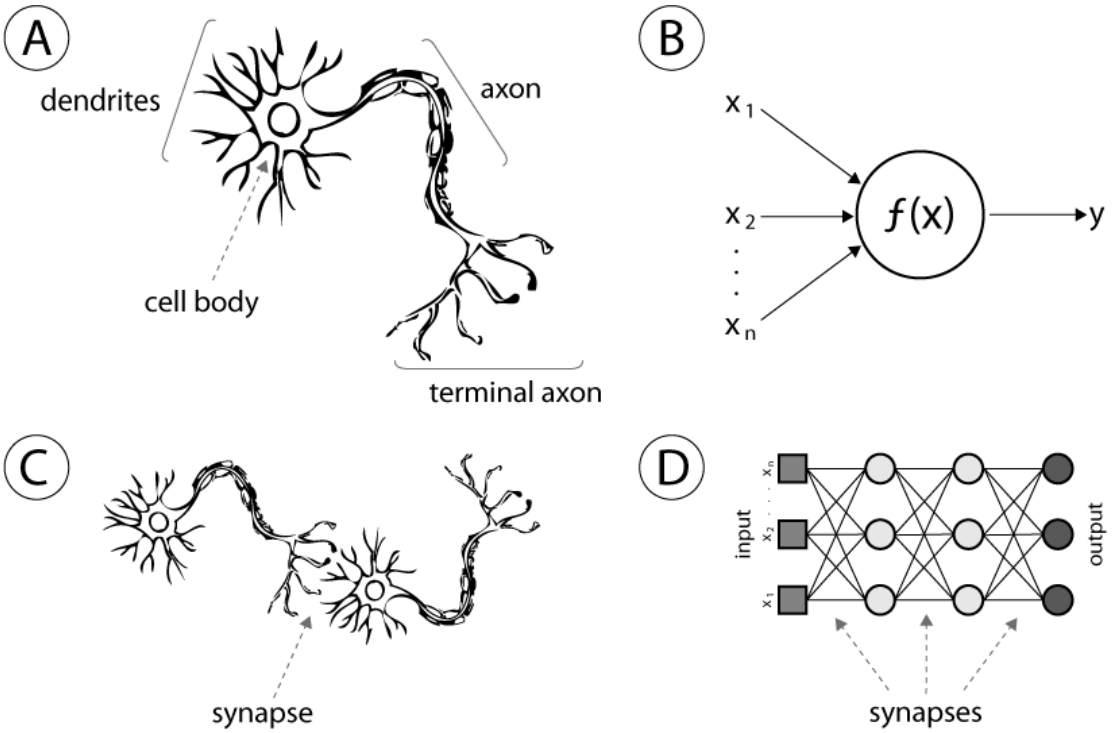


Figure 2.1: (A) neuron (B) abstraction of neuron (C) natural neural network (D) abstraction of neural network (adapted from [20])

neural network) learn to use the incoming sensory information in such a way that they arrange themselves accordingly to efficiently produce the desired output. Our brain is a complex system which can dynamically change its structure according to the incoming needs.

Artificial Neural Networks are an abstraction of this biological phenomena. Figure 2.1 shows the level of abstraction for a single neuron (B) and for a connected network (D) with the input/output layers and two hidden layers. An artificial neuron has n inputs. It fires when the input $\mathbf{X} = (x_1, ..., x_n)$ activates its computable activation function $f(X)$ and delivers the target output value y. Typical activation functions are modeled with *sigmoid*, *tanh*, *softmax*, and more recently with *Rectified Linear Unit (ReLu)*.

A single neuron can solve many algebraic tasks. However, already for the XOR problem, which is a classification task of linearly inseparable inputs in two dimensional space, one neuron is not sufficient anymore. The task is solved only with the addition of more neurons. Therefore, various network architectures of neurons arise displaying complex hierarchical structures in both horizontal and vertical directions.

Scholars have talked about a zoo of neural network architectures, as shown in Figure 2.2[1]. In such a zoo, orientation becomes difficult—for beginners as well as for experts[2]. Feedforward (FF) neural networks (see Figure 2.1(D) and FF in Figure 2.2) have a simple architecture where the data flows into one direction from input to output. This class can solve many linear problems. However, once long-time dependencies in data become important for solving more complex tasks, the necessity for more elaborated architectures arises.

2.2.2 Recurrent Neural Networks (RNN)

RNN is a class of neural networks which has installed neurons with *recursive loops* in its design (see RNN in Figure 2.2), allowing it to remember things from the past by forwarding them back to itself. It somehow resembles the development of automata theory in theoretical computer science, where the introduction of self-loops to finite-state-machines enables further possible computability to a wide range of mathematical problems across the complexity classes and their correspondences along the Chomsky hierarchy. For the field of text mining, the recurrent feature is of particular importance as textual input data consists of sequences with long-term dependencies.

A technical issue with RNN arises when the sequence length becomes very long: the problem is known by the term *exploding/vanishing gradient* [9]. Usually, when a certain information value is carried forward into to the future across ongoing time steps, it is multiplied repeatedly to the weights of the recurrent cell. In practical applications, typically after 10 time steps, the weights are multiplied so often that, depending on the eigenvalue of the weight matrix, either its gradient starts to get very large or shrinks very much. In either case, the weights slowly squash the stored value leading to bad training performance and overall difficulties while applying the RNN. In the text domain, there are any extremes possible, we might have one important fact mentioned on the first page of a book and its second dependency on the last one. Hence a better architecture becomes necessary for dealing with such long-term textual dependencies.

LSTM [13] has shown to be that architecture that solves the gradient problem. The incorporated memory cell saves necessary information for long-terms without exposing them to the repeated multiplication during training with backpropagation through time. Since the best performances of LSTMs on various Speech, Language

[1] http://www.asimovinstitute.org/neural-network-zoo/

[2] There is still no foundational theory about the mathematics of neural networks, hence some people prefer a black-box approach for deploying neural networks and their (pre-trained) subcomponents. However, this strategy requires various trials and errors until a functioning system is modeled. A deeper analytical understanding stays essential for an efficient research and development in this direction.

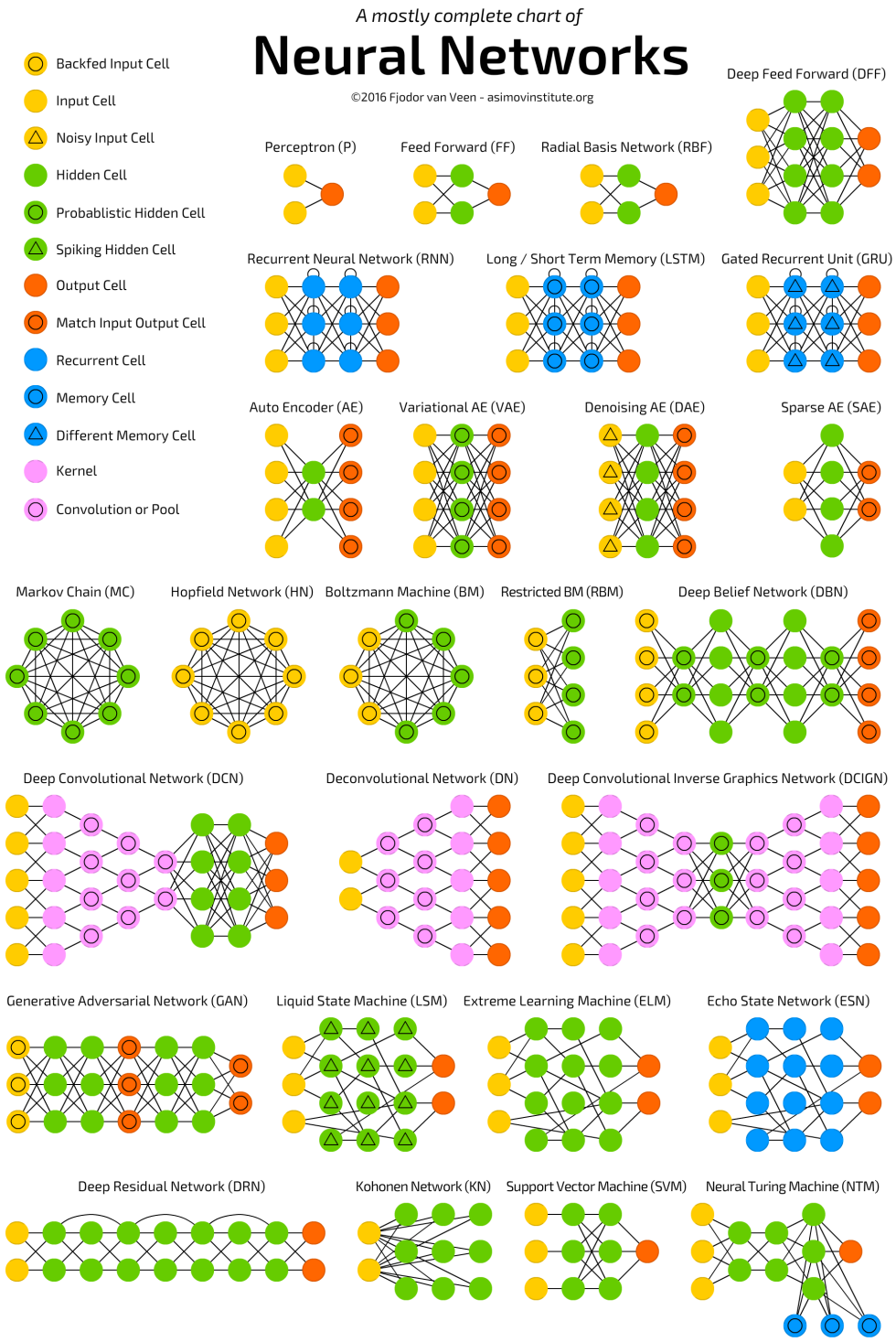


Figure 2.2: Zoo of neural networks. In our work, we use a variation of LSTM and a related version of NTM.

and Object Character Recognition tasks, further development of this class is on rise with various new extensions like the MemNN with an external memory block. In our study, we will use this overall class of neural memory networks and give further details in the next sections.

Chapter 3

Related Literature

Traditionally NLP tasks like Part-of-Speech Tagging, Named Entity Recognition, Sentiment Analysis, Machine Translation, Natural Language Inference (NLI), and ultimately QA are performed with a two-stage approach [39]. First, extracting handcrafted features out of raw text typically with bag-of-words or bag-of-ngrams and their TF-IDF. Next, classifying these features typically with some linear classifiers, Support Vector Machines (SVMs) or Conditional Random Fields (CRFs) to get the final results [39]. In the recent past, neural networks have been used with great success in a wide variety of NLP problems and predominantly replaced the traditional method by establishing a new state-of-the-art. Here, the tedious task specific feature engineering has been replaced by the *word vector* conversion and a neural network is able to extract features out of it itself. With this the two steps of classic methods have been joined by an end-to-end trainable neural network, where a conversion of raw text to word vectors is typically the first step in the neural architecture.

In the domain of QA, the usage of neural networks is relatively new. Within the class of *QA on Passages*, two major lines of research have formed out in academia where neural models are applied to some extent. We will describe the first and focus on the second for the remaining part of this thesis.

3.1 QA with Knowledge-Base

As described in the Introduction, this line of research continues on the traditional direction, where the decisive task is to translate natural language questions into appropriate machine queries. It has replaced some aforementioned traditional NLP components by neural networks like the preprocessing and manual feature engineering of raw text by word vectors. However, the usage of some knowledge-base (KB) and related Information Retrieval (IR) methods are still central to

all solutions, without which no information can be utilized for the final answer generation. There is a growing body of research in this direction. Table 3.1 gives a review of the current state-of-the-art in this line.

Study	Data & Method	Content & Results
A Neural Network for Factoid Question Answering over Paragraphs (QANTA) [14]	QANTA quiz bowl (100k QA pairs) Method: word2vec combined with dependency tree on paragraph (IR method)	Factoid-QA: given a description paragraph of an entity, identify its name; Human-like performance on this task, however no general QA possible
Large-scale Simple Question Answering with Memory Networks [5]	100k *single fact* questions & list of KB triplets (*Freebase*) containing answer resources Method: Word Embeddings of queries and KB entires for MemNN	Good performance & transfer learning possible; but limited application due to usage of KB
Reasoning With Neural Tensor Networks for Knowledge Base Completion [29]	*Wordnet* (113k relational triplets) & *Freebase* (316k relational triplets) Method: train word2vec model of KB entries to predict new relationship triples	"Reasoning over relationships between two entities" in KB by taking "average of constituting word vectors of entities", however QA is limited to relationship classification of KB triplets

Table 3.1: Overview of state-of-the-art QA systems with KB/IR

These studies are not directly relevant to our work and are only mentioned here for comparative purposes.

3.2 QA on Passages (Machine Comprehension)

As stated in the Introduction, we aim to depart from a KB/IR based approach and analyze the usage of neural networks throughly by replacing remaining traditional methods with neural models, thus creating a homogeneous, end-to-end trainable system. In this light, the class of Passage-QA is reviewed here. Forerunner to Passage-QA is the class of cloze-style QA [12]. There, the task is to find the missing entity in a given query about a passage. We can say that cloze-style QA is a special form of Passage-QA where the answer is limited to a singular entity

with a length of one word (i.e. *1-word-QA*)[1].

However, the class of Passage-QA itself is relatively new. Although there were previous attempts in this direction e.g. with the Machine Comprehension dataset of MCtest [26] and its proposed solutions, the QA data was too small to allow the development of useful systems. Therefore, no good performing predecessors are known to us which either use traditional NLP methods [35, 25] or are solely based on neural networks. The publication of the bAbI dataset [35] can be seen as the first major step in this direction breaching the *barrier of big data* necessary for applying neural networks successfully. Only since the release of SQuAD [25] during the preparational work of this thesis in 2016 there has been a general rise of this class of QA.

Two models are reviewed here for both these datasets, namely the MemN2N for bAbI and the Match-LSTM for SQuAD. These are the two main pillars of our thesis. Common to both models is the usage of memory: Due to parameter sharing a neural network with memory is able to remember relevant facts from long sequences (i.e. passage) in order to answer questions on it. This property of capturing long-term dependencies is important in sequence-to-sequence learning. We differentiate on variations of local vs. global memory component in the neural architecture. We will give an overview of their related literature here and later in chapter Methodology describe both models more in depth.

3.2.1 Local Memory: Long Short-Term Memory Networks (LSTM)

The variation of a neural network with local memory represents among the simplest form of a neural architecture where some kind of memory is used. This idea is implemented with the model of Long Short-Term Memory Networks [13]. Every LSTM cell has its own memory, and decides independently how to use it. On the level of local memory, LSTM has been successfully applied to various sequence-to-sequence problems achieving state-of-the-art performance. Natural Language Inference (NLI) is such an example.

NLI is related to QA and is its pre-requisite. NLI has been solved with various different LSTM architectures. Notable is the study of Rocktäschel et al. [27], applying the important concept of attention mechanism of Bahdanau et al. [3] on top of the basic LSTM model. The study of Wang et al. [33] improves the system of [27] through introducing a sequence matching architecture to the existing LSTM, thereby creating the so called Match-LSTM. Next, they extend it to solve the

[1]Coincidently, we also employed a similar approach while analyzing MemN2N for the simplified version of SQuAD data, i.e., we transformed the SQuAD task to the cloze-style task before applying MemN2N on it.

Passage-QA task while attaching a Pointer Network [32] on top of it and achieve among the best performances on SQuAD [34]. This system is one of the central pillars of our thesis. We examine its first part of Match-LSTM and dig deeper into it.

3.2.2 Global Memory: Memory Networks (MemNN)

Memory Networks are a recent class of RNN which contain a global memory component. The distinctiveness of their architecture is the usage of an external memory component. The LSTM also contains a 1-bit memory component (a *memory cell*), but for long sequences, larger memory become necessary due to obvious reasons. The only way to increase the memory is by increasing the number of LSTMs, so that subparts of a sequence can be better processed. In this scenario, each cell has its own memory component and thereby acts independently for each subpart. However, unlike the LSTM, a MemNN treats the memory as an external global (static) component (i.e. it can increase its size independent from the remaining architecture) and learns to read from and write to it for the whole sequence. This is comparable to humans making notes while answering questions on an arbitrary exam paper. The notemaking is done on one paper (MemNN), rather than on many small chits (LSTM). It is claimed by [36] that the approach of MemNN is more suitable than the one of LSTM for storing facts from longer sequences. We want to examine their claim.

Nevertheless, it is the first system of its kind on a large scale which has been successfully applied to QA tasks. Although there have been earlier attempts at using a global memory, there are not many predecessors of it which use this for machine comprehension and QA. For example the Neural Turing Machine (NTM) is a closely related neural architecture with a global memory [10]. However, the overall design of NTM is made for solving basic algorithmic tasks like copying or sorting of sequences, therefore in its current form it is not suitable for QA. We focus on the MemNN architecture—the only choice available up to date. Following variations exist of this class:

Memory Networks [36]: MemNN is the original version within this family of neural architectures. It solves the QA toy tasks, however, while training the model it needs strong supervision at each layer. Internally, this is done by applying max operations at each layer with the help of supporting facts, rather than using softmax operations which would make the model end-to-end differentiable. Hence, it is not effortless to train the system with the algorithm of backpropagation.

Dynamic Memory Networks [18]: The DMN is an extended version of MemNN with additional modules. It improves the basic model and solves the bAbI QA toy tasks with higher accuracies, however, it still requires (partly) strong supervision. Again, it stays difficult to train the system with the algorithm of backpropagation.

End-to-End Memory Networks [30]: The MemN2N is another extended version of MemNN. It successfully solves QA toy tasks. In contrast to DMN, the key difference here is that in this extension the pointers for string supervision are not needed. Instead of using supporting facts at each layer with a max operation, the MemN2N rather uses softmax thereby relinquishing the need for strong supervision and allowing a smooth training of the network. MemN2N becomes end-to-end differentiable, allowing an easier setup for training. In this thesis, we focus only on this form of neural architecture, as no extensive task specific feature engineering of training data is required and we can better test the extendibility of the overall MemNN class to realistic standard language tasks[2].

[2]DMN+ [37] is an extension of DMN and combines the key property of MemN2N, making it end-to-end trainable as well. DMN+ solves the bAbI QA toy tasks and is also applied to Visual QA tasks. However, due to its larger scope, we only experiment with MemN2N and leave DMN+ for future research.

Chapter 4

Methodology

In this chapter, the methods and tools are described which are used throughout the thesis for examining the selected datasets of chapter 5. We start with describing the foundational method of *Word2ec* which is vital for processing any kind of text data, and continue with the two neural network architectures of *LSTM* and *MemNN* which were considered for performing the ultimate analysis for answering the research questions RQ-1 and RQ-2. Finally, this chapter closes by shedding light on the technical aspects of software engineering underlying this thesis.

4.1 Word2vec

The language model of continuous space word representations is the foundation of all current works with neural networks in the domain of natural language processing (NLP) and Text Mining. The concept was already presented by [4]. Since its overall improvement in usability in year 2013 [21], the development of text mining based on neural networks has increased rapidly.

The neural network based model of [21] converts words to a high dimensional vector model, based on the surrounding context of the words within the input texts. The fact that this method is unsupervised makes it very attractive for various domains, as the manual labeling of raw data is often a costly and time consuming procedure.

The skip-gram model is used as the core element of Word2vec. Figure 4.1 provides a sample overview. This neural model is trained by maximizing the following equation[1]

$$P(w_t|h) = softmax(w_t \cdot h) \tag{4.1}$$

[1] https://www.tensorflow.org/tutorials/word2vec/

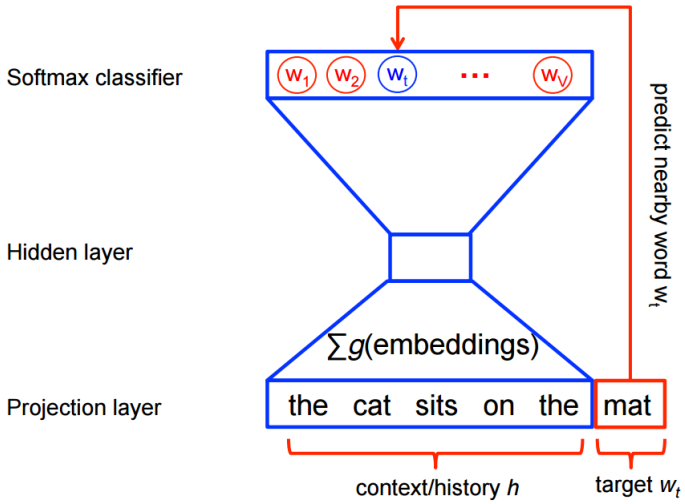


Figure 4.1: Word2vec CBOW model for a sample sentence[2]. For the skip-gram model the situation is simply mirrored by switching h and w_t.

where w_t is the target word vector and h the context word vectors. The aim is to maximize the probability of the next word w_t given the surrounding context words h. Mathematically, this is done by applying the principle of maximum log-likelihood with optimizing the equation

$$max(\log P(w_t|h)) = min(-\log P(w_t|h)). \tag{4.2}$$

Semantics of words and phrases are captured by this model to such an interesting extent that algebraic operations (*analogical reasoning task*) on certain dimensions of the word embeddings generate meaningful relationships. This can be seen from the popular example of *word2vec(king) - word2vec(man) + word2vec(woman)* which yields the embedding vector *word2vec(queen)*. This property is immensely useful for our further application of QA. Thus, we chose the Word2vec model as the first step in our computational pipeline of this thesis.

Unseen words In the recent research, instead of training a word2vec model from scratch, people prefer to use pre-trained word embeddings of high quality for saving computation time and energy. In this thesis, word embeddings trained on the *Google News* dataset[3] (about 100 billion words) are applied. While using such models an important issue should be kept in mind. Within the new dataset under investigation there are such words which are not part of the original text corpus on

[2]https://www.tensorflow.org/tutorials/word2vec/
[3]https://code.google.com/archive/p/word2vec/

which the word embeddings are pre-trained, the so called out-of-vocabulary (OOV) words. Sometimes these unseen words might be those rare key terms which are important for the semantics of the whole text! A simple technical solution is to map all OOV-words to a *NULL word vector*. More sophisticated solutions have been discussed in the literature [19], like calculating the mean vector out of next, related word vectors in the existing model. However, we prefer the vanilla version of using NULL vector, as no comprehensive solution is yet available.

4.2 Match-LSTM

Since the development of RNN, many different forms of it have been proposed in the community. Experts have talked about a zoo of architectures. Noteworthy is the LSTM [13], which itself has also a huge number of variations. We describe variant used in [33] used to solve the task of Natural Language Inference (NLI) on the SNLI dataset (see section 5.3). For NLI the input consists of two text sequences and the output is a classification of their relationship. This variant is central to our thesis.

4.2.1 Basic LSTM

The basic structure of an LSTM is as follows: It consists of five neural elements: gates of (1) input i_k (2) output o_k (3) and forget f_k, the (3) hidden state h_k and most importantly (4) the memory cell c_k which can save the previous state h_{k-1}. This construction allows the network to learn to either remember important facts from the past or forget those which are irrelevant for solving tasks with sequences containing long-range dependencies. Figure 4.2 gives an overview of the model. Let $X = (x_1, ..., x_N)$ be an arbitrary input text sequence where each $x_k \in \mathbb{R}^l$ is an embedding vector (with dimension l) of the underlying raw word. The embeddings can be created with the word2vec model of the previous section 4.1. The following equations define the LSTM architecture and its temporal behavior for transitions at each location k:

$$i_k = \sigma(W^i x_k + V^i h_{k-1} + b_i) \tag{4.3}$$

$$f_k = \sigma(W^f x_k + V^f h_{k-1} + b_f) \tag{4.4}$$

$$o_k = \sigma(W^o x_k + V^o h_{k-1} + b_o) \tag{4.5}$$

$$c_k = f_k \odot c_{k-1} + i_k \odot \tanh(W^c x_k + V^c h_{k-1} + b^c) \tag{4.6}$$

$$h_k = o_k \odot \tanh(c_k) \tag{4.7}$$

with σ being the sigmoid function defined as $\sigma(t) = 1/(1+e^t)$, and $\odot$ the element-wise multiplication of vectors. All model parameters $W \in \mathbb{R}^{d \times l}$, $V \in \mathbb{R}^{d \times d}$ and $b \in$

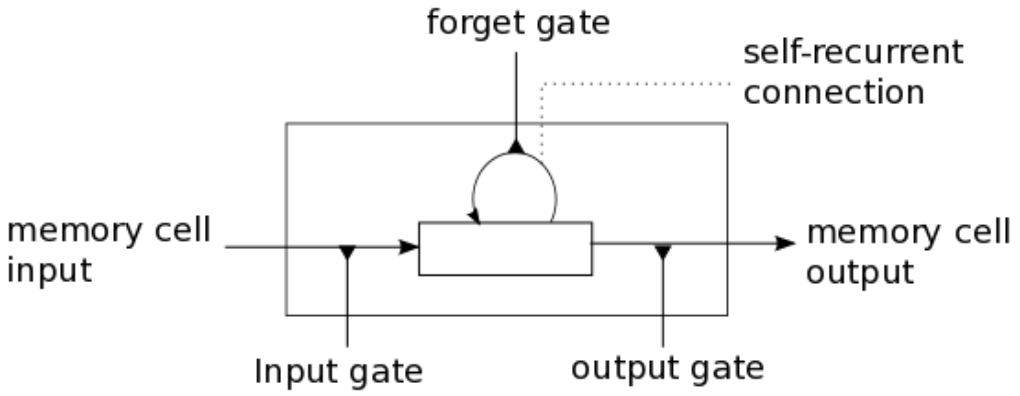


Figure 4.2: Basic architecture of the LSTM[4]. The *memory cell output* is the the hidden state h_k of Equation 4.7.

$\mathbb{R}^d$ (with d being the dimensionality of all hidden states) are randomly initialized and learned during the training.

4.2.2 Model Details

Wang et al. [33] extend the basic LSTM of Rocktäschel et al. [27] to an architecture of *Match-LSTM* (mLSTM) which introduces a matrix-like sequence matching mechanism. In the following we give a sketch of the extension. Let $P = (p_1, ..., p_N)$ be the textual sequence of the passage and $Q = (q_1, ..., q_M)$ the sequence of the question, where p_k $(1 \leq k \leq N)$ and q_j $(1 \leq j \leq M)$ are the respective embedding vectors. Furthermore, let h_k^p and h_j^q be the corresponding hidden representations of p_k and q_j once they are processed by an LSTM cell. The extension of mLSTM matches the hidden representation of every word of the question $q_j \Rightarrow h_j^q$ to one passage element $p_k \Rightarrow h_k^p$ over the neural attention model defined as

$$a_k = \sum_{j=1}^{M} \alpha_{kj} h_j^q \tag{4.8}$$

with

$$\alpha_{kj} = \frac{\exp(e_{kj})}{\sum_{j'} \exp(e_{kj'})} \tag{4.9}$$

$$e_{kj} = w^e \cdot \tanh(W^q h_j^q + W^p h_k^p + W^m h_{k-1}^m) \tag{4.10}$$

[4]http://deeplearning.net/tutorial/lstm.html

where $\cdot$ is the scalar product of two vectors. The vector $w^e \in \mathbb{R}^d$ and all matrices $W \in \mathbb{R}^{d \times d}$ are model parameters which are learned during the training. The *mechanism of neural attention* is the second important pillar in the recent success of sequence-to-sequence learning. Although the name itself can be misleading—it has nothing to do with human form of attention, which is by far more complex than most other mechanisms—the functionality is very useful for our problem. It allows to match words on the embedding space and *focus* on important semantic matches necessary for solving the learning tasks of QA.

The proposed architecture of mLSTM combines both components of LSTM and attention layer. With this the mLSTM architecture extends the Equation 4.3 to Equation 4.7 as follows:

$$i_k^m = \sigma(W^{mi} m_k + V^{mi} h_{k-1}^m + b^{mi}) \tag{4.11}$$
$$f_k^m = \sigma(W^{mf} m_k + V^{mf} h_{k-1}^m + b^{mf}) \tag{4.12}$$
$$o_k^m = \sigma(W^{mo} m_k + V^{mo} h_{k-1}^m + b^{mo}) \tag{4.13}$$
$$c_k^m = f_k^m \odot c_{k-1}^m + i_k^m \odot \tanh(W^{mc} m_k + V^{mc} h_{k-1}^m + b^{mc}) \tag{4.14}$$
$$h_k^m = o_k^m \odot \tanh(c_k^m) \tag{4.15}$$

with $m_k = \begin{bmatrix} a_k \\ h_k^t \end{bmatrix}$ being the straightforward concatenation of attention and hidden vector of the k^{th} word p_k from the passage. Finally, out of the matrix

$$\mathbf{H^m} = (h_1^m, ..., h_N^m) \Rightarrow h_N^m \tag{4.16}$$

the last column vector is used for predicting the target y for a given input pair (P, Q). In section 6.3, we further explore this hidden vector unit h_N^m and its semantic properties and usefulness for additional tasks (as part of our thesis) and further postulate a semantic summarization property of it.

4.3 End-to-End Memory Networks

The artificial QA data described in next section 5.1 forms the foundations of the End-to-End Memory Networks (MemN2N) architecture. As it will be discussed, the toy-tasks proposed therein try to test different aspects of question types which an intelligent system can deal with. The proposal is noteworthy, however as the name indicates the distance between this artificial task and some real-world scenarios seems to be still big enough. In this work we want to examine the behavior of MemN2N when used for tougher tasks (and depending on results eventually find some limitations).

The model is pretty straightforward. The main component of the MemN2N is the global memory. The model is end-to-end trainable, meaning that only

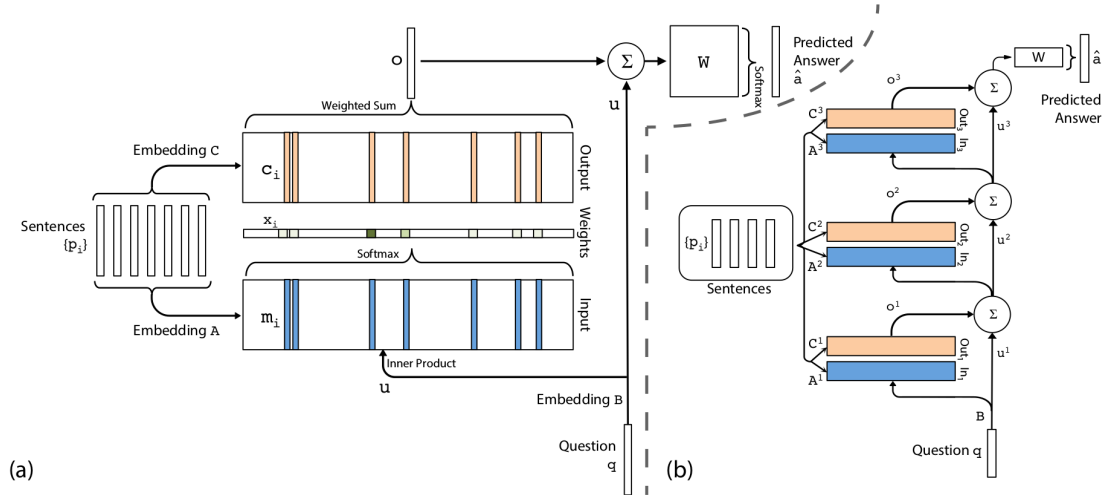


Figure 4.3: MemN2N model (taken from [30]). (a): basic structure for single layer with $K = 1$ hop. (b): whole stacked model with $K = 3$ hops.

input/output pairs are needed to train the system with backpropagating the error into it. For Passage-QA, these are the (P, Q, A) triplets, with (P, Q) pair being the input, and A the target value (i.e. desired output).

4.3.1 Model Details

Let $p = (p_1, ..., p_n)$ be the textual sequence of the passage, $q = (q_1, ..., q_m)$ the question, and $a = (a_1, ..., a_l)$ the answer, with p_i, q_i and a_i being the (raw) *word* coming from a dictionary V (with $|V|$ words). For the bAbI set-up, a is mainly limited to a 1-word answer. Figure 4.3 shows the details of the basic structure (single layer) and the whole stacked model. First, the word embedding of each word is calculated, with two different matrices A and C for the passage and one matrix B for the question.

$$m_i = A \cdot p_i \tag{4.17}$$

$$c_i = C \cdot p_i \tag{4.18}$$

$$u = B \cdot q \tag{4.19}$$

In the embedding space, the match between m_i and u is calculated

$$x_i = softmax(u^T m_i) \tag{4.20}$$

with $softmax(z_i) = \exp(z_i)/\sum_j \exp(z_j)$, interpreting the result as probabilities over the embedded input vectors. Next, these probabilities are taken and matched

to another embedding matrice C (called the *"memory"*), giving

$$o = \sum_i x_i c_i \tag{4.21}$$

This can be interpreted as an overlap of two matrices A and C. Finally, in the single layer case, the *final answer prediction* is done with

$$\hat{a} = softmax(W(o + u)) \tag{4.22}$$

by taking a softmax over the final output o and the question embedding u. The performance error is given by the difference between the model prediction $\hat{a}$ and the true value a.

Above, we just described one such step (i.e. "hop") which allows one-time reasoning over the memory embedding of the passage. For improving the reasoning capability, the basic structure is repeated recursively giving K hops as shown in Figure 4.3(b) thereby allowing multiple considerations of the input passage (through joining neighboring layers with $u^{k+1} = u^k + o^k$). For this, the following two methods are used for tying the weight matrices of neighboring layers.

- Adjacent tying, where neighboring embeddings matrices are the same, (i.e. $A^{k+1} = C^k$), or
- Layer-wise (RNN-like) tying, where irrespective of layers all input and output embeddings are the same (i.e. $A^1 = A^2 = ... = A^K$ and $C^1 = C^2 = ... = C^K$), making this similar to an RNN.

All embedding matrices A, B, C and the output matrices W are part of the greater leaning task and initialized with random values at the beginning. Further model tunings like temporal encoding etc. are applied, as described in [30]. The embedding matrices are not created with the word2vec model as described in section 4.1, rather they are "automatically" created for being part of the whole learning task in the chain of neural network. Hence, this gives an interesting version of the original embedding model.

4.4 Tools (Technical Description)

4.4.1 Implementation Details

In this work, the implementation is done predominately in *Python*. Further core libraries and tools supporting the code development are: *Pandas*, *NumPy*, *Gensim*, *Scikit-Learn*, *Matplotlib*, the *Spyder IDE* and the Linux terminal with *Emacs/Vim* editors for computing on the remote high-performance cluster.

Python is a high-level programming language which supports multiple paradigms like object-oriented, procedural and functional programming. Especially due to the dynamic typing it becomes simple and short in its programming style when compared to other major languages like Java or C/C++. This allows scientists and developers to quickly implement theoretical ideas into working code, making it alongside *R* a favorite candidate for machine learning. In particular in the field of deep learning, Python is by far the most popular language as all major deep learning libraries offer Python bindings. In these days, neural networks are usually deployed with the help of some libraries which support many APIs necessary for defining, building, training and analyzing neural networks efficiently. One can also individually develop everything from the scratch, but the technical performance and scalability can become an issue. Most open-source libraries are highly optimized for performance [28], which is critical due to the large number of data and associated computations involved in this field.

4.4.2 Deep learning software (Tensorflow & Theano)

The implementation of neural networks is conducted in Google's computational framework of *TensorFlow* [1]. A typical implementation in TF has two-stages: First, the computational data flow graph is defined consisting of variables and mathematical operations on them which together model the desired algorithm. Second, in a `tf.session()` the actual constructed graph is run and fed with training data consisting of input and target values. Once the training is completed, TF allows to analyze the final performance of the model and save it as well for later usage. After a detailed investigation and exploration, TF was chosen in favor of other frameworks like Microsoft's CNTK for the first part of MemNN. CNTK promised to be fast and parallelizable, but until late 2016 its interface was not user friendly, as it lacked any dynamic Python interface. Only a cryptic script language for network creation and utilization (Network Description Language) was supported. Furthermore, CNTK lacks a good documentation and a vibrant community like TF. Interestingly, since its recent release (v0.8, April 2016) TF also includes the feature of highly parallelizable distributed computing, hence the major advantage of CNTK becomes obsolete in the light of this new development.

Theano [31] is the forerunner to TF and shares many similarities to it. Like TF, Theano takes the definition of the data flow graph and compiles it down to optimized C/C++ code. Since its recent release, Theano also includes the property of parallelizable distributed computing across multiple GPUs (v0.8, April 2016). Compared to TF, Theano offers more low-level features. For some users this might create some overhead, however, for experienced users this facilitates to have more freedom in their programming. The core function of Theano is the API `theano.function()` which calls a graph for a given symbolic chain of mathemati-

cal expressions and allows to automatically compute its derivatives to perform the training on it. For designing the neural architecture, Theano is supported on top by helper packages like *Lasagne*, which offer various ready-made architectures from the vast field of deep learning. Theano was chosen as the second framework for the part of Match-LSTM to reuse available open-source components and accelerate the overall development.

When is a neural network "deep"? In the deep learning literature, commonly every architecture is referred to be *deep* which has more than one hidden layer. In the case of MemNN with K hops we have variable depth of $(K + 2)$, while with the Match-LSTM we have a fixed depth of 4-5 layers.

4.4.3 Working Environment

In Python (with Pandas, NumPy and Matplotlib), the statistics of the raw data are explored, the raw data prepared and converted into the appropriate training set format. With the libraries of Tensorflow/Theano the network architecture is fitted to the new training cases. Moreover, the partially processed data out of the whole neural network pipeline is extracted for exploring the hidden vectors and performing further analytical investigations on its space. Figure 4.4 gives an overview of the workflow in this thesis.

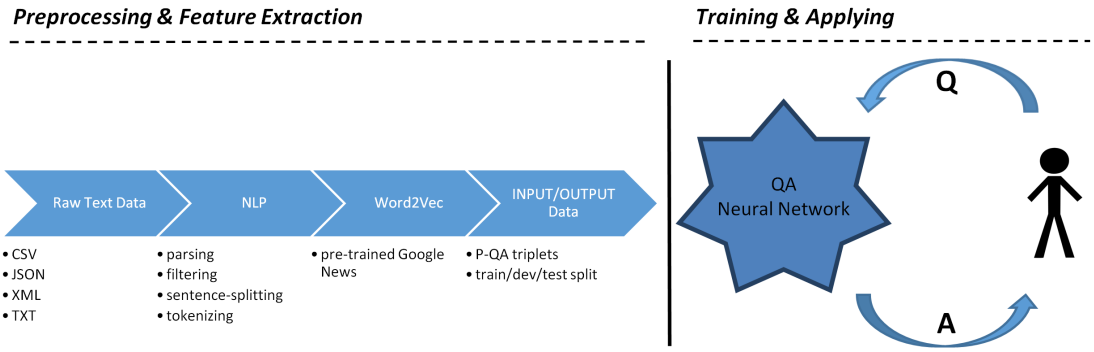


Figure 4.4: Pipeline of Computation for training and applying QA system for steps towards answering user's questions.

Training the data-intensive neural networks requires high amount of computational power and large memory capacity. A demonstrative example from an early stage of the thesis has been with the creation of a simple word embedding model—an essential pre-step for any neural network processing text in natural language: for creating a basic word2vec model with Gensim on SQuAD data, the required

run time on the local machine was 400 mins[5]. On a single GPU the very same computational task was solved within 2 mins only! This early example demonstrated the vital need for GPU Computing in this field.

The current student cluster at Fraunhofer Institute IAIS consists of six compute nodes (*dpl01* to *dpl06*) with each node containing four separate powerful *NVIDIA GPU*s of the family *GeForce GTX TITAN X*. Apart from minor test cases and preprocessing steps, all final tasks of the thesis are performed on the cluster over the Linux command-line interface via a *Secure Shell (SSH)* connection. Each single training task is performed on only one GPU, as so far to the knowledge of the author no neural software supports an automatic and efficient parallelization of neural network optimization across multiple compute nodes. One experiment of training a neural network usually takes at least 1 hour. To parallelize the development, the *GNU Screen* terminal multiplexer was heavily utilized allowing up to $6 \times 4 = 24$ parallel executions of training processes and their various modifications.

[5]Local machine with Intel i7 CPU@1.73GHz, 10 GB RAM on Ubuntu v16.04 LTS. As part of the thesis the RAM has been increased by manually installing two 2 GB RAM cards to facilitate the work with the language model which was hardly fitting onto the machine—not to speak of the whole neural model.

Chapter 5

Data & Preprocessing

In the modern data-driven research, data is the one major driving force behind scientific developments. We say that *data is the gold for scientists*. In this chapter, those text datasets are presented which we have chosen for the methods and tools described in chapter 4. We start with describing two QA datasets *bAbI* and *SQuAD* used for the task of Passage-QA, and continue with a dataset *SNLI* for supporting the preceding step of NLI for QA. Next, for a proof-of-concept and application of paraphrase detection in the later part of this thesis, the MSRP is described. Finally, this chapter closes by giving a short sketch of the dataset *SemEval*, which is taken from the Social Media and plays a subsidiary role in this thesis. All datasets are used throughout the thesis and referenced accordingly.

5.1 Facebook QA bAbI Task

QA is strikingly broad and a vast field of research. Facebook AI Research has taken a step towards structuring and formalizing that field in their *bAbI*[1] project with the ultimate aim of achieving automatic text understanding and reasoning by machines. For this they have published a list of (toy-)task categories (question categories), which according to the authors are the most simple prerequiste for a QA system which claims to be AI-complete [35]. In this way, we can say it is a formalized approach of the more hypothetical Turing Test.

The bAbI project consist of 20 different QA toy-tasks. The data for the tasks are created in a simulated artificial world of actors moving around objects while interacting with them (similar to text adventure games [23]). The goal is that each task test some specific category of (logical) reasoning capability. For example the simplest task of Single Supporting Facts (Task 1) checks if a system can find a single queried fact among irrelevant facts out of a list of sentences. Once a task

[1] https://research.fb.com/projects/babi/

is solved by the system, we say that this system contains that specific reasoning capability. (The task is synthetic, meaning that the system solving it does not pre-exist, in contrast to an analytic task, where the data is engineered for a (partly) known system.) With this dataset a contribution to the nowadays trend of data-driven research is made, where such "benchmarks" give a new momentum to the development of new techniques and push the overall advancement of AI.

The format of the bAbI QA tasks is as follows. A list of sentences make up a story S (i.e. passage). To each such story a certain number of question-sentences Q's are associated, each asking independent questions about the story. The answer A to the question is present within the story. A is mainly a string consisting of 1-word (*1-word answer*), and only in some cases we have A as a list of words. In favor of simplicity, we neglect this latter case in the following description. The file format is for the bAbI data file[2] is as follows:

```
ID sentence-text
ID sentence-text
ID sentence-text
ID Q[tab]A[tab]supporting fact ID's
```

Th ID starts with 1 and increases for every next sentence of the whole story S. After a certain number of sentences a question-answer-line (Q-A-line) appears, posing a question Q about the current S, giving the answer A (as described above) and providing supporting fact ID's pointing to the relevant sentences containing the answer (the supporting facts are not required in the extension of MemN2N). These three elements are each separated by a tab symbol. If the ID increases after a Q-A-line, the story continues as well, and further questions can be asked on the larger context. If the ID is set back to 1, a new story starts and the previous content becomes irrelevant. The following Listing 5.1 gives an example[3]:

Listing 5.1: Excerpt from bAbI task 1

```
1 Mary moved to the bathroom.
2 John went to the hallway.
3 Where is Mary?        bathroom        1
4 Daniel went back to the hallway.
5 Sandra moved to the garden.
6 Where is Daniel?      hallway 4
7 John moved to the office.
8 Sandra journeyed to the bathroom.
9 Where is Daniel?      hallway 4
10 Mary moved to the hallway.
```

[2]Ibid.
[3]Ibid.

```
11 Daniel travelled to the office.
12 Where is Daniel?     office  11
13 John went back to the garden.
14 John moved to the bedroom.
15 Where is Sandra?     bathroom        8
1 Sandra travelled to the office.
2 Sandra went to the bathroom.
3 Where is Sandra?      bathroom        2
```

The data consist of 1k-10k samples for English and 10k samples for Hindi, which are not considered in our study. Table 5.1 gives some stats about each dataset for the first four tasks.

task	vocabulary size	avg. sentences	longest sentence	answer
1	19	6	7	1
2	33	15	7	1
3	34	50	8	1
4	14	2	8	1

Table 5.1: Stats for the first four bAbI datasets. The column *avg. sentences* shows the number of average sentences in all passages. Remaining columns show the number in the measure of words.

Especially from the entries *vocabulary size* and *longest sentence* we can see that the bAbI dataset pictures a pretty simple artificial scenario. On this data the neural network model of section 4.3 has very good performance (up to 99% acc.). We will examine how its performance is on realistic QA tasks composed in standard language. No further preprocessing of the data was needed for this.

5.2 Stanford Question Answering Dataset (SQuAD)

The Stanford Question Answering Dataset (SQuAD) is a QA benchmark published by Rajpurkar et. al. [25]. The distinctiveness of this Machine Reading Comprehension (RC) dataset is that, unlike most QA datasets, SQuAD offers alongside the question and its answer also the passage which contains the overall textual context of the QA-pair, especially the relevant answer-sequence out of the passage. A machine solving the RC task should have the ability to *read* (i.e. parse) a text in natural language and *understand* (i.e. utilize) its content in such a way that it can locate/extract the relevant text part/information out of the greater passage [25], like humans do it while answering questions e.g. on arbitrary exam papers. The uniqueness is, that the size of the dataset is large compared to its predecessors and

at the same time has a real-world set up [25], unlike bAbI [35] which is large but lacks the real-world complexity in its language usage and content.

Remember our overall chosen direction was *QA without structured Knowledge-Base* (see RQ-1 and RQ-2). With this dataset, we get list of *Passage-QA* triplets, which is sufficient for realizing the goal of QA where no Knowledge Base is necessary. Here, the necessary amount of (domain) knowledge for answering a question is given by the passage. Table 5.2 gives an example for some P-QA triplets out of SQuAD.

Characteristics of dataset: The SQuAD data consists of passages extracted from 536 English Wikipedia articles[4], on which Amazon Turk crowdworkers have posed various questions and marked the answer-sequence in the passage. The data contains 107,785 P-QA triplets with 23,215 distinct passages. In contrast to bAbI dataset here the passage lengths cover up a wide area, with shortest passages made of 151 characters (around 1 sentence) to longest passages of up to 3706 characters (around 25 sentences). To the knowledge of the author this is so far the largest type of RC dataset publicly available.[5]

5.3 Stanford Natural Language Inference (SNLI)

Natural Language Inference (NLI) stands for the task of inferring the semantic relationship between two texts as either *entailing* (positive), *contradictive* (negative) or *neutral*. This classification task is one of the foundational necessities for understanding the meaning of natural language, from a scale of short dictionary entires up to documents composed of longer sentences and paragraphs [6]. Hence, in our QA & RC world NLI composes an essential part, for being the first capability in the reasoning chain solving the (greater) QA task.

Bowman et. al. [6] have introduced the Stanford Natural Language Inference (SNLI) corpus, which consists of a large amount of labled sentence-pairs. Unlike previous NLI datasets, SNLI is two orders of magnitude larger than all its predecessors of similiar types [6]. In nowadays research with data-intensive systems (like neural networks) [6], the improvement of performance is only achieved after a certain large amount of training data is available. Therefore, this dataset rightly fills the current needs (in this direction) and opens up new doors for public research.

[4] *title-tag* in the raw JSON file

[5] Only the train/dev sets are publicly available in the JSON format. We reorder both datasets and split the test set *test-v1.1* into test/dev by 90%/10% ratio and take the dev set *dev-v1.1* as the unseen test set throughout the thesis.

Passage	Questions	Answers
According to PolitiFact the top 400 richest Americans "have more wealth than half of all Americans combined." According to the New York Times on July 22, 2014, the "richest 1 percent in the United States now own more wealth than the bottom 90 percent". Inherited wealth may help explain why many Americans who have become rich may have had a "substantial head start". In September 2012, according to the Institute for Policy Studies, "over 60 percent" of the Forbes richest 400 Americans "grew up in substantial privilege".	Q1: Who owns more wealth than the bottom 90 percent of people in the U.S.? Q2: What may explain why some Americans who've become rich may have had a head start?	A1: richest 1 percent A2: Inherited wealth
Little is known about the bacteria that degrade cellulose. Symbiotic bacteria in Xylophaga may play a role in the degradation of sunken wood; while bacteria such as Alphaproteobacteria, Flavobacteria, Actinobacteria, Clostridia, and Bacteroidetes have been detected in wood submerged over a year.	Q1: What type of bacteria are present in Xylophaga? Q2: How long was the wood submerged in water in the study that discovered the types of bacteria in it?	A1: Symbiotic A2: over a year
Proportionality is recognised one of the general principles of European Union law by the European Court of Justice since the 1950s. According to the general principle of proportionality the lawfulness of an action depends on whether it was appropriate and necessary to achieve the objectives legitimately pursued. When there is a choice between several appropriate measures the least onerous must be adopted, and any disadvantage caused must not be disproportionate to the aims pursued. The principle of proportionality is also recognised in Article 5 of the EC Treaty, stating that "any action by the Community shall not go beyond what is necessary to achieve the objectives of this Treaty".	Q1: How long has Proportionality been recognized as one of the general principles of EU law? Q2: Where is the principle of proportionality recognized in the EC treaty? Q3: Which measure must be adopted when there is a choice between several?	A1: since the 1950s A2: Article 5 A3: the least onerous

Table 5.2: Sample excerpt of the SQuAD raw data data.

Text (Sentence-1)	**Hypothesis (Sentence-2)**	**NLI**
A man inspects the uniform of a figure in some East Asian country.	The man is sleeping.	contradiction
An older and younger man smiling.	Two men are smiling and laughing at the cats playing on the floor.	neutral
A black race car starts up in front of a crowd of people.	A man is driving down a lonely road.	contradiction
A soccer game with multiple males playing.	Some men are playing a sport.	entailment
A smiling costumed woman is holding an umbrella.	A happy woman in a fairy costume holds an umbrella.	neutral

Table 5.3: Example for SNLI raw data.

Details of dataset SNLI consists of 570,152 sentence pairs, each labeled with either *entailment*, *contradiction* or *neutral* by Amazon Mechanical Turk crowd-workers. The sentence-pairs and its labeling are all authored by individual crowd-workers and later "cross-validated" for ensuring high quality. Figure 5.3 gives an excerpt of the SNLI data. No further NLP-preprocessing of the raw data is required.

5.4 Microsoft Research Paraphrase Corpus (MSRP)

Microsoft Research Paraphrase Corpus (MSRP) [7] is currently to the knowledge of the author among the largest paraphrase datasets, consisting of 5800 text-pairs written in text-book language. The text is written in standard/academic language and does not contain any grammatical errors or colloquial tone unlike texts from Social Media (Twitter, Facebook). The corpus was created by collecting thousands of news articles in the Internet, after which the sentence-pairs were carefully assessed and labeled by two human annotators. A third annotators verified the judgments and made the final decision. As a result, a dataset of high semantic quality was produced, without any errors typically arising if some automatic processes are applied while generating the data.

For the task of paraphrase detection in the latter part of this thesis on page 42, the dataset MSRP of is used. The raw data is already preprocessed and can be directly applied. Table 5.4 gives an example of this dataset.

Sentence-1	**Sentence-2**	**Paraphrase**
Singapore is already the United States' 12th-largest trading partner, with two-way trade totaling more than $34 billion.	Although a small city-state, Singapore is the 12th-largest trading partner of the United States, with trade volume of $33.4 billion last year.	1
Qanbar said the council members would possibly elect a chairman later Sunday.	US authorities have said the council would include 20 to 25 members.	0
The largest gains were seen in prices, new orders, inventories and exports.	Sub-indexes measuring prices, new orders, inventories and exports increased.	1
No dates have been set for the civil or the criminal trial.	No dates have been set for the criminal or civil cases, but Shanley has pleaded not guilty.	0
Trading in Loral was halted yesterday; the shares closed on Monday at $3.01.	The New York Stock Exchange suspended trading yesterday in Loral, which closed at $3.01 Friday.	1
Wim Wenders directed a widely praised film of the same name, based on the sessions.	A widely praised film of the same name was directed by Wim Wenders.	0
Earnings per share from recurring operations will be 13 cents to 14 cents.	That beat the company's April earnings forecast of 8 to 9 cents a share.	0

Table 5.4: Example for MSRP raw data. The column *Paraphrase* indicates whether both sentences are paraphrases of each other (1 for YES and 0 for NO).

5.5 Semantic Evaluation (SemEval)

The SemEval2016-Task3 corpus [24] consists of annotated textual data from the Question Answering forum *Qatar Living*[6] containing questions and answers posed by human users about various general life topics in the English language. Additionally, there is also an Arabic textual corpus (Fatwa corpus) coming from a legal and ethical domain, which is not considered for the thesis.

The data consists of a list of questions (thread-question) each containing associated 10 comments (comment-block). Every single comment is assessed by humans (Mechanical Turk workers) for its quality in answering the original question as either *good*, *neutral* or *bad*. Several tasks are posed on this dataset. We give only a sketch of the subtask-a (similar to SemEval2015-task3-a) which comes close to our extended application of Passage-QA in this thesis:

- **subtask-a:** Ranking of answers to questions according to their relevance (question-comment similarity)

5.5.1 Preprocessing

Unlike previous datasets SemEval text is predominantly in its raw form just as it can be found on the QatarLiving forum. The following steps were necessary to clean and prepare the text for the analysis:

1. extracting data from XML file
2. lowercasing the alphabet
3. splitting sentences with *NLTK* sentence splitter
4. removing punctuations & white-spaces

As the data is taken from an online forum, the language has a colloquial nature. Hence a huge number of misspelling and grammatical errors are present in the text, which can pose a problem. However, we do not touch this issue to avoid over-engineering of features.

Following the general observation from the deep learning literature it can be said that the less preprocessing is done for the raw data the better the neural network learns automatically to utilize the full bandwidth of information.

[6] http://www.qatarliving.com/forum

Chapter 6

Experiments & Results

6.1 Match-LSTM for QA

With the realistic data of SQuAD containing standard language usage, we started a quest for a suitable neural architecture in the preparatory phase of this thesis (see RQ-1). While doing so, the Match-LSTM (mLSTM) and PointerNet of [34] became noteworthy. Especially the subpart of mLSTM (with attention mechanism) of the whole QA system is of particular interest, hence in this section we start to examine this component more in depth.

As discussed in chapter Related Literature, the task of NLI can be viewed as the requirement (preliminary task) for solving the greater Passage-QA task. While solving Passage-QA the preceding logical step is to find out, if Q can be answered by a given P. In other words, if P entails Q, it indicates that P contains the textual answer to Q and we can proceed to the next step of finding out where the exact location of the answer-sequence is within P. We term this step towards Passage-QA as the *binary Passage-QA* task.

6.1.1 Conversion of SQuAD to NLI format

To perform the aforementioned extension, we formulate the binary Passage-QA task out of the SQuAD data by engineering the training set as follows: We label the existing (P, Q) pairs (passage-question pairs) in the raw data with *entailment* and take them as matching pair samples. These are those original pairs where the passage contains the answer to the question. Next, for all same questions we take randomly different passages from other not related topics and put them together in a non-matching pair with the label *contradiction*. We make sure that in this new match the passage cannot deliver the answer to the old question by choosing passages coming from different Wikipedia articles (i.e. different title-tag) from the SQuAD data. Table 6.1 gives an excerpt out of this training data.

Passage	**Question**	**Answer**
Only a few contemporary societies are classified as hunter-gatherers, and many supplement their foraging activity with horticulture and/or keeping animals.	What do modern hunter-gatherers use to produce food in addition to gathering?	1
On February 6, 2016, one day before her performance at the Super Bowl, Beyonce released a new single exclusively on music streaming service Tidal called Formation.	Beyonce released the song Formation on which online music service?	1
Although it had equipment capable of doing serious damage, the problem for the Luftwaffe was its unclear strategy and poor intelligence. OKL had not been informed that Britain was to be considered a potential opponent until early 1938.	Beyonce released the song Formation on which online music service?	0
Bahrain's prime minister, Sheikh Khalifah bin Sulman Al Khalifah has been in the post since 1970, making him the longest serving non-elected prime minister.	When did Khalifa first take the post of prime minister?	1
Many early LDs were not manufactured properly; sometimes a substandard adhesive was used to sandwich together the two sides of the disc.	When did Khalifa first take the post of prime minister?	0
The Oklahoma School of Science and Mathematics, a school for some of the state's most gifted math and science pupils, is also located in Oklahoma City.	Where is The Oklahoma School of Science and Mathematics located?	1
In recent years a number of well-known tourism-related organizations have placed Greek destinations in the top of their lists.	What do modern hunter-gatherers use to produce food in addition to gathering?	0

Table 6.1: Sample excerpt of the converted SQuAD training data for mLSTM. The column *Answer* indicates whether the raw passage text contains an answer to the question (1 for YES and 0 for NO). In case the answer exists, it is underlined in the passage for this example.

In this way, we engineer the task of classifying whether a question can be answered by a given passage. This is an extension of the inference problem. We do not achieve it by vigorously amending the neural architecture of the model, rather by engineering the features of the training data for a new learning task (i.e. feature & learning task engineering instead of architecture designing).

6.1.2 Single Training (SQuAD)

We want to explore how well the mLSTM model can be trained for a different, but related task of binary Passage-QA classification with the original setup of NLI. For this performance analysis, an experimental setup of single training on one homogeneous dataset is created, i.e. we only consider the transformed SQuAD data, as described in previous subsection 6.1.1.

We split the data into train/development/test sets according to the common convention in machine learning community of around 80%/10%/10% split. We generate "only" 10k training samples for the first run. The training took 13 min on one GPU. Further training details can be taken from Table 6.2.

data size (train,dev,test)	batch size	learning rate	dropout-rate	output units
10k, 1k, 1k	30	0.001	0.3	2

Table 6.2: Run1: training setup

Results: After 10 epochs of training, the performance stays around **50% acc**. Figure 6.1 displays the training curves.

The result is very close to random performance, given that our trained algorithm has to predict only two outcomes (cf. tossing coin). It shows that the system is not able to solve the learning task successfully.
To improve the training performance, we create a 2^{nd} run with varying the input parameters (see Table 6.3). We increase the size of the training set to a maximum of 88k samples by scooping out the SQuAD data according to our conversion strategy as described in previous subsection 6.1.1, and reordering the train/dev split. By following our conversion strategy it is not possible to get more converted (training) data, as there are no more than 100k distinct questions in the SQuAD raw data which is publicly available[1]. The training took 1.82 hours on one GPU.

[1]By leaving out questions from overlapped articles for a proper train/dev/test split, some questions are omitted.

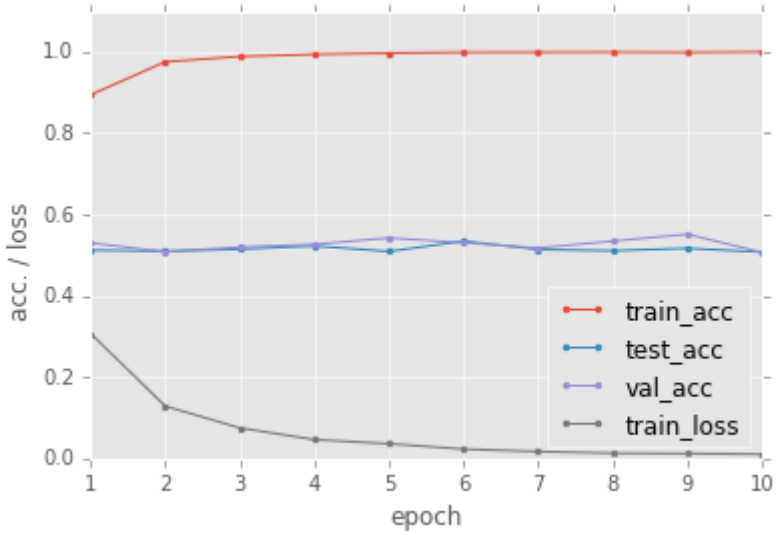


Figure 6.1: Training curves for the 1st run of single training setup with SQuAD. The model does not learn the complexity of data which can be seen from the large gab between *train_acc* and *test_acc* curves. It fails to generalize on unseen data.

Data Size (Train,Dev,Test)	batch size	learning rate	dropout-rate	output units
88k, 1k, 10k	30	0.001	0.3	2

Table 6.3: Run2: varying training params

Results: After 10 epochs of training, the performance reaches the highest value of **71% acc.**, significantly different from a random algorithm. Figure 6.2 shows the training curves and further details. This demonstrates that given enough training data are available the system is able to solve the learning task successfully. With these results mLSTM successfully solved the binary Passage-QA task for the SQuAD dataset.

Further Improvement of Training There are various ways of improving the training of neural networks and closing the gab between the train and test curves. Apart from increasing the training data size, varying the params (dropout, L_2 and learning rates), or making the network architecture simply wider or deeper, one can also apply other sophisticated optimization algorithms like Adagrad [8] or Adadelta [38] apart from our choice of Adam [17]. However, the recent developments in academia show that instead of experimenting with various optimization algorithms, it is better to use (pre-trained) neural networks again on top of the target neural

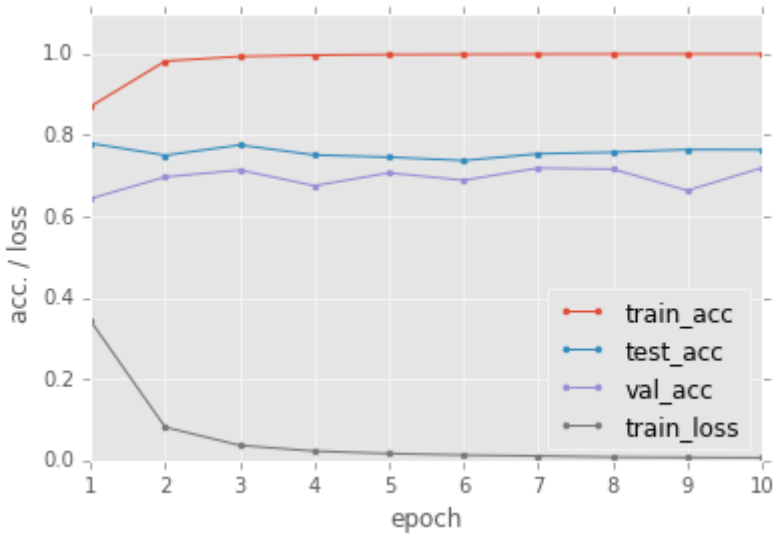


Figure 6.2: Training curves for the single training setup of SQuAD. The model learns quickly with success. The still existing gab between *train_acc* and *val_acc* curves indicate slight overfitting effects. Further improvement of training can close the remaining gab.

model for this specific optimization task (see NIPS 2016).

Keeping the recent development in mind, we covered only the aspect of varying the parameter space and did not explore different optimization algorithms. While varying the epoch length and dropout rate and additionally introducing the L_2-*Regularization* we got further improvements with our best achieved accuracy of **77.39%**. The training has been improved by the following specific parameter choice as described in Table 6.4. All remaining settings stayed the same as in the previous run.

epoch	dropout rate	l2 regularization
20	0.6	10^{-5}

Table 6.4: Tuning training params

6.1.3 Joint Training (SQuAD & SNLI)

In contrast to the previous subsection 6.1.2 of single training, we widen our experiment by reverting back to the original learning setup of NLI and combine it with the new extension of QA. In this setup of joint training, we mix the SNLI data with the converted SQuAD data and create a heterogeneous training dataset.

The aim is to find out, if the mLSTM can solve two different but related tasks simultaneously. The NLI task has 3 possible outputs (entailment, contradiction, neutral) and our binary QA task has 2 possible outputs (YES, NO). Overall, we have a two-stage classification task: First, identifying to which class a new data sample belongs (SNLI or SQuAD), and second, performing the final appropriate classification.

To join both tasks into one workflow, the category YES is mapped to entailment and NO to contradiction. We keep the splitting scheme from the previous subsection 6.1.2. By joining both datasets, we get large number of 600k training samples. The training took 9.86 hours on one GPU for 10 epochs. Further training details can be taken from Table 6.5.

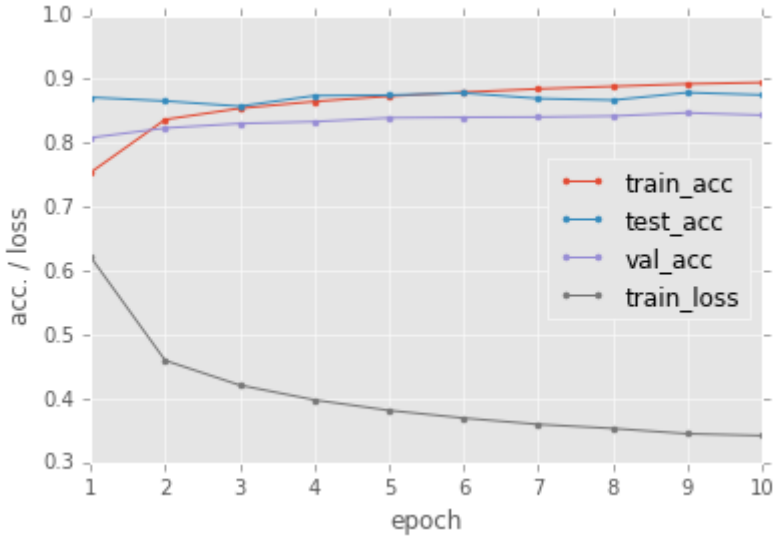
Data Size (Train,Dev,Test)	batch size	learning rate	dropout-rate	output units
636k, 11k, 20k	30	0.001	0.3	3

Table 6.5: Run3: SNLI + SQuAD

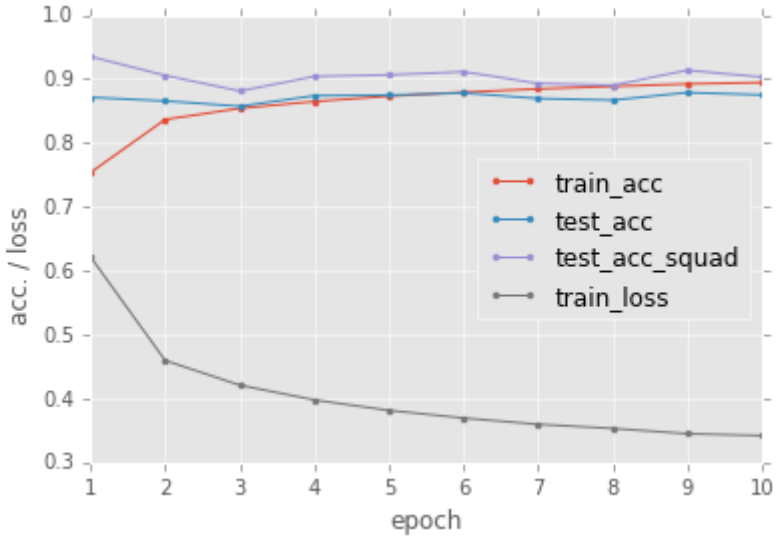
Results: The results are very good. We achieve the highest value of **84.18% acc.** by introducing this variety into the training data. See Figure 6.3 for the training behavior. We can see that the model learns pretty quickly and stays stable above **80% acc.** value after the 3rd epoch. Interestingly, the SQuAD values are always better then the overall value where SNLI data dominates the performance. This is either due to the fact that our SQuAD task is a binary classification task and thus slightly simpler than NLI task—or due to the larger data size of SNLI, which introduces wider textual variety thus making it more challenging.

Overall, this shows that the mLSTM is able to solve two different (but closely related) classification tasks simultaneously. Our model is not totally task dependent, and has the ability to be adaptive to some extent. We successfully created a new extension of mLSTM. We assume that such a joint-training is better, because of the larger training size (compare to subsection 6.1.2, where increasing training size for SQuAD single training improved the results).

Besides, we also experimented with a better alignment of SNLI and SQuAD for the joint training. We removed the neutral category from SNLI by deleting those data samples with the label *neutral*. To our surprise this did not improve the results of the previous setup with mixed data (with three *output units* including one for the neutral category).



(a) Joint training with SNLI+SQuAD



(b) SQuAD focus in joint training

Figure 6.3: The training progression is shown for the joint training setup of SQuAD + SNLI. (a): *train_acc* *train_loss*, *val_acc* and *test_acc* curves show values for the whole dataset. (b): *test_acc_squad* gives an isolated view on SQuAD test set. The model solves the learning task successfully with a high accuracy.

6.1.4 Conclusion

This section shows empirically that the mLSTM is suitable for solving the extended NLI task of binary Passage-QA that involves longer and more sophisticated texts. It shows that a new adaptation can be developed for a given neural architecture, while achieving similar high performance on new training data.

It shows additionally, that for a given learning task, the need for big data in deep learning is essential. If there is not enough training data available, then even the most-sophisticated architecture and choice of parameters (etc.) cannot achieve any good performance.

Various Applications With this we have identified a neural system which is able to help human users to solve the greater task of QA. Especially for situations, where we do not have enough annotated training data, we can use this algorithm for supporting the first step of automated IR. Our system can be viewed as a general supportive tool for QA tasks.

This extended mLSTM for binary QA can be used for searching and assessing relevant passages to a given question. We will discuss further possible arising applications later in more detail. In section 6.3 this functionality is extended and generalized into the *best-replacement* search through the k-NN algorithm on the Hidden Vector Space (HVS).

6.1.5 Applying pre-trained mLSTM to Paraphrase Detection

Overall, it appears that we can train the ability to classify the relationship between two given texts. It might be possible that classification tasks of different types can be tackled with this NLI decider. Further examples in this direction are: paraphrase detection, news categorization[2], article classification etc. While some tasks could be directly solved, some others (like news categorization) have to be reformulated first to fit the original model setup.

We suspect that mLSTM for NLI & binary QA might be expandable to paraphrase detection without any further modifications of the model (see RQ-3). We verify this hypothesis by performing two new experiments of paraphrase detection on our previous pre-trained model of subsection 6.1.3. This *proof-of-concept* will show us further opportunities and limitations of our model.

First, we make a *quick-test* where we simply apply the pre-trained model with SNLI & SQuAD data to the unseen MSRP test data (sample size of 1.7k) without using it for training or tuning the model. The pure testing delivers negative results

[2]I.e. given news and news-type, classify if the news belongs to the news-type.

with a value around **50% test acc.**, which is similar to random performance (disproving our hypothesis at the first attempt). Therefore, we move on to the main experiment: We create a setup of training the pre-trained system further in an extended *triple joint training* (SNLI+SQuAD+MSRP) in order to verify the new training behavior. This is done by simply adding further data of MSRP to the SNLI+SQuAD joined dataset, analogous to the subsection 6.1.3 with SQuAD to SNLI. With this we get a total of 640k train data samples. To analyze the performance solely based on the train set, no MSRP data is included to the val set. All other settings stay the same, as in previous runs (see Table 6.6 for training details).

Data Size (Train,Dev,Test)	batch size	learning rate	dropout-rate	output units
640k, 11k, 21.7k	30	0.001	0.3	3

Table 6.6: Run4: SNLI + SQuAD + MSRP

Results: With a value of **70.57% acc.**, the results on MSRP (test data) improve considerably and paraphrase detection becomes possible. See Figure 6.4 for the isolated training behavior while the joint triple training is performed. It is not only astonishing that the very same model can solve various different tasks, but also the performance itself is considerably good, being close to the state-of-the-art on MSRP (80.4% acc., see footnote[3]).

Our experiment demonstrates the importance of training data. The mLSTM is able to solve even related tasks simultaneously, *only if* the appropriate data is available. *Transfer learning* is a well supported mechanism for mLSTM. Our results fit into the picture of recent reports of Google's NMT model, where a single neural network is able to translate between various languages [15]. With these results the adaptiveness and dynamic extendability of neural networks is reconfirmed in our study.

6.2 Memory Networks

6.2.1 Applying MemN2N to SQuAD

In this section, the performance of MemN2N is analyzed when applied to the dataset of SQuAD for solving the task of Passage-QA (see RQ-2). We design the

[3]https://aclweb.org/aclwiki/index.php?title=Paraphrase_Identification_(State_of_the_art)

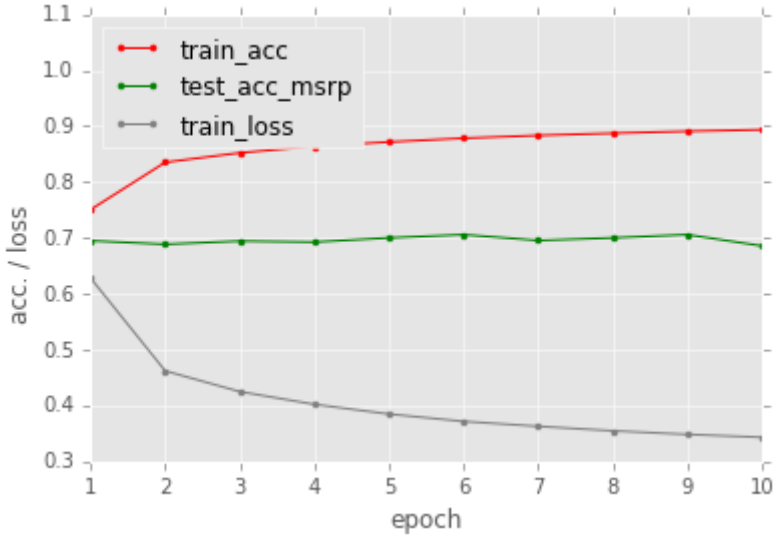


Figure 6.4: Training curves for merged triple training SNLI+SQuAD+MSRP. Although the system is trained jointly, in the *test_acc_msrp* curve the periodic progression data are gathered only for MSRP to get a better focused view.

most simple scenario possible with SQuAD. For this, first we perform a categorization of the SQuAD questions to introduce a structure for analyzing the data according to the bAbI QA tasks (which are mainly limited to *1-word-answers*). We decide to start with a straightforward *word-level* distinction. A better option might have been using some form of semantic categorization, analyzing the actual content of the question and distinguishing at this level. However, we decide to leave this for future research. Interestingly, Wang et. al. [34] have a similar word-level approach, confirming our heuristics.

Categorizing SQuAD questions (and descriptive stats) We start by dividing the questions into the following question categories: *what*, *when*, *which*, *who*, *where*, *why* (wh-words) and *how*. For each type there were composed variants like "how many" or "for what" which we treated as belonging to the "how"- and "what"-category, respectively. Variants which are semantically similar, like "in which year" and "when", were still treated separately in our word-level categorization. Furthermore, we focus only on those Passage-QA triplets were the length of the words in the answer is exactly one (*1-word-answers*). Table 6.7 and its partial visualization in Figure 6.5 give the results for this analysis.

From the table it can be inferred that with our selection criteria, the maximum number of QA-pairs in SQuAD are located around the passage length of 4 sentences. From there onwards the amount of the QA-pairs with 1-word-answers goes

sentences	**qa-pairs**	**unique-a**	***what***	***how***	***when***	***which***	***who***	***where***	***why***
1	662	549	417	93	43	52	27	38	0
2	2157	1655	1338	345	175	154	123	75	2
3	4546	3115	2734	684	438	314	286	196	5
4	6309	4076	3741	955	654	440	402	230	14
5	5750	3835	3284	958	585	448	376	218	10
6	4199	2952	2437	692	443	297	260	178	2
7	2703	2022	1536	411	284	179	225	106	1
8	1639	1314	915	274	159	127	133	59	0
9	915	788	492	149	75	84	78	53	2
10	565	497	311	103	64	34	48	17	1

Table 6.7: Statistics for SQuAD with fixed 1-word-answers per number of *sentences* per passage[4]. The column *unique-a* shows the unique answer words in the dataset. This shows that contrary to the bAbI tasks with a small vocabulary size (see Table 5.1), SQuAD has a realistic scenario already on the answer level.

down. This is reasonable, given that for larger passages the QA task is usually more demanding and consequently requires longer answers. Regarding the structures of the questions, in Figure 6.5 it can be seen that the what-category makes up almost over 50% of all available questions within the selected part of SQuAD, and all other questions together represent the other half. Hence, the what-category solely is an interesting candidate for an initial analysis. Overall, we want to keep the passage length as small as possible to create a very simple scenario out of SQuAD. Therefore, we decide to focus on the subpart with length of *sentences*$\leq$*2*. With this selection (*1-word-answers, what-category, sentence*$\leq$*2*) we get over 1.7k QA pairs out of SQuAD. This is the most simple scenario possible we could have designed for the Passage-QA task.

Conversion to bAbI Format We convert the described selection to the bAbI format necessary for feeding the data into the MemN2N model. The conversion is straightforward: We take the passage from SQuAD and treat it as the storyline. As we have focused on the shortest passages, the structure becomes very simple. A storyline consists of 1-2 sentence, therefore in the following 2^{nd} or 3^{rd} line the Q-A-line appears (see section 5.1). We do not continue the story after a Q-A-line (as there is no content left), and with the next line, a new passage with a new content starts again. With this conversion, the MemN2N can be applied on the SQuAD selection.

[4]Due to the straightforward string search some questions are counted double. However, the number is small enough to be disregarded.

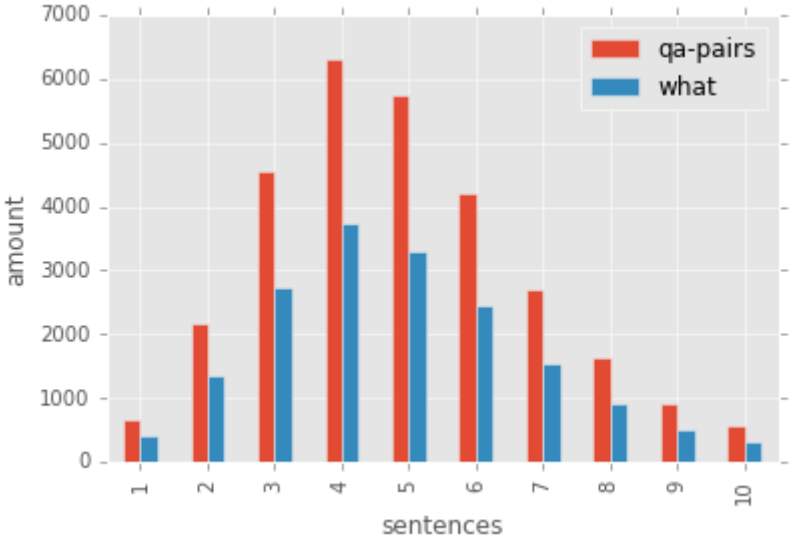


Figure 6.5: Visualization SQuAD stats from Table 6.7.

Results We ran the experiment on one GPU and it took 58 mins. Figure 6.6 shows the training results and accuracies. The results are negative, contrary to our expectations arising from promising statements made by [36, 30]. We achieve a good training accuracy with up to **92%**, however, the test accuracy stays very low and fluctuates around **5%**. This clearly indicates that our model is over-fitting. The model is memorizing the training set, but fails to generalize to the unseen test data. Hence, this model fails the learning task and is not suitable to capture the intrinsic dynamics of the QA data. This gave us the indication that the MemN2N architectures is not suitable for the dataset of SQuAD.

6.2.2 Minor Experiments on SemEval

Before conducting the experiments on SQuAD, we initially started with the SemEval data. It turned out quickly that this direction is not fruitful by any means. For the sake of completeness we present here in short the results.

Conversion to bAbI Format We convert the SemEval data to the Passage-QA format of bAbI (see section 5.1) for inputting it to the MemN2N architecture. Following steps are conducted for the conversion.

1. A comment-block is treated as the whole storyline (i.e. passage).
2. For every good answer A within the comment-block we repeat the whole storyline *excluding all other available good answers* A' and then put the

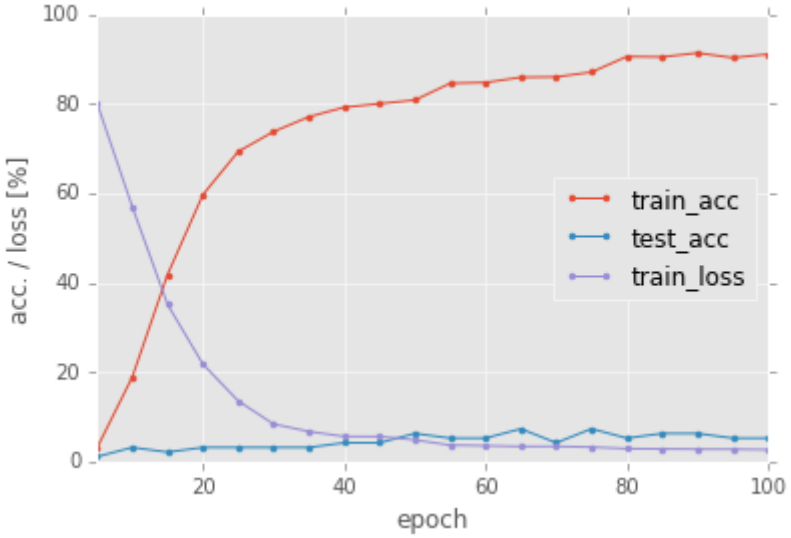


Figure 6.6: Training progression for MemN2N on SQuAD.

current Q-A-line afterwards. In this way, every passage consists of all neutral and bad comments and one specific good answer A for the thread-question Q.

3. Neutral comments are treated like bad comments in this conversion.

With this approach we can extract 2k QA-pairs on 1k passages out of the SemEval data.

Results The training took 2.23 hours on one GPU. The results are negative. We have overall very low values with **0.07% test acc.**. We experiment with various parameters but no change in results can be observed.

Possible Reasons of Failure

One can think of many possible reasons of failure: the large vocabulary, longer sentences and overall the usage of a realistic standard language which has a more complex grammar than the artificial setup of bAbI. However, we assume that the primary reason is the architecture of MemN2N. It does not have the capacity to process long sequences and utilize their semantics. Simply computing the dot-product of passage and question in the embedding space (see section 4.3) does not seem to match the important facts of both text pairs. A more sophisticated word-by-word measure is needed, comparable to the attention mechanism and the matching architecture of LSTM.

Overall, MemN2N is not suitable for the dataset of SQuAD or SemEval for solving the task of Passage-QA. Hence we decided to abandon this family of neural networks and stick to other architectures for realistic QA tasks—like the LSTM and its various extensions.

6.3 Match-LSTM for Semantic Text Pair Search

A number of different (but related) tasks can be solved with the mLSTM model, as it was demonstrated in the section 6.1. The results were good for binary Passage-QA and Paraphrase Detection. The key element for the successful classification was primarily the vector h_N^m. From this end, we postulate that this vector has some semantic summarization property (see RQ-4). We start a deeper exploratory analysis into the nature of this structure.

6.3.1 Constructing the Hidden Vector Space

Let us view the neural conversion in a more abstract way. For a given sample of passage P_i and question Q_i, we get the associated hidden vector $h_{N,i}^m$, after going through the (computation chain of the) neural network. Let us define the function

$$mLSTM(P, Q) \Rightarrow h_N^m \tag{6.1}$$

for this conversion (see section 4.2 for more details). If we compute a further hidden vector of another sample $mLSTM(P_j, Q_j) = h_{N,j}^m$, we construct a vector space on both the hidden vectors $\{h_{N,i}^m, h_{N,j}^m\}$. Hence, we generalize this idea and define the high-dimensional *Hidden Vector Space* (HVS) on an arbitrary number of P-Q samples. We want to analyze the relationship and properties of such vectors in the HVS (comparable to the algebraic analysis on the word vector space [22]). We start experimenting with the cosine similarity as the distance measure. The cosine similarity between two vectors $\vec{X}$ and $\vec{Y}$ is defined as

$$similarity(\vec{X}, \vec{Y}) = \cos(\alpha) = \frac{\vec{X} \cdot \vec{Y}}{|\vec{X}| \cdot |\vec{Y}|} \tag{6.2}$$

$$distance(\vec{X}, \vec{Y}) = 1 - similarity(\vec{X}, \vec{Y}) \tag{6.3}$$

It gives us the cosine value of the angle α between two vectors in the HVS. In high-dimensional spaces, this measure is more appropriate, rather then simply comparing the geometric distance in euclidean metric (of vectors) which can lead to systematic errors due to the *curse of dimensionality*.

We apply this idea to our original setup: for each single (P, Q) sample out of the SQuAD test data (with size T) we calculate the mLSTM conversion on the jointly trained system on SNLI & SQuAD and get a list of T hidden vectors

$$mLSTM(test_{SQuAD}) \Rightarrow \{h^m_{N,1}, ..., h^m_{N,T}\} \tag{6.4}$$

.

6.3.2 Finding Patterns with k-NN Search in HVS

We analyze the properties of the new data in Equation 6.4 and try to find patterns and regularities in it. The k-NN Search[5] is used for this purpose. For each single converted (P, Q) sample in Equation 6.4 the k nearest neighbors (with $k = 4$) are calculated using the measure of cosine distance. As a result a matrix is produced consisting of hidden vectors from Equation 6.4 and their next $k = 4$ neighbors. The resulting density-distributions of cosine distances are plotted in Figure 6.7. The following patterns emerge:

1. Overall, the observation is made that the course of the curves can be explained by two different overlapped distributions.
   - i **Distribution-1**: with mean around 0.95
   - ii **Distribution-2**: with mean around 0.8
   - iii There seems to be a threshold around 0.9, dividing both classes
2. Furthermore, there appears to be a small peak around 1
3. Apart from neighbor1, all other next neighbors show weaker but similar patterns

To analyze the reason behind this structure, we go back to the source texts of h^m_N and look for textual patterns. We focus only on *neighbor1* due to its stronger appearance.

From HVS back to source $(\mathbf{P}, \mathbf{Q})$ texts: By reviewing manually the source text, a dynamic picture emerges. Three major classes are detectable and fit to the evidences in the form of peaks in the distribution of Figure 6.7. After conducting further manual analysis, it was noticed that the underlying density distribution depends on the input (P, Q) data. So if there are many exact matching pairs in the raw text data, then we might have only one peak around cosine=1. In our case

[5] http://scikit-learn.org/stable/modules/neighbors.html

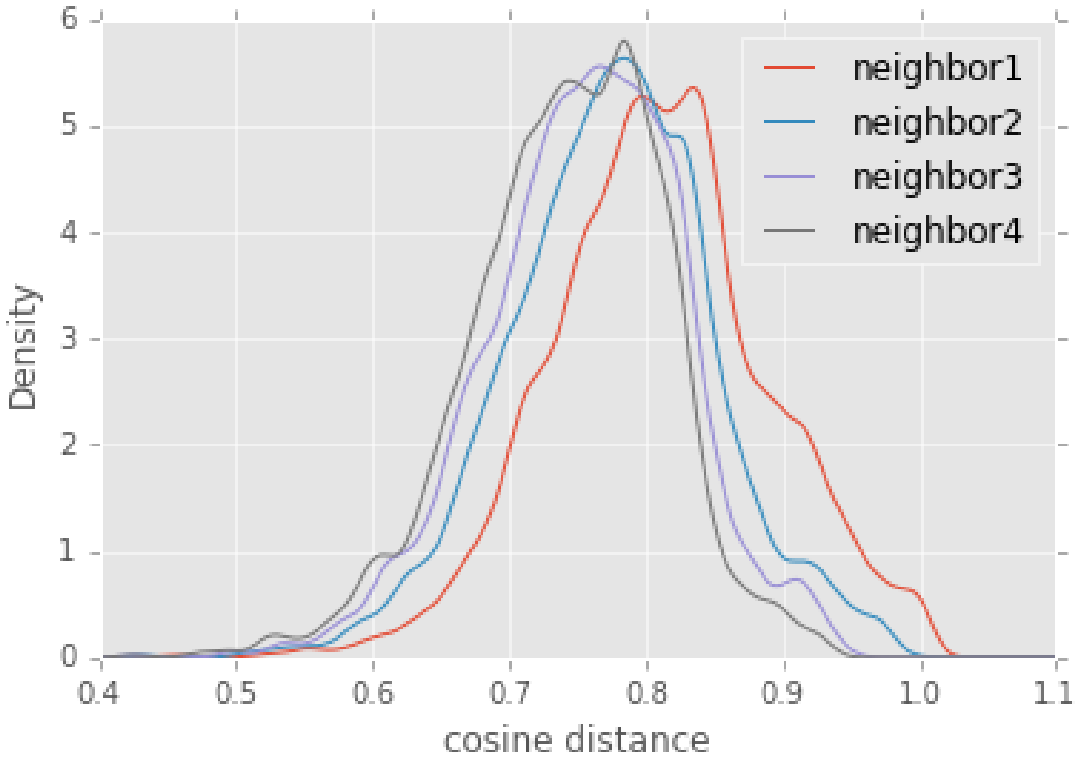


Figure 6.7: Distribution of next h_N^m neighbors for SQuAD test set[6]

here the data is in such a form that we can detect three classes along the scale of cosine similarity. In the following listing, for these three classes and one remaining case a description is provided alongside sample of source text pairs (apart from the 1st class where the situation is obvious).

- **cosine=1.0:** *exact match* of both text pairs.
- **cosine$\in[0.9, 0.95]$:** *partial match*, i.e. one text pair is an exact match (either P_1, P_2 or Q_1, Q_2) and second pair is very similar. This is an interesting aspect for search on HVS and will be utilized later in chapter 7 as the *best-replacement* algorithm.
  - $P_1 = P_2$: *One ctenophore, Mnemiopsis, has accidentally been introduced into the Black Sea, where it is blamed for causing fish stocks to collapse by eating both fish larvae and organisms that would otherwise have fed the fish. The situation was aggravated by other factors, such as overfishing and long-term environmental changes that promoted the growth of the Mnemiopsis population.*
  - Q_1: *What was introduced into the Black Sea?*
  - Q_2: *What event was blamed on the introduction of mnemiopsis into the Black Sea?*
- **cosine$\in[0.8, 0.85]$:** some sort of partial semantic or syntactic similarity exists. Some *common topic* is detectable. In the sample below, it is about "comics and its rise of acceptance in the American society". Both passages P_1 and P_2 are from the same Wikipedia article and are re-matched in the HVS.
  - P_1: *Comics in the US has had a lowbrow reputation stemming from its roots in mass culture; cultural elites sometimes saw popular culture as threatening culture and society. In the latter half of the 20th century, popular culture won greater acceptance, and the lines between high and low culture began to blur. Comics nevertheless continued to be stigmatized, as the medium was seen as entertainment for children and illiterates.*
  - Q_1: *What was thought to be only good for children and those who could not read or write?*
  - P_2: *The graphic — novel-book-length comics — began to gain attention after Will Eisner popularized the term with his book A Contract with*

[6]Due to the Floating-Point Arithmetic some values are erroneously slightly over 1.

God (1978). The term became widely known with the public after the commercial success of Maus, Watchmen, and The Dark Knight Returns in the mid-1980s. In the 21st century graphic novels became established in mainstream bookstores and libraries and webcomics became common.

  - Q_2: *Who helped "graphic novel" get public attention?*

- **cosine<0.8:** few words are similar, but predominantly the syntactic similarity disappears.

  - P_1: *Six-time Grammy winner and Academy Award nominee Lady Gaga performed the national anthem, while Academy Award winner Marlee Matlin provided American Sign Language (ASL) translation.*
  - Q_1: *Who sang the national anthem?*
  - P_2: *In late November 2015, reports surfaced stating that "multiple acts" would perform during the halftime show. On December 3, the league confirmed that the show would be headlined by the British rock group Coldplay. On January 7, 2016, Pepsi confirmed to the Associated Press that Beyonce, who headlined the Super Bowl XLVII halftime show and collaborated with Coldplay on the single "Hymn for the Weekend", would be making an appearance. Bruno Mars, who headlined the Super Bowl XLVIII halftime show, and Mark Ronson also performed.*
  - Q_2: *What was the name of the single that Coldplay and Beyonce collaborated on?*

Overall, it looks like the cosine value gives a nice degree of similarity between two input text pairs. Everything above the threshold of 0.9 has good semantics and a common topic is detectable. For everything below the threshold this property gradually disappears and merely some syntactic matches remain.

With these results we might be able to talk about a *syntactic-semantic similarity continuum*, with a threshold dividing both "spheres" of interest. Once the cosine similarity is higher than this threshold, the word-by-word (or better word-embedding-by-word-embedding) similarity increases to such an extent that an underlying semantics can be seen. Hence, this can be seen as a step towards understanding the nature of textual semantics within QA.

6.3.3 Conclusion: Novel Method for Clustering Text Pairs

The Semantic Text Pair Search in the HVS allows us to cluster related text pairs in high-dimensional space and perform simple algebraic operations on them like calculating cosine distances and applying the k-NN search. Further investigations

have shown that with this method we are actually able to sort back the randomly ordered (*P*,*Q*) pairs to their original source passages, to which they belonged to, and further to those related passages, which came from the same Wikipedia article out of raw SQuAD data[7]. We formally define this newly discovered property of HVS search from a broader perspective.

Formal definition: Given a pair of base text and supporting text (B, S), find the *best-replacement* for S out of some alternative texts $\{s_1, ..., s_r\}$ for a fixed B.

Algorithm 1 best-replacement algorithm

Input: positive sample pair (B, S) and alternative list $l = \{s_1, ..., s_r\}$
Output: reordered best-replacement list $l \Rightarrow l^+$

1: join l with fixed B into tuples $l' = \{(B, s_1)..., (B, s_r)\}$
2: convert l^* and create the HVS with hidden vectors $mLSTM(l') \Rightarrow \{h^m_{N,1}, ..., h^m_{N,r}\}$
3: find the nearest neighbors of $mLSTM(B, S) = h^m_{N,*}$ in the HVS
4: sort them according to the cosine distance to $h^m_{N,*}$ (in descending order)
5: **return** l^+

Overall, it can be seen as a search algorithm which finds a best semantic replacement subtext for a text pair out of a pool related texts. We will theoretically extend and apply these new findings in the next chapter Application & Business Use Cases. To the knowledge of the author there is no similar method available[8].

6.3.4 Proposal: Cosine-Distance based Analogical Reasoning Task

Ideas about semantic vector spaces have been around for a while. Kindermann et al. [16] have constructed semantic spaces over word-frequency vectors (related predecessor to word embeddings) by using SVM classifiers (rather neural networks

[7]Remark: we have made further experiments with Algebraic Operations in HVS. However, the difficulty has been finding the right examples, similar to the popular analogical reasoning task of w2v("king")-w2v("man")+w2v("women")=w2v("queen"). In our case, the problem lies in P-Q pair consisting of whole sentences rather than simple atomic words (i.e. concepts). Nevertheless, we have found out that overall the operation of addition/subtraction seems to move the given hidden vector around those with whom the operations are performed.

[8]We might also draw parallels to methods for text pair summarization. To some extent the mapping of text pairs to high dimensional hidden vectors is comparable to similar efforts made by sentence2vec or paragraph2vec approaches.

Algebraic Equation	**Answer**
Athens - Greece + Iraq	Baghdad
Paris - France + Italy	Rome
Tokyo - Japan + Spain	Madrid
Abuja - Nigeria + Australia	Canberra
Cairo - Egypt + England	London
Stockholm - Sweden + Pakistan	Islamabad
Tehran - Iran + Germany	Berlin

Table 6.8: Examples for some analogical reasoning tasks (from subtask *capital-common-countries*) for assessing a trained word2vec model.

like in this thesis) to find some threshold for determining the membership of entities to pre-defined categories. With these Classifier Induces Semantic Spaces (CISSs) they analyzed textual contributions made to a public e-discussion forum and categorize its content into four distinct classes ('giving information', 'making objection', 'asking a question' and 'giving a reply') for summarization purposes.

In this thesis, we recognized the benefits of treating words in semantic spaces within the field of neural networks and empirically demonstrated the advantage of the measure of cosine similarity on the HVS where a threshold around $[0.85, 0.9]$ was observed for the underlying data. We propose to apply and generalize these findings to the following evaluation method of a trained word2vec model: Mikolov et al. [21] have designed the approach of *analogical reasoning task* for evaluating the quality of a trained word2vec model. Table 6.8 gives some examples for such *algebraic equations* out of the *question-words* task from Google's project [21]. Each equation is successfully solved only if the model predicts the exact same answer as shown in the table. However, this approach can raise some complications. What if for Germany a trained model predicts *Bonn* instead of *Berlin*? This would *also* be a legitimate answer for Bonn being the former capital of (West) Germany. Depending on the underlying scope and variety of the text corpus, it might be possible that a trained model captures different senses of one word and is more complex than a perfect model for the question-words task. In any case, the basic semantics might be still captured by the model. However, with the former method of binary assessment the model is termed to be simply wrong for such equations. This can make the evaluation method inappropriate for practical purposes. For such cases a more sophisticated evaluation method has to be considered, which can deliver a numeric measure of success. To this end, based on the findings of previous sections we propose an analogical reasoning task using the measure of *cosine-distance*. Instead of simply performing a string comparison on the predicted word (i.e. binary measure), the method measures the "semantic" distance to the

target word vector (for every entry in the *question-words* task). If for a certain threshold k the answer (Berlin) is among the k nearest neighbors of the predicted word (Bonn), then we regard the equation as passed. Analogous to our approach about HVS the data-dependent threshold dividing the semantic spaces can be determined to further enhance the understanding of the cosine values and the next neighbors in the vector space.

We assume this to be a more detailed evaluation method for a trained embedding model as more variety is introduced to the evaluation which saves the method to fail from measuring such complex behavior as described above. We leave an exact proof-of-concept for future works. However, it should be reported here that these new insights are already applied at the Fraunhofer Institute IAIS for assessing the quality of word embeddings within customer projects.

Chapter 7

Application & Business Use Cases

PricewaterhouseCoopers[1] (PwC) is a professional services network focusing on the services of audit, assurance, tax and consulting. Although numbers, images and audio/video records play an important role in its daily business, its primary medium of work and communication is based on text. Our text-based methods ideally fit into the profile of this company and have a potential of accelerating its digitalization agenda. The following use cases are created while keeping the landscape of legal and consulting companies like PwC in mind allowing a possible application therein.

In section 6.1 it was empirically demonstrated that apart from NLI the mLSTM model is able to solve various classification tasks like paraphrase detection. With this functionality, a *paraphrase detection application* can be constructed (see RQ-4) by simply following the procedure described in that same section. Once the model is successfully trained, a system is built which can be fed with an input of two arbitrary sentences and determine whether they are paraphrases of each other. Overall, we can apply such an application to many fields of science and business: like detecting paraphrases in Social Media for finding related popular opinions about some stock market development [2]. Specifically, it can be thought of a use case where the task is to extract all potential paraphrases from a certain document, to for example reduce the size of the written text and summarize its key points, or for any other algorithmic reason. It should be noted that the improvement of the application performance can depend on the availability of further domain-specific training data.

Next, we consider the findings of section 6.3. With the defined functionality of *best-replacement* some possible use cases are created in this chapter for an application

[1] http://www.pwc.de

in the landscape of business and legal occupations. We develop the following realizations. The first use case has been implemented in section 6.3 with the class of $cosine \in [0.9, 0.95]$. The remaining two use cases are theoretically build upon the first one and should be seen as a sketch.

7.1 Use Case 1: finding alternative sources for answering the same question

The first use case is created for the general task repeatedly occurring in real-life. In many text processing scenarios, one has the requirement to find further sources for answering a certain question. Possibly one source with an answer is available, but many other potential answers are hidden in a heap of unstructured documents which might be even better in their quality. In such a scenario, our HVS search can become very useful.

Case: Given a pair of (P^*,Q^*) where P^* can answer Q^* and an alternative list of passages $\{p_1, ..., p_r\}$, find the best passage p_i which can also positively answer the same, fixed question Q^*.

In order to apply the *best-replacement* functionality, the following adjustments have to be made. The necessary steps are described in the Algorithm 1 on page 53 with input (Q^*, P^*) and alternative list $l = \{p_1, ..., p_r\}$. After its execution, it delivers a sorted list l^+ which shows the best-replacement of P^* in accordance to the semantic relevance (in descending order). A human eye can look onto the list and make the final choice. This is much easer then going through all the passages for a large collection of documents. This application can be used as a pre-step towards QA, if nothing else can be applied or not enough labeled training data are available.

7.2 Use Case 2: detecting common opinions among online customer reviews

Products which are sold on online platforms like *Amazon*, *eBay* contain a textual description of the product and a list of reviews and comments made by users about the product. These comments are mostly ordered after the date and do not reflect any semantic arranging. Often it is interesting to find some specific opinion regarding a certain product to group such comments together for further assessments. In such scenarios our HVS search becomes useful. With the function-

ality of best-replacement, we create the following use case. The text of product description is treated as the base text B and all comments as supporting text s_i.

Case: Given a product description with a specific comment on it (B^*,S^*) and an alternative list of comments $\{s_1, ..., s_r\}$, find those comments s_i which share the same opinion of S^*.

Technically, this will be solved in the same way as the previous use case with Algorithm 1. The input is the pair of product description and comment (B^*, S^*) and the list of all comments $l = \{s_1, ..., s_r\}$. After its execution, the sorted list l^+ is returned which shows the nearest semantic neighbors of S^* in accordance to the opinions they share. Again, a human eye can look onto the list and make the final choice. Further sentiment detection can enhance the quality of the results. While being in the quest of buying products, this functionality allows the customer to get a better picture of the product and overall improves his decision making process.

7.3 Use Case 3: clustering similar paragraphs from legal codes

Legal codes usually have a strict writing format. The specific structure depends on the legal system (Roman civil law, Common law, Muslim law, Jewish law etc.) and the country where it is codified, but mostly they are made of basic *articles* which are composed of further legal additions and specifications in the form of *paragraphs*. Cross references within the code play an important role and are heavily used. For example, it is often the case that an article's paragraph shares its content with other paragraphs placed in different articles. This and similar forms of semantic referencing increases the complexity of the written code making law difficult to understand for layperson as well as for experts.

With our best-replacement algorithm, we propose an automatic clustering method to facilitate the process of finding semantic links and overall patterns in law codes. We focus on the example given above, but one can imagine any other type of scenario where text pairs can be constructed within the legal code.

Case: Given a pair of article and its paragraph (A^*,P^*) and a list of all other paragraphs $\{p_1, ..., p_r\}$ out of the legal code, find the best paragraphs p_i which are most similar (and closest semantic matches) to the fixed P^*.

Again, this will be solved in the same way as the previous use cases with Algorithm 1. The input is the pair of article and its paragraph (A^*, P^*) and the list

of all paragraphs $l = \{p_1, ..., p_r\}$. After its execution, the sorted list l^+ gives those paragraphs which are most similar to the input pair in its original textual context. This functionality allows the reader to get a better picture of the legal code and improves his overall performance while searching for similar paragraphs from various other pages within a large code.

Chapter 8

Discussion & Conclusion

In the beginning, due to the exploratory nature of our approach, the research topic was not precisely defined, and it was not known where to go and which direction to take. The ambitions where to cross existing frontiers and discover new insights within the class of deep neural memory networks in the field of QA, but the specific frontiers were not in sight. However, we say that this is truly a right way to start with, as this very scenario characterizes the basic principles of research where there is an endless striving for an ideal balance between exploration and exploitation. Originally, we started with the MemN2N architecture. But we realized soon that this direction is not prolific and requires more time-consuming investigations. Due to the limited time scope of the master thesis no deeper experiments for MemN2N could be conducted, hence the focus was shifted to the model of mLSTM. There, the success rate was relatively fast with the support of the large-scale dataset of SQuAD, resulting into new insights applicable for practical usage. Table 8.1 provides the answers to the major research question of the thesis.

Many challenges and difficulties were faced on this way. Understanding the deep learning topics and creating an efficient environment for conducting experiments were among the first major challenges. Creating a bridge between the theoretical research findings and applied business use cases and applications was a thought-provoking process. Most demanding, however, was the issue of time. Finding manually patterns in the large dataset of (P, Q) pairs and its hidden representations is just an example for a time-consuming task, which required a lot of concentration and patience.

Technical Challenges Although designing and applying neural networks are pretty technical tasks, being closer to software engineering than to the typical R/MATLAB-based data analysis, the tools available till late 2016 lacked the dynamic support which is typical for modern software engineering languages like Java with its various frameworks in the environment of Eclipse. When the thesis was

Question	Findings & Answer
RQ-1	- single training (SQuAD) has test accuracy **71%** - increasing training set size is a decisive step - parameter optimization (SQuAD) improves to **77.39%** acc. - joint training (SQuAD & SNLI) has highest achieved **84.18%** acc. *The performance is excellent with a high accuracy!*
RQ-2	- SQuAD task is transformed to most simple scenario of 1-word QA (cloze style) - MemN2N has high train acc. but very low test acc. **5%** - MemN2N is overfitting to SQuAD *The performance is negative with a very low accuracy. MemN2N fails to solve the QA task on SQuAD!*
RQ-3	- transfer learning allows to use pre-trained models for related tasks to boost the performance - extension of paraphrase detection possible with **70.57%** acc. *A Paraphrase Detection application can be developed with a close to the state-of-the-art accuracy due to the advantage of transfer learning.*
RQ-4	- hidden vectors have summarization property for texts Cosine similarity allows semantic text pair search - with the HVS a semantic-syntactic continuum is created - use cases for the domains of QA, Customer Review and Legal *Cosine similarity was shown to be highly useful for analyzing the hidden vectors of mLSTM. Many use cases can be designed on it, as shown in the previous chapter.*

Table 8.1: Summary of experimental results and answers to research questions.

started, the deep learning frameworks were in their initial state of release being relatively rigid in their structure and having till late 2016 unsatisfactory debugging features and hardly understandable error messages.

Besides, the developing and usage of neural networks is overall a pretty time-consuming matter, irrespective of the underlying framework. Every round of "test" required at least 1-6 hours depending on the size of data and the design of the network. If one wants to do fine tuning on the final set of design and training data, which is usually very large, one round can take up to 1 day! This setup is not ideal for fast parameter tuning and the overall development. However, thanks to the powerful HPC cluster at Fraunhofer IAIS, the development could be exceedingly accelerated by performing many parallel executions on different *GNU Screen* sessions across all available compute resources.

8.1 Scientific Contributions

This work has contributed to two major fields of computer science, specifically to machine learning with neural networks and computational linguistics in the domain of QA. Practically, it has shown that the chosen neural architecture of MemNN is not appropriate for realistic natural language QA tasks. Furthermore, it has extended the mLSTM model for NLI to one which can solve many tasks simultaneously like binary Passage-QA or Paraphrase Detection, and thereby producing summarization of text pairs which can be further utilized for semantic searches. Theoretically, it has shown the benefits of the measure of cosine-similarity in high dimensional vector spaces of neural networks, and building on it resulted in the proposal for a cosine-distance based analogical reasoning task for evaluating customized word embeddings. Apart from solving the RQs (see Table 8.1 for more details), some further insights where gained during the course of the thesis. Specifically, the contributions are:

1. Verifying MemNN performance on domain-independent (*businesslike*) QA text data of SQuAD—this study is the first to the knowledge of the author which empirically solves this question raised by [34]

2. Extending NLI task to binary Passage-QA with Match-LSTM

3. Extending NLI task to Paraphrase Detection with Match-LSTM

4. Showing effects of joint training and transfer learning for neural networks (NLI $\rightarrow$ binary Passage-QA & Paraphrase Detection)

5. Introducing a novel text pair abstraction method applicable for semantic search on HVS and creating the *best-replacement* algorithm on it

6. Demonstrating the advantage of cosine similarity on HVS with a threshold around $[0.8, 0.9]$ and proposing the cosine-distance based analogical reasoning task as its generalization

7. Creating business use cases for legal and consulting firms

We have created a thesis which encompasses many branches of science. From theoretical foundations through experimental applications till business use cases (aka. in business terminology: *from strategy till execution*). By applying insights from the hidden representations of neural networks back to the foundations of word embeddings we close the circle of the thesis. Overall, the work has taken part in the advancement of artificial intelligence to human-like performance and possibly even reached new frontiers in this field of science.

8.2 Future Research

The scope of future works is wide. The findings of this thesis raise numerous new questions and pave the way for further research. Various models and techniques applied in the thesis can be theoretically extended or practically examined with further datasets.

For the model of mLSTM additional text pair classification task can be examined like *news categorization* and *article classification* (see page 42). To achieve this, the appropriate training data has to be gathered first, and next converted to the format similar to binary Passage-QA task, as it was done with the dataset of SQuAD in this thesis. Furthermore, a knowledge base could be integrated to the QA model to enrich the domain knowledge (available in the passage) with the component of general knowledge to improve the quality of answers and the performance of the automatic QA system. Overall, with such a QA system, we propose a straightforward deep learning show case for a robot/chatbot which can be applied to many real-world demands for automatic customer services. By installing the pre-trained mLSTM model for QA into an empty, unintelligent machine alongside a state-of-the-art speech recognition software, an object is brought to life, which can interact with humans and answer (binary) questions on passages, which are read aloud to them by humans.

For the model of MemN2N a deeper analysis can be performed with a lager training dataset. This might change the performance of MemN2N, as it was the case for mLSTM when applied to the setup of single training. However, such larger data should fulfill our simple scenario of subsection 6.2.1. Most probably such data will not be publicly available yet, as we performed an extensive search and apart from SQuAD no further large datasets could be found. Hence, we propose to create such data manually by annotating the shortest passages from Wikipedia articles

with simple what-questions, given that the resources for manual annotating are available.

For the Semantic Text Pair Search, the analysis of HVS is solely a deep topic. Experimenting with various further textual datasets to understand the nature of the semantic-syntactic continuum and its threshold, formalizing the mathematical properties of it, and similar analytical tasks are potential future works. Building on this, the topic of *cosine-distance based analogical reasoning task* should be considered as well. The very next future work is a scientific verification of the proposal on the question-words task, outside the scope of customer projects. Overall, it should be noted that this topic is for itself a major research direction, which might be better examined in a doctoral thesis due to its longer time frame. If time permits, we might be able to do this in a future project.

In this work, many doors have been opened up for further research. After an enduring time of striving and endeavor, the work has finally come to an end. Yet, this is nothing but just another starting point of a never ending chain of researches, seeking to discover the hidden truth of our world. This thesis is merely a drop of water in the ocean of knowledge—an ocean if turned into ink for the words of nature, sooner would the ocean be exhausted than would the words come to an end.

Appendix A

Study Resources for Machine Learning/Deep Learning Skills

As the topic of ML/DL is relatively new in academia and industry, most universities do not offer a comprehensive curriculum for acquiring the skills necessary for conducting research in this field. Consequently, the importance of autodidacticism grows. Students and professionals who are interested in replicating and building on this thesis can consider the following listing. It gives an overview of such interesting resources for studying ML/DL topics which arose from the personal experience made by the author while working on this thesis.

- *Deep Learning* lecture at Bonn-Aachen International Center for Information Technology (B-IT) (by Dr. R. Bardeli)
- *Deep Learning Seminar* at Fraunhofer IAIS[1] (by Dr. G. Paass and Dr. J. Keuper)
- *Data Science & Text Analytics Seminar* at Fraunhofer IAIS[2] (by Dr. J. Kindermann)
- *Machine Learning* lecture at Goethe-University Frankfurt (by Prof. V. Ramesh, Prof. M. Kaschube, Prof. Dr. N. Bertschinger) (Resource: *Pattern Recognition and Machine Learning* by C. Bishop)

Besides, various scientific papers mentioned in chapter Related Literature and the online tutorials of *Tensorflow* and *Theano* were essential for understanding the depth of this field. But for all that, the most important key to success was the on-site work at Fraunhofer IAIS with Dr. J. Kindermann and further colleagues.

[1] http://www.bigdata.fraunhofer.de/de/datascientist/seminare/deep_learning.html

[2] https://www.iais.fraunhofer.de/de/geschaeftsfelder/big-data-analytics/uebersicht/data-scientist-schulungen.html

Bibliography

[1] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Sajawel Ahmed. Contrasting wikipedia activity patterns with stock market activity. Bachelor thesis, Goethe University Frankfurt, 2013.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[5] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

[6] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[7] Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*. Asia Federation of Natural Language Processing, January 2005.

[8] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

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

[10] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[11] Donald O. Hebb. *The organization of behavior: A neuropsychological theory*. Wiley, 1949.

[12] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

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

[14] Mohit Iyyer, Jordan L Boyd-Graber, Leonardo Max Batista Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *EMNLP*, pages 633–644, 2014.

[15] Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*, 2016.

[16] Jörg Kindermann, Gerhard Paass, and Edda Leopold. Analysis of e-discussion using classifiers induced semantic spaces. *GLDV-Journal for Computational Linguistics and Language*, 22(1):21–27, 2007.

[17] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*, 2015.

[19] Pranava Swaroop Madhyastha, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Mapping unseen words to task-trained embedding spaces. *arXiv preprint arXiv:1510.02387*, 2015.

[20] Vinícius Gonçalves Maltarollo, Albérico Borges Ferreira da Silva, and Káthia Maria Honório. *Applications of artificial neural networks in chemical problems*. INTECH Open Access Publisher, 2013.

[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[22] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Hlt-naacl*, volume 13, pages 746–751, 2013.

[23] Nick Montfort. *Twisty Little Passages: an approach to interactive fiction*. Mit Press, 2005.

[24] Preslav Nakov, Lluis Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. Semeval-2016 task 3: Community question answering. *Proceedings of SemEval*, 16, 2016.

[25] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[26] Matthew Richardson, Christopher JC Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, volume 3, page 4, 2013.

[27] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[28] Shaohuai Shi, Qiang Wang, Pengfei Xu, and Xiaowen Chu. Benchmarking state-of-the-art deep learning software tools. *arXiv preprint arXiv:1608.07249*, 2016.

[29] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013.

[30] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[31] The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

[32] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

[33] Shuohang Wang and Jing Jiang. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*, 2015.

[34] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*, 2016.

[35] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

[36] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[37] Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic memory networks for visual and textual question answering. *arXiv*, 1603, 2016.

[38] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[39] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

Schriftliche Erklärung

Hiermit bestätige ich, dass ich die vorliegende Arbeit *Steps towards Question Answering with Deep Neural Memory Networks* (zu Deutsch etwa: *Schritte zur Automatischen Beantwortung von Fragen mit Tiefen Neuronalen Speicher-Netzwerken*) selbstständig verfasst habe und keine anderen Quellen oder Hilfsmittel als die in dieser Arbeit angegebenen verwendet habe.

Sankt Augustin, 31.03.2017