The University of Sussex
School of Engineering & Informatics

Master's Dissertation

Programme:	MSc in Evolutionary and Adaptive Systems
Candidate number:	152176
Title of Dissertation:	Applying Deep Learning to Question Answering
Approximate number of words:	12200
Supervisor(s):	Dr Bill Keller

Date submitted …….. Time submitted …….

Declaration

I certify that the information on this cover sheet is correct.
I certify that the content of this dissertation is my own work, and that my work contains no examples of misconduct such as plagiarism, collusion, or fabrication of results.

Candidate's signature …….

UNIVERSITY OF SUSSEX

MASTER'S THESIS

Applying Deep Learning to Question Answering

Author:
Indeera MUNASINGHE

Supervisor:
Dr. Bill KELLER

A thesis submitted in fulfilment of the requirements
for the degree of Master of Science in Evolutionary and Adaptive Systems

in the

School of Engineering and Informatics

August 29, 2017

Declaration of Authorship

I, Indeera MUNASINGHE, declare that this thesis titled, "Applying Deep Learning to Question Answering" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed:

Date:

"... Problems are often stated in vague terms, because it is quite uncertain what the problems really are ..."

John von Neumann

University of Sussex

Abstract

School of Engineering and Informatics

Master of Science in Evolutionary and Adaptive Systems

Applying Deep Learning to Question Answering

by Indeera MUNASINGHE

Asking questions is an innately human characteristic. Our knowledge is enriched to a great extent and distributed across large groups by the questions we ask and answers we get. As the base of knowledge is ever increasing and the computing resources are ubiquitous, we require an automated method of answering questions.

Deep learning is a relatively new area of connectionist approach to solving computational problems, that had shown great promise in various domains such as vision, speech as well as natural language processing.

In this project we explore the applicability of deep learning techniques to question answering by way of reranking question answer pairs using word embeddings and convolutional neural networks. We generated a large number of models with an evolutionary hyper parameter optimiser and found that higher dimensional word embeddings coupled with a deeper network achieve better performance without resorting to explicit feature engineering of any kind.

Acknowledgements

I would like to thank professor Bill Keller for giving me the opportunity to explore this very interesting and challenging project and his valuable guidance along the way. I also would like to thank Dr. Pawel Swietojanski, Dr. Zafeirios Fountas and Pedro Mediano for interesting discussions and valuable suggestions.

Contents

List of Figures

List of Tables

Dedicated to my daughter Lena who has an infinite stream of amusing questions about many things and to my parents who forever see me as a little boy who has a lot to learn …

Chapter 1

Introduction

It is human to ask questions about the nature of reality. It is how we build up a model of the world around us and come to understand it. As soon as we develop sufficient speech capability, we ask a lot of questions about how things around us operate and relate to one another.

The ability to answer questions related to a subject is the hallmark display of knowledge. Since the advent of computers or simulated automatons, have them answer questions on various topics had been a quest to scientists and a prominent tool for Sci-Fi writers. In modern Sci-Fi advanced technology is almost always depicted together with a computer system that can communicate and answer questions through voice and natural language. Combining the capability of computer systems to search and access a vast amount of information through a natural language interface is long sought after.

On-line web search is ubiquitous. People who are looking for answers to a particular question will type in a phrase or a keyword and they are presented with a list of links with brief summaries of what they could contain. Question Answering in NLP aims to go beyond this type of document retrieval to return exact answers to natural language questions. For example factoid questions such as "What's the tallest building in the world ?" can be answered with a short phrase. Most of these questions are of the variety *Who, What, Where, When*. These key question words can be used for refining the answer selection. For example, a question of type *Who* is looking for a person, *Where* a location, *When* a time or a date, but *What* or *Which* is more abstract. For example "What was the Beatles' first hit single ?" refers to a song. There are other types of questions, such as the ones seeking definitions, philosophical or rhetorical. To keep the focus narrow, in this thesis, we only consider the factoid type of questions.

Question Answering (QA) is important, but deceptively difficult. An open domain question answering system consists of a number of stages before exact answer can be extracted, for example, finding the relevant documents, finding the relevant paragraph(s) in those documents and then ranking of the potential answers depending on their suitability to the question asked.

Various attempts at engineering QA systems such as knowledge bases quickly become out of date as they are costly and difficult to maintain with ever increasing information content. Hence scalable methods that can learn to answer questions from the data itself is of great interest.

In information processing, deep learning (DL) with deep neural networks (DNN) are becoming popular as they can scale well with the increasing amount of data available. Convolutional neural networks (ConvNets) is one such method that has gained wide popularity among machine learning community because of its versatility and excellent performance in wide variety of domains.

Recently, distributed representation using word embeddings have produced remarkable results surpassing the previous attempts at semantic modelling. Combining all of this, we attempt to train a ConvNet using word embeddings to a popular dataset in QA, namely TREC8-12 & TREC8-13 in a question answer pair reranking task.

Chapter 2

Background

2.1 Question Answering

Question answering is the computational process of automatically answering a question posed in natural language to a computing entity. Ability to answer questions posed by humans is of great importance as it could enrich many people's lives. Answering questions manually using human effort is costly and would not scale well.

Question answering can be broad, as the questions could be abstract, philosophical, factual etc. Even though answering abstract and philosophical questions are even hard for human counterpart, factual question answering has a lot of value. In this domain our goal is to address questions such as "What's the capital city of United Kingdom?". In the following sections we look at the various other methods of question answering attempts and leading to why our method could be better suited.

2.1.1 Information Retrieval based QA

IBM Watson was a system that surpassed human ability to accurately answer questions (although in Jeopardy television quiz show its job was to formulate the questions given a brief clue). "This required it to answer open domain questions as well as assess its performance to each question or category. The system was required to deliver a high degree of precision and confidence over a broad range of knowledge and natural language content within a 3-second window" [1]. DeepQA, the system behind IBM Watson, was a massively parallel probabilistic evidence-based architecture, which had access to a wide range of encyclopaedias, dictionaries, news-wire articles and literary works both structured as well as unstructured [3]. Access to a large amount of knowledge is critical for open domain QA.

In the process of answering a question, information retrieval (IR) based system such as the DeepQA system would search for a potential answer containing documents and then perform deeper content analysis such as named entity recognition NER and coreference resolution. NER is the process of extracting the entities such as place names, person names, addresses, dates etc. from the text. This helps in answering

question types *Who, Where, When*. Coreference resolution is finding the relationship between a pronoun and a noun in the text. For example, a passage containing "First seven attempts to summit Mount Everest failed. Edmund Hillary and Tensen climbed it in 1953.. ", The *it* in the second sentence refers to *Mount Everest* in the first sentence. Therefore in order to successfully answer questions such as above, these references have to be resolved.

This two stage process can be subdivided into a four stage process where document retrieval is preceded by a question analysis stage and succeeded by a passage retrieval stage [5]. Hence, a successful QA system for real world utilisation, must have a several stages of analysis and processing.

2.1.2 Automatic query augmentation

Attempts have been made to improve the document retrieval by applying a stemming algorithm at document indexing time and performing morphological query expansion at retrieval time [4].

Bilotti et al. discovered that using Porter stemmer [6] reduces the recall and morphological query expansion increases the recall. Stemming is the process of reducing a word to its base form [4]. For example, the words *consign*, *consigned*, *consigning*, *consignment* would have the base form *consign*. The Porter stemmer uses an orthographic algorithm to alter the words without any semantic understanding. This causes semantically unrelated words to be conflated leading to poor recall performance. Harman D. also showed that stemming does not improve document retrieval, as the number of queries that were improved by stemming tended to equal to the number of queries that were negatively affected [10]. However, these effects of stemming had previously been contradicted by other research [7] [8] [9] to be beneficial.

Query expansion extends the question by adding new words that are not included in the query to improve the document retrieval. Morphological query expansion is a form of inverse lemmatisation, where multiple forms of a word, are added in addition to the query word. For example, *sanction* can be expanded with *sanctioning, sanctioned, sanctions, sanctionable*. Bilottie et al. used an inflectional expansion, for example, a query like "What lays blue eggs?" can be expanded with a structure as follows: "what (lays, laying, lay, laid) blue (egg, eggs) ?". Additionally, weights can be assigned to each of the expanded words [4]. Another method of expanding a query would be to augment with a list of synonyms. This process improves the recall as it increases the likelihood of matching with an expanded word in the document set.

The natural language is complex, and words can have the same form and different meanings (polysemy). For example, *bat*: *animal, sports equipment*. Likewise, two

morphologically different words may agree semantically; for example, *bicycle* and *bike*. Therefore trying to match exact terms of the query and the potential documents is challenging.

2.1.3 Exploiting mutual information

Berger et al. proposed to use Mutual Information based method to *bridge the lexical chasm* between question and answers. This process involves training a model using a machine learning technique. Therefore it requires a dataset of questions and the correct answer as the training set. The mutual information is calculated between the query terms and proposed answer terms. The intuition behind this approach is that a query containing a specific word incite responses with another set of words and there's a pattern to it. For example, a query containing a *Why* would most likely to be answered with a sentence containing *because*. A query about *vacation* is likely to be answered with words such as *flight, booking, hotel, trip* [11]. These associations can be learned by a machine learning model which could then be used to augment a query with the related set of words that in turn improve the recall of document retrieval. In order to train a model, Berger et al. used a Usenet FAQ and a call centre dialogue dataset. These datasets covered a broad range as well as contained real world examples of how people ask questions and what they regard answers to be. Thus the model training was to identify the patterns between the question and the answers from the data itself.

2.1.4 Term Frequency . Inverse Document Frequency (TF.IDF)

TF.IDF is a widely used tool in information retrieval to represent documents as vectors of real numbers, rather than the words themselves.

Each document can be represented by the set of words it contains. In numerical form, this could be a vector representing frequencies of each of the word in that document.

$$D = \langle tf_1, tf_2, ...tf_n \rangle$$

i.e term frequencies as the vector elements.

But some words occur frequently without representing much semantically. For example, the stop words like *the, is, are* occur very frequently in every document. These words, therefore, don't characterise the document. For example, if a document is about cars, we would like to have the word *car* and the related words to have a high prominence in a vector representing the document. Thus, document representation

can be improved by scaling down these commonly occurring words. IDF is a measure that computes the number of documents a term appears in a set of documents. This is computed as follows:

$$IDF = log\frac{N}{(1+Dc)}$$

Where N is the number of documents, and Dc is the number of documents, the word appeared in.

TF.IDF is, therefore, the product of these two terms, gives a measure of how important a term is (discriminant) in representing the document. Now a document can be represented by a vector of TF.IDF measures.

$$D = \langle TF.IDF_1, TF.IDF_2 ... TF.IDF_n \rangle$$

In order for document comparison, a common vocabulary must be chosen. Document similarity can be computed by vector dot product where documents that are similar would be closer together in a high dimensional vector space.

Being a statistical measure, it does not how ever try to assert semantic similarity directly. For example a response to a question "Where is a good place to have dinner?" could be "Sticky Joe has excellent steak and great atmosphere", but these two documents have no terms in common.

TF.IDF is one of the first methods of projecting documents into a vector space and using linear algebra for processing a large number of documents with ease. A similar but much more representational method called word vectors is described later in the section.

2.1.5 Knowledge based QA

A knowledge base can be a structured database such as DBPedia, Freebase or a program written in Prolog. Prolog is a logic programming language created in 1972 for natural language processing in French with predicate logic [12]. In this approach, a semantic representation of the query is made, and then the answers are retrieved from a knowledge database. This semantic representation is performed by applying formal logic to the query. Formal logic or predicate calculus deals with reasoning with symbols, and is a branch of discrete mathematics. For example, from the following two sentences:

1. All birds have wings
2. Dodo is a bird

one can deduce that Dodo has wings. In order to form the argument, *terms* like Dodo has to be identified as well as their properties and predicates. This is the process of predicate calculus. In the above example a predicate is the *have* relationship in first sentence or *is a* relationship in the second sentence. Nouns and pronouns in English language has a similar role as *terms* in predicate calculus.

In contrast to IR based methods described above, where answers are sought through matching lexical properties, knowledge based systems try to model the semantic relationships in the query. The knowledge based question answering was partially used in DeepQA system mentioned above. But in practice creating and maintaining a knowledge base is difficult and expensive [2]. For example DBPedia is a crowd sourced knowledge graph where information is extracted from Wikipedia into a structured format [13]. This is then queried using a special query language called SPARQL [14]. It represents close to 4.5 million entities [1] in English but keeping up with constantly changing and update information is a challenge. Translating the question into a semantic representation has its own challenges such as lacking the sufficient evidence to detect predicates as well as some questions not adequately representable in predicate logic [15].

2.1.6 Text Retrieval Conference (TREC)

TREC provides a common platform for evaluation of text retrieval methods which has become a standard in the information retrieval community [2]. It provides a number of datasets in QA track that are used by researchers to compare their methods. The ability to compare the methods and results via a common performance measure is vital in improving the QA algorithms. TREC8-12 and TREC8-13 datasets are widely used for factoid QA tasks. The properties of this dataset is presented in Methodology section.

[1] http://wiki.dbpedia.org/about
[2] http://trec.nist.gov/overview.html

2.1.7 Evaluation methods

Normal precision-recall measures are less useful in evaluating question answering systems as they return true positives as well as false positives without any measure for ranking of the answers.

Mean Average Precision

Mean Average Precision (MAP) is the mean of the average precision of the ranked correct answers where average precision is the precision at a given rank. Average precision is *only* computed for the correct answers.

$$AveragePrecision = \frac{1}{n}\sum_{r=1}^{n} Prec_r$$

$$Prec_r = \frac{hits_r}{r}$$

$$MAP = \sum_{q=1}^{m} AvgeragePrecision_q$$

$Prec_r$ represents the precision at rank r for correct answers. $hits_r$ represents the number of correct answers so far at the current rank r. The n is the number of answers for a given question, and m is the number of questions in the dataset.

Mean Reciprocal Rank

Mean Reciprocal Ranking (MRR) is the mean of the reciprocal ranking of the first correct answer. This measure only looks at one correct answer, therefore better suited for questions with only one correct answer. In most research papers both MRR and MAP are presented for better comparison.

$$ReciprocalRanking = \frac{1}{r}$$

$$MRR = \sum_{q=1}^{m} ReciprocalRanking_q$$

Where r is the rank of the first correct answer and m is the number of questions in the dataset

2.2 Deep learning

Our method relies on the techniques of deep learning and we describe the relevant related information in detail below.

2.2.1 Brief history

Neural networks in machine learning has a long history. McCulloch & Pitts in the 1940s devised a technique called *thresholded logical unit* which mimicked how a neuron was thought to work at the time. Frank Rosenblatt et al. [16] improved upon that idea to come up with the *perceptron*. This is the basis of the computing unit of the current neural networks. Perceptron is an algorithm for learning a binary classifier. Its function is to map a real valued input to a boolean value {True, False}. Perceptron consists of a single layer of computational units and is only capable of learning a linear decision boundary.

Multi Layer Perceptron (MLP) is a network which consists of at least three layers of nodes each with a non-linear activation function as opposed to the linear activation function of a perceptron previously described. Non-linear activation used by the computational nodes was loosely modelled on a biological neuron's action potential and commonly represented by sigmoids such as a hyperbolic tangent or a logistic function. Learning the non-linear decision boundary was done by propagating an input signal through the network and comparing the output to the desired target output and propagating back the error such that the weights are updated to minimise error [17]. This method commonly known as *backpropagation* using *gradient descent* is the backbone of training any neural network today. MLPs are also known as multilayer feed-forward networks and are universal function approximators [18]. Given enough data and a sufficient number of hidden units, multi layer networks can be trained to learn any function to any desired degree of accuracy.

Multilayer feed-forward networks laid the groundwork for deep neural networks and deep learning. More layers and nodes were required to learn an increasingly complex function but this also made it difficult to effectively train the networks. What is currently known as deep learning started with the introduction of *Deep Belief Networks* (DBN) by Hinton et al. in 2006 [19]. DBN consisted of an unsupervised network called a *Restricted Boltzmann Machine* (RBM) [20]. RBMs are capable of a variety of functions and among them notably are, dimensionality reduction [21] and feature learning [22]. Due to the fast learning algorithm invented by Hinton et al., RBMs could be used to initialise a deep network by pre-learning features.

Motivations for deep architectures are stemming from the world around us. Brains are deeply layered. Our vision system has a hierarchy of processing stages from

sensing the light and colours to detecting edges and contrast to primitive shape detectors to a higher level compositions such as the perceived world around us. Our ideas are mostly organised hierarchically by composing of simpler ideas and we learn simpler tasks which we combine to perform increasingly sophisticated actions. Therefore it naturally follows that deep architecture that is composed of layers that can learn a series of intermediate representations, which increase in complexity, can be successful in learning higher level concepts such as faces and natural language [23].

Deep networks are a great success in a wide range of domains. Most notably in vision tasks, they beat conventional methods such as *support vector machines* (SVM). They also can handle large datasets where SVMs fails to scale. Deep architectures display distributed representations which are shown to be more efficient than local representations. A concept is represented by many connections (distributed representation) and the connections are shared as opposed to a local representation where each concept would bind to a single or small number of connections.

2.2.2 Non linear activation

The non-linear activation is essential for the network to learn an arbitrary function. There are variety of activation functions such as the logistic function $f(x) = \frac{1}{1+e^{-x}}$, hyperbolic tangent $f(x) = tanh(x)$, and ReLu (Rectified Linear Unit) $f(x) = max(0, x)$. Logistic function was historically used as it vaguely modelled the biological neuronal activation and also was differentiable. But it has major drawback when used in larger networks, due to saturation at 0 or 1 leading to non learning networks. Another drawback is that it is not zero centred, resulting in oscillations while gradient updates. Hyperbolic tangent is an improvement on the logistic function, where the output is zero centred, thus shown to perform better because of its symmetry [39]. ReLu is a computationally efficient approximation which was shown to achieve faster convergence [38]. Currently most networks use ReLu or one of its variants due to it high performance vs accuracy ratio.

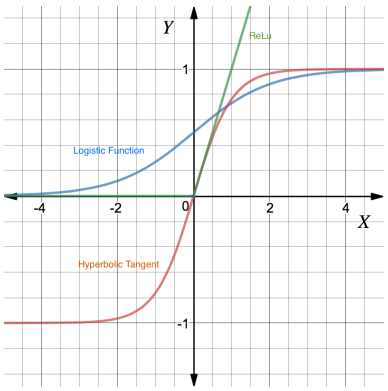


FIGURE 2.1: Activation Functions

2.2.3 Backpropagation & optimizers

Neural networks learn the parameters of the network from the data, given an objective function. This is a supervised method, where a target label is present [3]. Training dataset is fully passed through the network (an epoch) and the mean squared error is computed [46]. This error is propagated back through gradient descent and the network parameters are updated by computing the partial derivatives of error with respect to the network parameters. This process requires a lot of computing resources when the dataset is large and slow to train as network updates only after the whole dataset is passed through.

Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent is an improvement on the standard gradient descent, where errors are back-propagated and parameters are updated for a small batch of samples (mini batch). This is an approximation to the true error gradient, but has been shown to converge to the global error minimum if the error function is convex [32]. However this requires the random shuffling of the training samples as the technique can depend on the order of the samples presented.

These basic methods of optimisation has many variations to improve various shortcomings [37]. Here we describe a small number of them.

Adagrad

Adagrad adapts the learning rate depending on the frequency of updates to the parameters [33]. Infrequently visited parameters get a larger update than the frequent ones. Therefore it is well suited for sparse datasets. It is found to improve the robustness of SGD [35] and was used in training the GloVe word embeddings [34].

Root Mean Square Propagation (RMSProp)

With RMSProp, learning rate is set for each parameter and adapted based on the recent gradient magnitudes with an exponential decay. This method is an unpublished proposal by Geoffrey Hinton, to resolve the fast diminishing learning rate problem of Adagrad.

[3] There are unsupervised initialisation methods such as Autoencoders

Adam - Adaptive Moment Estimation

This method is a combination of advantages of the above two methods and similarly utilises an adaptive learning rate and adopted widely in natural language processing. The properties of this optimisation method are that it is computationally and memory wise efficient, well suited for large datasets and networks, and requires little tuning [36].

2.2.4 Over-fitting

Deep networks have a lot of parameters. The reason for a lot of parameters is to effectively learn a model from a very large dataset. These terms 'lot of parameters' and 'large dataset' are arbitrary in the sense there's no rule to say that a specific number of parameters must be chosen for another specific number of samples. As the model we seek is unknown, so is its complexity. A balance must be struck between these two opposing variables where the model generalises enough so that it is not just remembering the samples that were seen. When a model just remembers the samples it has seen, it would perfectly (or very closely) fit every one of those samples. The goal of the model building is to learn from the data and be able to predict effectively for an unseen sample. Over-fitting can be observed while training when the training set accuracy improves while validation set accuracy lags far behind or worsens. It could also be seen when the validation set *loss* increases along with the number of epochs.

2.2.5 How to combat over-fitting

There are few techniques to combat over fitting in Deep Learning. Increasing the dataset with representative samples can reduce over-fitting.

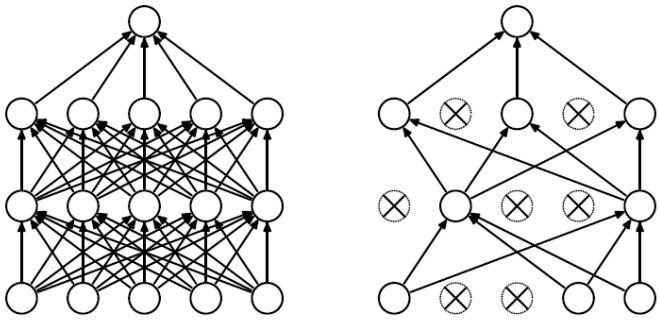
Increasing the dataset can be done by either collecting more data (expensive and time consuming and may not always be practical) or by augmenting the dataset with variations [4].

Using a smaller network with fewer parameters reduces the model complexity and reduces the likelihood of over-fitting. Standard regularisation methods such as L1 and L2 [5] can be used but currently more popular and simpler method called Dropout [42] is used where specified random number of neuron's activations are set to zero effectively making them inactive (Figure 2.2).

[4] Easier for images where an image can be rotated, cropped, varied contrast/brightness etc..

[5] Described in detail in $http://www.machinelearning.org/proceedings/icml2004/papers/354.pdf$

Early stopping is one of the easiest methods. It monitors the performance of the model on the validation set at each epoch and heuristically terminates training if the loss increases or the training accuracy doesn't improve.



(a) Standard Neural Net (b) After applying dropout.

FIGURE 2.2: Dropout Neural Net Model. Left: A standard neural net with 2 hidden layers. Right: An example of a thinned net produced by applying dropout to the network on the left.Crossed units have been dropped..
Source: Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting."[42]

2.3 Convolutional Neural Networks

Convolutional Neural Networks (ConvNets) are a type of deep networks that were inspired by the visual cortex of animals [24][25]. The animal visual cortex contains receptors that respond to a small part of the visual field and are hierarchically connected to neurons that process more and more abstract types of features. Deep networks although able to process a large number of inputs, high dimensional inputs such as images makes them impractical. ConvNets reduce this dimensionality problem by extracting features through a non-linearity and sub-sampling. The nature of the regular deep networks is that it ignores the structure of the input. The convolutional stages, on the other hand, are very good at responding to the structure of the input where discriminating features reside.

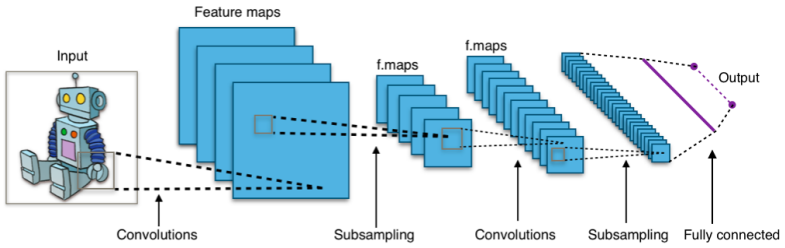


FIGURE 2.3: Typical Convolutional Neural Network.
Source: https://commons.wikimedia.org [41]

High dimensional domains like visual signals are highly correlated spatially, audio signals are correlated temporally and to a certain extent, natural language text has local correlations as well as long range correlations. Convolutions over the input signal integrate local features which are input to the next layer, which is then convolved again for several numbers of stages.

Convolutions are effectively filtering kernels that operate on a spatial or temporal dimension. Therefore a single filter extracts one type of a feature. A number of kernels operate on the input signal to extract different types of features. This enables overlapping feature extraction from the input signal. Collectively these convolutions produce a number of feature maps. These feature maps are then sub-sampled before being fed into the next layer. This operation reduces the resolution of the input signal as well as makes it less sensitive to small distortions and shifts. The alternating nature of convolutions and sub-sampling makes the network operate on increasingly specialised features. For example for a visual input signal, the first stage of convolution can generate a set of feature maps representing edges of various orientation. Second stage convolutions operate on these feature maps after sub-sampling and can combine these features to learn a shape of a certain orientation (Figure 2.3). This recursive processing results in features being extracted regardless of the scale and orientation of the real world object they belong to.

Supervised training of the ConvNets makes it possible to learn the convolutional feature extractors automatically. This is in contrast to the traditional methods where the feature extracting kernels had to be hand designed. After the convolutional stages, the resulting signal is fed into a fully connected network with one or more hidden layers. The classification or regression is learnt at these stages.

Since their invention in the 90s ConvNets has been adopted heavily in visual, speech and natural language tasks, largely dues to the availability of a large amount of data as well as computational power by way of commodity graphical processing units (GPU). LeNet is a ConvNet developed by Yann LeCunn in 1998 [26] which was used to read zip codes and digits. In 2012 Alex Krizhevsky et al. created AlexNet which beat the ImageNet challenge by a large margin [38]. ImageNet is an image classification and object detection challenge at large scale (1000 categories 1.2 million images) [6]. In 2014 GoogLeNet was created by Szegedy et al. which won the ImageNet challenge. This was an improvement on AlexNet by reducing the number of network parameters from 60M to 4M [27].

2.3.1 ConvNets in Natural Language Processing

ConvNets had been successfully used in various NLP tasks and some of them are briefly discussed below.

ConvNets were successfully used for classification tasks in other domains such as vision using a raw signal, Zhang et al. (2015) used characters as the raw signal for text classification [28]. Each character was one hot encoded with regard to an assumed alphabet, and 1 dimensional convolutions applied. The networks consisted of 9 layers with 6 convolutional layers and 3 fully connected layers. They found that ConvNets are an effective method for classification tasks.

Kalchbrenner et al. (2014) used a ConvNet for sentence modelling. Their network was able to handle input sentences of varying length and was able to capture short and long rang relations. It was not relying on a parse tree and was applicable to any language. They were very successful in binary classification, multi-class segment prediction, six-way question classification to name a few [29].

Severyn et al. (2015) created a method for ranking question answer pairs of TREC dataset using a ConvNet which beat the state of the art then by 3% in MAP and MRR scores [43]. Taking words as input and using very little external features, this model closely resembles the network model described in this report.

[6] http://image-net.org/challenges/LSVRC/2012/index

2.4 Distributional semantics

Meaning of a word is contextual (Firth 57). Words could have different meanings depending on the context they are used. Also words which are similar in meaning occur in similar contexts [47]. The distributional similarly and meaning similarity is correlated such that former could be used to estimate the latter. Distributional Semantic Models (DSMs) denote the context by counting the frequency of co-occurring words and having those represented as vectors. DSMs therefore can be used to measure the relevance of candidate answers to a given question by exploiting the paradigmatic relations between words [60].

Many attempts have been made to model this property of a language, such as Latent Semantic Analysis, Latent Dirichlet Allocation and Neural Language Models. We describe them in the following sections.

2.4.1 Latent Semantic Analysis (LSA)

Statistical techniques operate on the lexical properties of what is being said, but the intention is to extract what is actually meant or referred to. This challenge is largely two fold: A word may be *polysemous* where it can have different meanings in different contexts. A word may also have synonyms that are morphologically different and may also depend on the context. Latent Semantic Analysis is a technique that attempts to address these two challenges.

A large collection of documents are considered for LSA and each document is represented as a vector of word frequencies or TF.IDF. A matrix is formed with these vectors transposed such that the documents are represented as columns. This matrix is decomposed using linear algebraic technique called Singular Value Decomposition [50] to a lower dimensional space known as Latent Semantic Space which represents semantic relationships [49].

LSA uses a truncated SVD where it only keeps k largest singular values and their associated vectors. The original high dimensional vector would be sparse as various terms in the overall vocabulary may not occur in all documents. But the lower dimensional space is likely to be not sparse and as such meaningful associations could be derived between two documents even if they don't have any terms in common. It is assumed that the terms having similar meaning are loosely mapped to the same direction in the latent space [49]. With these resultant term and document vectors, similarity among them can be computed using cosine similarity or distance metrics such as Euclidean or Manhattan.

LSA is good at capturing synonyms and reduces the noise by decomposition process, also provides high recall and is language independent. But the drawbacks are that this process is computationally expensive as well as subject to high variance for

small changes in the original matrix [51]. Interpreting the resultant semantic space has known to be difficult or incomplete. Also as a term is represented by a point in space, polysemy is largely not captured. Also this model assumes a Gaussian distribution of terms where term occurrences are unlikely to be normally distributed.

2.4.2 Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation is a probabilistic generative model for discovering latent semantic topics in a large corpus of text [64]. It is quite similarly to probabilistic version of the LSA described above. LDA models documents as a distribution of topics, and topics as a distribution of words. For example a document can assumed to have n topics $(T_1, T_2...T_n)$. Each topic has an associated probability of appearing that sum up to 1. Taking the topic T_1, it can generate a random distribution of words $(W_1, W_2..W_m)$, each with a probability that adds up to one for that topic. Taking words as the observations, topics can be discovered with this technique giving applications such as document classification, summarisation and similarity and relevance judgement, all of which are applicable in QA context. The limitations of LDA are that, the number of topics to discover must be known ahead of time, requiring a deep prior knowledge of the document being modelled; Topic composition is overlapping with same word appearing in multiple topics which requires attention in comparing them such as cosine similarity.

2.4.3 Neural Language Model

It was shown that neural networks can be used to obtain distributed representations of concepts as early as 1986 by Hinton et al.[57]. Neural network language models represents words as high dimensional real valued vectors. Advantages of these models are that, they achieve a level of generalisation in distributed representation that is not possible with n-gram language models as the latter don't capture the inherent word relationships [52]. One significant problem of the n-gram or statistical language models is the *curse of dimensionality*. A word sequence where the model was trained on is likely to be different from what it sees at test time [56]. To mitigate the *curse of dimensionality*, Bengio et.al [56] proposed the following approaches.

- Associate each word in the vocabulary to a real valued feature vector (distributed representation)
- Express the joint probability function of word sequences in terms of the above feature vectors
- Learn both feature vectors and parameters of probability function

The joint probability function is learnt by the layers of the neural network by predicting the next word in the sequence, given previous ones.

Classic neural network language model as shown above proposed by Bengio et al. in 2003 has the basic building blocks that subsequent improvements model up on. The embedding layer generates the word embedding by multiplying the input (1 hot encoded) with a word embedding matrix. The Intermediate layer is a fully connected layer that applies a non linearity and the *softmax* layer that produces the probabilities that sum up to one for each of the word in the vocabulary. This requires the most computation. The goal here is to learn the embedding layer and the hidden layer weights by training to predict the next word in a sequence. The error is propagated back using backpropagation algorithm and the weights are updated proportionately.

"Generalization is obtained because a sequence of words that has never been seen before gets a high probability if it is made of words that are similar (in the sense of having a nearby representation) to words forming an already seen sentence." - (Bengio et al 2003 - A neural Probabilistic Language Model)

Collobert and Weston [54] improved the above neural language model by using a different training strategy. Instead of predicting the probability of the next word given previous, they trained their language model to discriminate a binary classification task: If the word in the middle of the input window is related to its context or not. They constructed a dataset with positive samples where a word window of certain length is considered, and for negative samples the middle word replaced with a random word. They utilised a ranking type cost function and essentially avoided computing an expensive softmax which was a major bottleneck in the previous methods. This also provided better embeddings as the complete context of the word was considered to predict its relevance [54].

Training a neural network language model is versatile as the learnt word representations can be used for other tasks [54]. Collobert et al trained a convolutional neural network language model that can generate a host of outputs such as POS tagging, chunking, NER parsing etc. This was possible as the vector representations learnt by the model could be shared across these various tasks. Neural language models were shown to perform significantly better than LSA and LDA [53][55].

2.5 Word embeddings

Word embedding is a neural language model that maps a vocabulary of words to a set of vectors of real numbers. Neural networks operate on real valued numbers rather than textual strings. It represents the words in a high dimensional vector space, which integrates the distributional semantic information to a unit that can be processed. Therefore word embeddings model a language in a rich form which is amenable for various downstream processing. In QA tasks, it is used to represent words of a question as well as a candidate answer and project them both to a common vector space where similarities are computed.

Word embeddings are increasingly used for natural language processing tasks as they perform better and pre-trained embeddings are readily available, compared to previous methods described above. In the sections below, these models and techniques are detailed.

2.5.1 Word2Vec

Word2Vec is a versatile neural language modelling technique. It is one of the key ingredients of deep learning models for NLP. In 2013 Tomas Mikolov presented a technique that can generate high quality word vectors from a very large dataset (billions of words) with a large vocabulary (millions of words) [52]. These vectors represent words in such a way that not only similar words tend to be closer, but also contains multiple degrees of similarity [53]. There are two main learning algorithms in word2vec, namely, *continuous bag of words (CBOW)* and *skip-gram*. These two algorithm are trained with two strategies, namely, *hierarchical softmax* and *negative sampling*. Word vectors generated by this technique, captures many linguistic regularities. For example:

$$V('Paris') - V('France') + V('Italy') \approx V('Rome')$$

$$V('King') - V('Man') + V('Woman') \approx V('Queen')$$

Where V denotes the vector associated with the word.

This type of strong regularities are learnt by training on very large (billions of words) datasets. CBOW and skip-gram techniques are a refinement of a previous attempt to generated word vectors using recurrent neural networks [53]. Main features for its popularity are that it is a simple model and computation is several orders of magnitude faster than the previous methods (training in minutes as opposed to days,weeks or months). This is achieved by avoiding non-linearity in the hidden layer and replacing it with a linear activation.

The two model architectures described below has slightly different performance depending on the task. For example CBOW performs better in syntactic tasks and skip-gram performs better in semantic tasks [52].

Continuous Bag of Words (CBOW)

This architecture is similar to the neural network language model but the non linearity of the hidden layer was removed and projection layer is shared for all words. It is called continuous bag of words as the order of the words do not influence the projection. It predicts the probability of a word given its context. As shown in the Figure 2.4, it is a shallow network with three layers, Input , Projection (Hidden) and Output.

The Input layer has V nodes where V is the size of the vocabulary. Each input word is one hot encoded and fed into the Input layer. Hidden layer has N neurons, representing the dimensionality of the embeddings sought. Weights between Input and the Hidden layer is a matrix of size $V \times N$ and is initialised with small random values. Output layer has V neurons giving a matrix of size $N \times V$ representing weights between hidden and output layers. This matrix is also initialised with small random values.

Therefore we can represent the operation using linear algebra (vector matrix multiplication)

$$H^T = X^T W_i$$

$$Y = Softmax(H^T W_o)$$

$$Error = T - Y$$

Where

$X =$ Input vector (one hot encoded)
$H =$ Input vector at the hidden layer
$W_i =$ Weight matrix between input and hidden layers
$W_o =$ Weight matrix between hidden and output layers
$Y =$ Output probability vector (elements of which adds up to 1.0)
$T =$ Target vector (one hot encoded)

The error is thus backpropagated updating the weights of the two matrices.

In CBOW, All surrounding words are one hot encoded and projected to the continuous distributed space by multiplication with the embeddings matrix, and these words in the window are averaged. The rest of the process is as described above.

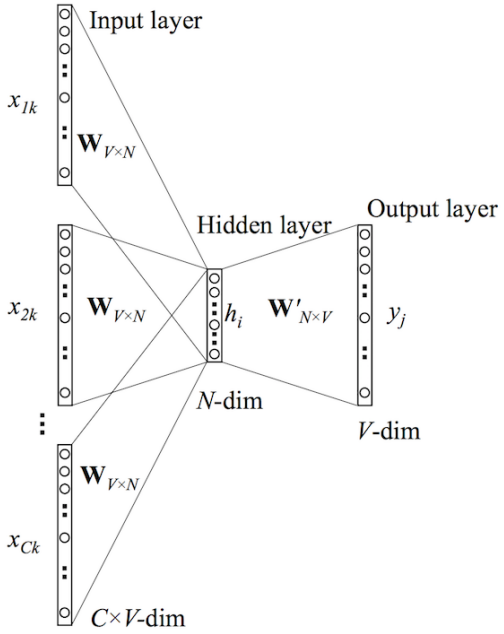


FIGURE 2.4: The CBOW architecture predicts the current word based on the context
Source: *Rong, X. (2014). word2vec Parameter Learning Explained. CoRR, abs/1411.2738.*

Once the training is done for all the words in the corpus, the resulting matrix between hidden and output layer is taken as the word embeddings.

Skip-gram

Skip-gram is the inverse of the CBOW architecture described above. It aims to predict the surrounding words given a word. This architecture contains three layers as in CBOW, Input, Projection and Output and shown in the Figure 2.5. Current word is one hot encoded and multiplied by the embeddings matrix of the projection layer and fed into a *log-linear classifier* (softmax). This classifier predicts the words within a certain window encompassing words before and after the current word.

Training procedure is similar to the CBOW described above, but instead of multiple inputs, there is only one input vector. Instead of single output vector, in the Skip-gram there are multiple outputs generated. Each of the output is compared with the corresponding target vector and error is computed by averaging the resulting errors. Weights of the matrices are updated using backpropagation as before.

Hierarchical softmax

Both CBOW and skip-gram uses softmax for the word probability computations in the vocabulary. This is a very expensive operation specially when the class count

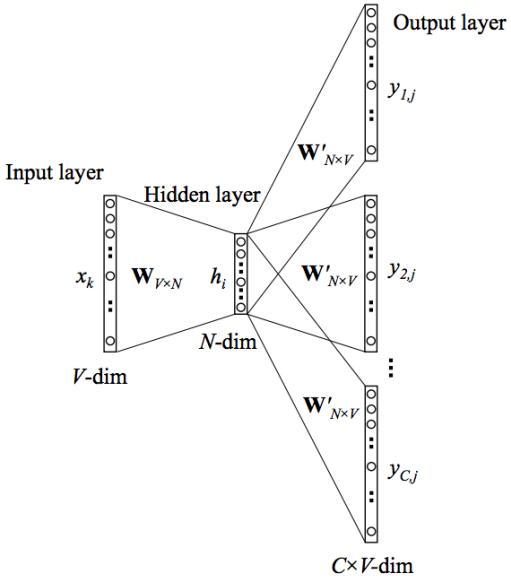


FIGURE 2.5: Skip-gram predicts surrounding words given the current word
Source: *Rong, X. (2014). word2vec Parameter Learning Explained. CoRR, abs/1411.2738.*

increases to thousands and perhaps millions, as required by word2vec. Hieararchical softmax is an efficient approximation to this process which was first introduced by Morin and Bengio [59]. The process cuts down the computation of having to evaluate W output nodes to evaluating about $log2(W)$ nodes. This is achieved by modelling the output layer as a binary tree where each leaf is a class (W word) and each node explicitly representing relative probabilities of its children [58].

Negative sampling

Negative sampling aim to reduce the computation burden of softmax computation. Normally computing softmax requires summing over all the classes (Whole vocabulary). Negative sampling reduces this summing to only a sample of negative classes plus the expected target class. Selecting the negative words is based on their frequency.

2.5.2 GloVe

Global Vectors for word representation or commonly known as GloVe is an improvement on matrix factorisation methods like LSA and the continuous methods like Skip-gram. In doing so they combined best features of both these methods. LSA efficiently leverage the statistical information by operating on the whole corpus but its performance is poor and vector space structure captured is suboptimal. Skip-gram based word2vec operates on a local window and poorly utilises the statistics of the full corpus.

GloVe is an unsupervised learning algorithm. The model is trained on the non zero entries of a global word-word co-occurrence matrix which counts word frequency and computed for a given corpus only once. Matrix factorisation methods described above (LSA/LDA) suffer from having frequent words (for example 'the', 'and') contributing disproportionate amount to the similarity measure.

GloVe addresses this by taking the ratios of word-word co-occurrence probabilities instead of the probabilities themselves.

Advantages of *GloVe* are that it trains fast and scalable to a huge corpora but gives good performance even for a small corpora with small vectors. Model produces a word vector space with meaningful substructure and over attains 75% accuracy on word analogy datasets and claims to perform better than *word2vec* [61].

Chapter 3

Methodology

Question answering task is posed as a ranking of short text pairs. A dataset D is composed of questions q_i and a set of answers A_i for each question. Each answer set A_i has a corresponding set of labels Y_i which has a judgement whether it is correct. This can be formalised as below.

$$(q_i, (A, Y)_i) \in D$$

$$(A, Y)_i = \{(a_{i1}, y_{i1}), (a_{i2}, y_{i2})...(a_{im}, y_{im})\}$$

$$y \in \{0, 1\}$$

The task is to take each (q_i, a_{ik}, y_{ik}) tuple and learn functions F, H.

$$H(F(q_i, a_i), y_i) \rightarrow R$$

where $F(.)$ represents a convolution function, and $H(.)$ represents the function that produces a ranking R.

This essentially is a pointwise approach to reranking and we train a binary classifier which produces a score which has a range 0 to 1 through a sigmoid. We use this score for ranking the set of answers where we assume that higher values represent a high likelihood that it is a correct answer.

Question answer pair is broken into words and assigned a distributional representation with word vectors, which is described in detail below.

We created two convolutional neural network architectures with one and two hidden layers respectively, and conducted experiments with 50 and 100 dimensional Word Embeddings. An Evolutionary strategy algorithm was used for optimising hyper parameters of these two network architectures, and resulting generated models were evaluated for validation set MAP score. We selected the top performing models in each experiment to compare against, and selected the median model of

the best performing combination of architecture and word embedding size, to perform detailed analysis of its properties.

A large number of experiments were carried out and each model was trained with a randomly selected subset of the dataset and validation on the similarly split subset. These operations are detailed in the following section.

3.1 Dataset

The dataset we used was a preprocessed version [1] of TREC8-12 and TREC8-13 dataset from WANG et al.,2007, in the form by Yao et al., 2013 [2]. It is widely used for benchmarks for answer reranking.

TRAIN-ALL dataset comes from the TREC8-12 which had automatic judgements and is therefore noisy. TEST dataset comes from TREC8-13 which are manual judgements. Both these datasets were preprocessed to convert from pseudo XML to a CSV format.

These two datasets contain a set of factoid questions where each question contains a set of answers. The candidate answers are split into sentences. Each question is paired with the corresponding candidate answer and a binary label is assigned to denote if that sentence contains the correct answer.

The dataset we used have the following composition:

TABLE 3.1: Dataset properties

Dataset	Q count	QA Pairs	Correct %
TRAIN-ALL	1216	44647	10.82
TEST	95	1517	18.72

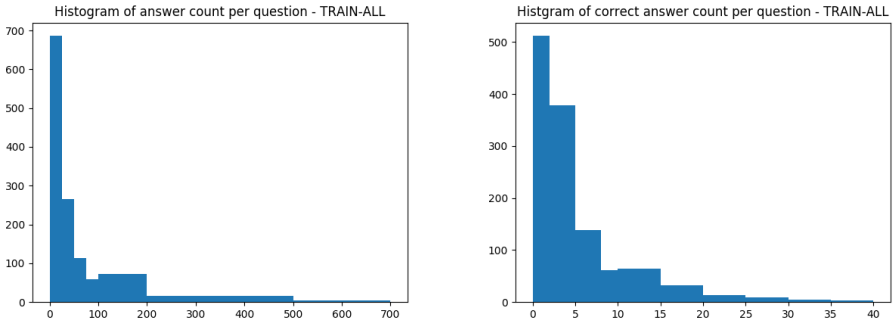
3.1.1 Dataset properties

We analysed the dataset to look how the questions and answers and the correct answers are distributed in both TRAIN-ALL and TEST datasets.

It can be seen from Figure 3.1 that TRAIN-ALL dataset contains a large number of questions (56%) with 25 or less answers. Also contains a large number of questions (42%) with 2 or less correct answers. Upon further inspection there were small number of questions without any correct answer.

[1] from https://github.com/brmson/dataset-sts/tree/master/data/anssel/wang
[2] as downloaded from https://code.google.com/p/jacana/

Figure 3.2 shows that TEST dataset contains 71% questions with less than 20 answers and 37% questions with 2 or less correct answers.

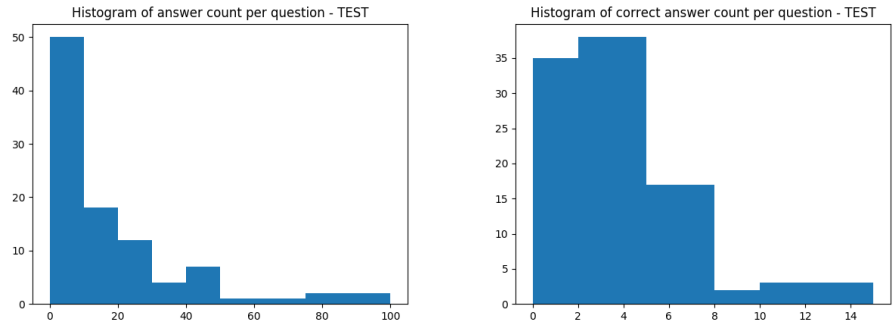


(A) Answer count (B) Correct answer count

```
question count      = [   686, 265, 113,  59,  73,  16,   4]
answer count bins = [0,  25,  50,  75, 100, 200, 500, 700]

question count      = [   512, 379, 138, 61, 64, 32, 13,  9,  5,  3]
correct answer      = [0,    2,   5,   8, 10, 15, 20, 25, 30, 35, 40]
count bins
```

FIGURE 3.1: Answer and correct answer distribution in TRAIN-ALL



(A) Answer count (B) Correct answer count

```
question count      = [   50, 18, 12,  4,  7,  1,  2]
answer count bins = [0, 10, 20, 30, 40, 50, 75,100]

question count      = [   35, 38, 17,  2,  3]
correct answer      = [0,  2,  5,  8, 10, 15]
count bins
```

FIGURE 3.2: Answer and correct answer distribution in TEST

The training and test datasets roughly match in their distributions of answer counts and correct answer counts. Due to the imbalance of the positive and negative samples for the binary classifier, as well as the skewness of the correct answer count, using a binary measure such as {correct,incorrect} would not give an adequate performance measure of the model trained. We use MAP and MRR (described above) for performance measurement and for comparison with other methods.

3.1.2 Word embeddings

We used the public domain pre-trained GloVe word embeddings [3]. Four sets of embeddings resulting from various corpora can be found in this distribution. For the training we used Wikipedia 2014 + Gigaword5 set which contains 6 billion tokens and 400,000 word vocabulary. This is commonly known as 'glove6B'. Embeddings themselves have four sets of dimensions (50,100,200,300). In our experiments we used 50 and 100 dimension embeddings as using any higher dimensional vectors were constrained by memory availability of the computer. The embeddings matrix has to be loaded in to the memory for training and testing.

3.2 Experimentation

The dataset is prepared for training the following way: In order to have no information leakage to the validation set, we split the training and the validation sets on questions, rather than on the resulting samples arising from splicing questions and answers. We used TRAIN-ALL dataset for training and validation with a 8:2 ratio. Models were compared on the validation MAP performance.

The evolutionary algorithm (Genetic Algorithm) used for hyper parameter tuning, encoded the following hyper parameters: filter count, kernel size, pooling window, dense layer sizes, drop out rates, batch size, and epoch count in a genome and subject these to generational improvement through GA operators such as mutation and crossover. Table 3.2 shows the range of each hyper parameter that this optimiser was allowed to vary in. We selected the MAP on validation set as the fitness function for optimisation. Improvement in MAP on validation set indicates the generalisation of the model and our aim was to train a model that generalises well.

TABLE 3.2: Hyper parameter optimisation

Hyper Parameter	Value Range	Type
filter Count	2,50	int
kernel Size	2,6	int
pooling Window	5,50	int
dense Size1	10,50	int
dense Size2	10,50	int
dropout1	0,1	float
dropout2	0,1	float
batch Size	4,64	int
epoch Count	10,50	int

[3] https://nlp.stanford.edu/projects/glove/

After the question and answer had been spliced in, each sample is regarded as a sequence of words. Before tokenisation, the words went through a process of removing basic punctuation and new line characters. An example of a filter shown below.

!"#$%&()*+,-./:;<=>?@[\]^_'{|}~\t\n

From these sequences, a dictionary was created that maps a word in the vocabulary to a numerical index.

After this process the sequences are represented numerically by indexes to the vocabulary. Each sequence is padded with zero to make all sequences the same size. This is because each question and answer sample could have variable number of words and the embedding stage of the deep network requires the vectors to be the same size. Also worth noting that indexing words in the vocabulary starts from 1 in order to represent 0 as the padding.

There are a large number of correct answers that appear at the top of the each answer list. In earlier runs of the models, we noticed that it is possible to get very high MAP scores but upon inspecting we noticed that the model only outputs a constant non-zero value for all the answers. If the answer scores (together with indices) were not shuffled before sorting (for MAP computation), a stable sort algorithm would return the same ordering for equal values.

3.2.1 Building embedding matrix.

Embedding matrix is formulated on the vocabulary generated above. For each word in the vocabulary, embeddings were looked up and matrix is set with the corresponding embedding vector of real values.

We first compute the vocabulary of the training set including the validation set and also the test set. Each sentence in the question and answer is split into words. Embeddings matrix was generated for the vocabulary. Even though the embeddings had a 400k base vocabulary, it is not possible to capture all variations of a word, and about 94% of the vocabulary was covered.

Embeddings contain a considerable number of numbers and dates and other numerical values that are unlikely to be useful. Some of the words captured in word embeddings are very specific. For example:

xp80,xj1100,xdarklegacyx2

Upon further inspections, we found that *xp80* refer to a camera model, *xj1100* refer to a motorcycle model and *xdarklegacyx2* refer to a name of a digital artist. Some of these may be useful in question answer scenario when the question context matches.

A sample of words that didn't have matching embeddings are shown below.

vinalon, faws, kaulukukui, maxvill, nachin, kartuz, mesiger, ffr73, gesco, rnate, phantombiz, masiga, ulmers, zhengmu, mamdou, shehadie, doulgas, enscribed, jungs, duchnowski, lohrey, honorado, wedl, subtropicals, infolink

As can be seen they are mostly very infrequent words (probably nouns) which doesn't seem to make much sense without the context. Embeddings for these words are initialized with all zero vector.

The embedding layer is set as frozen with this matrix in Keras[66]. This is to say that the embeddings layer is pre-trained and not to be changed during the back-propagation. Input layer consists of a vector of word indices representing a sequence. Each word index is converted into a one hot vector and multiplied with the embeddings matrix to get its distributional representation.

Rest of the network contains one dimensional convolutional layer followed by *max-pooling* layer and flattened before being fed into dense layers. The output layer contains one node as this is a binary classification. Each dense layer contains *dropout* setting which was varied by the hyper parameter optimiser. Dense layer non linearity is set to *ReLu* and the output layer non linearity was set to *sigmoid*.

We used *binary crossentropy* loss function and *Adam* optimiser. We also set *early stopping* based on the *validation loss*. Early stopping is required for tractable training as well as producing a model that is not over fitted.

3.3 Machine learning frameworks and tools

The deep learning framework we used was Keras [4], with Tensorflow [67] backend. Keras is written in Python, and is an abstraction over Tensorflow[5] which does the graph computations on heterogeneous hardware such as CPU, GPU. These frameworks are widely used in the machine learning community and are available as open source projects. Our code was written in Python, which is commonly used for machine learning and NLP.

[4] https://keras.io
[5] https://www.tensorflow.org/

Chapter 4

Results

The results of the experiments that were carried out with the two model architectures (single dense layer and two dense layers), are presented here, both with the two word embeddings used (50 and 100 dimensional). The hyper parameter ranges were established and the optimiser was free to select a suitable set of hyper parameters guided by the model performance.

4.1 Model with a single dense layer

Table 4.1 lists the performance of using 50 dimensional word embeddings.

TABLE 4.1: Top 5 Models on validation MAP score. 50d word embeddings and 1 dense layer

Model	Kernel Size	Test MRR	Test MAP	Val MRR	Val MAP
A1	2	0.6394	0.5918	0.5513	0.4871
A2	4	0.6265	0.5812	0.5506	0.4661
A3	2	0.6689	0.6082	0.5349	0.4658
A4	4	0.6607	0.6078	0.5274	0.4624
A5	4	0.6511	0.5921	0.5165	0.4563

Table 4.2 lists the performance of using 100 dimensional word embeddings.

TABLE 4.2: Top 5 Models on validation MAP score. 100d word embeddings and 1 dense layer

Model	Kernel Size	Test MRR	Test MAP	Val MRR	Val MAP
B1	4	0.6464	0.5865	0.5592	0.4899
B2	2	0.6154	0.5605	0.5249	0.4598
B3	2	0.6581	0.5962	0.5306	0.4568
B4	4	0.6454	0.5929	0.5379	0.4549
B5	2	0.6379	0.5801	0.5056	0.4323

4.2 Model with two dense layers

Similar to the above, another set of models having two dense layers were trained and their results are reported here.

Table 4.3 lists the performance of using 50 dimensional word embeddings.

TABLE 4.3: Top 5 Models on validation MAP score. 50d word embeddings and 2 dense layer

Model	Kernel Size	Test MRR	Test MAP	Val MRR	Val MAP
C1	2	0.6815	0.6232	0.5431	0.4810
C2	2	0.6317	0.5836	0.5390	0.4762
C3	2	0.6546	0.5826	0.5448	0.4520
C4	2	0.6686	0.6146	0.5287	0.4499
C5	2	0.6479	0.5905	0.5412	0.4466

Table 4.4 lists the performance of using 100 dimensional word embeddings.

TABLE 4.4: Top 5 Models on validation MAP score. 100d word embeddings and 2 dense layer

Model	Kernel Size	Test MRR	Test MAP	Val MRR	Val MAP
D1	5	0.6298	0.5461	0.6023	0.5060
D2	3	0.6678	0.6058	0.5754	0.4946
D3	3	0.6757	0.6075	0.5309	0.4737
D4	3	0.6975	0.5961	0.5490	0.4735
D5	3	0.6786	0.6046	0.5165	0.4553

Above results are summarised in the two figures, Figure 4.1 and Figure 4.2 in the two main performance measures. Two model architectures are labelled as L1 and L2 to denote 1 and 2 dense layers respectively. From these results it can be seen that the L2 model architecture with 100d word embeddings performs better.

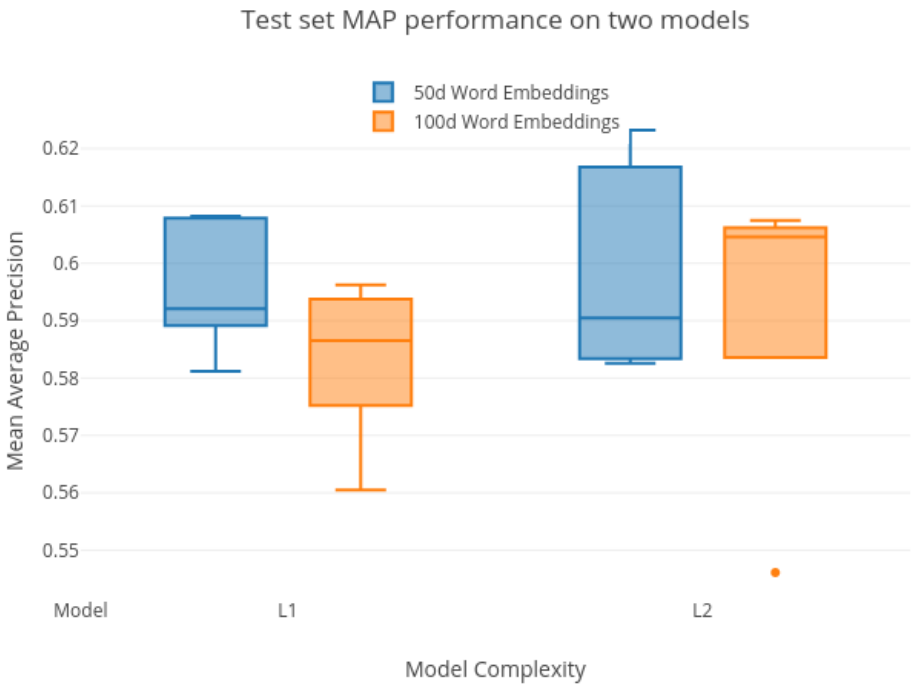


FIGURE 4.1: Model MAP Performance. L1 = One dense layer L2 = Two dense layers

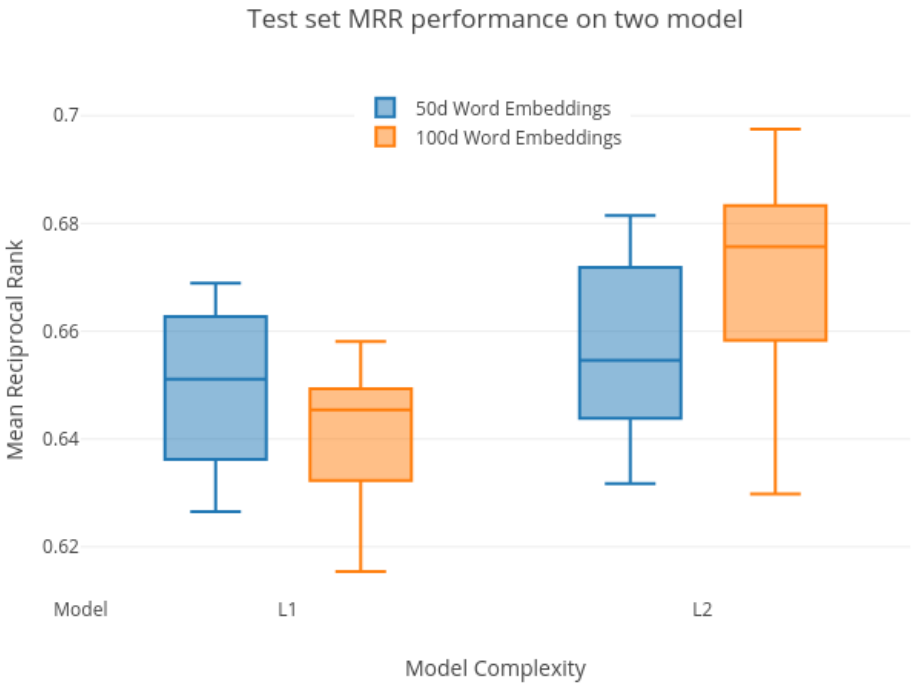


FIGURE 4.2: Model MRR Performance. L1 = One dense layer L2 = Two dense layers

Chapter 5

Analysis

We generated many models in order to select the best performing ones. With a lower complexity and lower word embedding dimension, median MAP score was about 0.59 (Figure 4.1). This result drops when the word embedding dimension is increased, indicating that possible under-fitting. More distributional semantics is encoded with a larger embedding dimension, but the models are unable to utilise the wider representation due to insufficient parameters. When the model architecture contains 2 dense layers the performance increases with the embedding dimension.

Kernel size represents the number of words covered in a convolutional operation, and this would be similar to using uni-gram, bi-gram, tri-gram etc. capturing word wise wider ranges. Evolutionary optimiser can uniformly randomly assign a kernel size of 2 up to 6 to each generated model, and the winning models tend to contain tri-grams (Column 2 of Table 4.4). It's interesting to note that with the single dense layer architecture, kernel size alternates between the 2 and 4 while on 2 dense layer architecture, selected kernel sizes are distinct for each of the dimension sizes.

In Table 5.1 performance is compared to the previous methods of Yu et al. [44] and Severyn et al. [43]. Our method performs better than Yu et al. but lower than what Severyn et al. has achieved. This could be attributed to us training and validating on the TRAIN-ALL data resulting in lower amount of samples to train with.

TABLE 5.1: Comparison of results previous methods

Model	**MAP**	**MRR**
TRAIN-ALL		
Yu et al. [44] (unigram)	.5470	.6329
Yu et al. [44] (bigram)	.5693	.6613
Severyn et al. [43]	.6709	.7280
Our Model	**.6046**	**.6757**

Source: Severyn et al. [43] Table 2. (Our Model MAP score is the median of Test MAP scores in Table 4.4)

5.1 Detailed look at the Test set performance

Next we look at the kind of questions the best model performs best, worst and the median according to the average precision yielded on each question on the TEST set.

5.1.1 Questions yielding best Average Precision

In this analysis we omit the questions that contained only one answer which would give a high AP value automatically. These questions are listed in the table A.5 in Appendix A.

The question Q13 and its answers and scores are listed in Table 5.2. The answer 0 contains the words *first, Burger, King, restaurant, opened* from the questions and expectedly, the answer was predicted to be highly probable. The answer 1 contains the words *first, Burger, King* followed by answer 2 which contains only the word *first*. Hence the model is capable of exploiting the shared words for prediction.

TABLE 5.2: Q13 Where was the first Burger King restaurant opened ?

Index	Score	Label	Answer
0	0.4797	1	"Coke has supplied Burger King for most of the restaurant chain 's history , starting with the first Burger King that opened in Miami in <num> ."
1	0.2676	1	<num> : The first Burger King opens in Miami .
2	0.0940	0	"Kroc opens his first restaurant in Des Plaines , Ill . , and eventually buys out the McDonalds ."

On the question Q15 (Table 5.3) the correct answer (00) was predicted with a good margin. In the highest ranked answer(00), the only fully matching words are *Lewis, Clark*, and partially matching *accompany*. *accompany* and *accompanied* are both present in word vectors as such a strong relationship could be associated.

Answer ranked second (03) matches only the words *expedition* and *Sacajawea* is misspelt as *Sacagawea* but the latter is present in the word embeddings. The words such as *believed to have died* correlates strongly with an expedition as such it received higher ranking. Lower ranking answers have little common with the question. Therefore it can be seen that the model had learnt to associate some form of distributional similarity between the question and a potential answer, beyond simple word overlap matching.

Another example, on Q31 shown in Table 5.4 The first ranked answer only contain the matching words *Constitution,commissioned* Tokens such as <*num*> would become *num* after stripping out the non alpha numeric characters in the tokenisation process. The word embeddings contain a vector for *num* as such it would be meaningfully represented (i.e not a null vector). When looking at the domain similarity, the question is related to naval ships, and the highest ranked answer captures this property followed by the rest of the rankings. Looking at the last ranked answer on its own,

TABLE 5.3: Q15 What years did Sacajawea accompany Lewis and Clark on their expedition ?

Index	Score	Label	Answer
00	0.4942	1	The coin honors the young woman and teen-age mother who accompanied explorers Meriwether Lewis and William Clark to the Pacific Ocean in <num> .
03	0.4390	0	"Little is documented about what happened to Sacagawea after the expedition , but she is believed to have died Dec . <num> , <num> ."
06	0.4231	0	"Q : Nearly <num> years ago , a nuclear reactor failed at the Three Mile Island power plant in Pennsylvania , creating a crisis ."
13	0.3233	0	"Her son , Jean-Baptist , was later raised in part by William Clark ."
01	0.2697	0	"In <num> , Toussaint was hired by Lewis and Clark , not for his own skills but for those of Sacagawea ."
05	0.2649	0	"A month later , a Japanese envoy made what was believed to be the first official apology on American soil for the attack <num> years earlier ."
14	0.1997	0	"Also , as other historians have noted , Svingen said , the Lemhi were told they would lose their food rations if they did not agree to leave the valley ."
09	0.1802	0	The Lewis and Clark bicentennial is going to be our last fight .
08	0.1649	0	But that tumbledown village was destroyed by the local authorities more than <num> years ago .
15	0.1639	0	The Lewis and Clark bicentennial is our last fight .
11	0.1592	0	"Lewis was counting on Sacajawea to be a translator with Shoshone people who lived in the mountains west of the Continental Divide , and he expressed concern that her death could doom the expedition ."
02	0.1548	0	"She knew several Indian languages , and being Shoshone could help the explorers make contact with her people and acquire horses that were crucial to the expedition ."
12	0.1296	0	"For her services on the expedition , Sacajawea was given nothing , though her husband , Charbonneau , received $ <num> ."
10	0.1273	0	"When Charbonneau signed on with Lewis and Clark , Sacajawea , though just <num> years old and six months pregnant , was enlisted as an interpreter ."
04	0.1043	0	"A : Yes , but many years after the fact ."
07	0.0790	0	"The transaction , the first sale of a nuclear plant by a U.S . utility , is expected to take one to two years to win approval from regulators , including the NRC ."

it is hard to say that this belong in naval ships domain and it is ranked as expected in that sense.

TABLE 5.4: Q31 When was the USS Constitution commissioned ?

Index	Score	Label	Answer
0	0.1720	1	"Dockside at the Charlestown Navy Yard , you can visit " Old Ironsides , " the Constitution , commissioned in <num> and never decommissioned ."
1	0.1276	0	"CHARLESTOWN NAVY YARD , USS Constitution Museum , Building <num> in the Charlestown Navy Yard , off Chelsea Street ."
4	0.1041	0	"Everything in the museum is related to the Constitution , docked at Pier <num> in the Navy Yard ."
3	0.0969	0	"Although it 's still a commissioned warship -LRB- with a crew of <num> officers and seamen -RRB- , the Constitution is more of a museum than anything else these days ."
2	0.0804	0	The Constitution also offers a pretty good place to survey the Boston skyline .

5.1.2 Questions yielding median Average Precision

Here we look at a few samples from the median 10 performing questions. The Table A.6 in the Appendix A contains a list of these questions answers along with their scores.

The Q77 shown in Table 5.5, the first ranked answer (0) contains consecutive matching words toward the end of the answer. Therefore expectedly it is placed in the correct rank. The second ranked answer (4) contains with it, the words *shuttle, space*, but more importantly *florida* and *researchers* towards the end of the sentence. The third ranked answer (2) contains the correct answer even though it was labelled incorrectly. But ascertaining that this is a correct answer just by looking at the candidate answer is not possible without resorting to real world knowledge about space shuttles. But as these words *florida, columbia, canaveral, kennedy* appear in the word embeddings, model seem to be evaluating the distributional similarities along the question answer combined sample. The last ranked answer (6), although contains

the word *shuttle* may or may not describe space flight. It could be said of any transportation method. But the higher ranked answers all has some similarity with space flight.

TABLE 5.5: Q77 What is the name of the first space shuttle ?

Index	Score	Label	Answer
0	0.4274	1	"It was among such treasures as Atlas , Apollo and space shuttle models ; his last job , before he retired in <num> , was to prepare the first space shuttle , Columbia , for launching in <num> ."
4	0.2592	0	"The shuttle landed right on time at the Kennedy Space Center in Cape Canaveral , Florida , where about <num> researchers waited with scalpels ."
2	0.2112	0	"During their space voyage , the astronauts will set as many as <num> small fires in Columbia 's laboratory ."
5	0.2021	0	The trip added <num> million miles to the odometer of the oldest shuttle of the National Aeronautics and Space Administration -LRB- NASA -RRB- .
1	0.1386	1	This is the third shuttle mission this year and the 83rd since Columbia made the first shuttle flight in <num> .
3	0.1061	0	"Liftoff was originally scheduled for <num> p.m . this afternoon , but a troubled leak check of the shuttle cabin made NASA delay the launch for another <num> minutes ."
6	0.0426	0	"After the shuttle landed , the crew was hustled off to medical tests that were expected to go on for days ."

A question like The Q68 (Table 5.6) *What rank did Nimitz reach ?* is difficult even for a human to know an answer to without specific knowledge about who or what *Nimitz* is. There is only very limited amount of information available for the model to associate the question with a domain.

Looking at the ranked answers below, even though the model got the first ranked answer correct, it only has the words *rank, reach, nimitz* on the question side to form some similarity with a domain. Surprisingly the proper noun *nimitz* appears in the 100d GloVe vectors along with the other two words.

Since we feed both question and answer concatenated, the model have no way to distinguish a question and an answer apart from the position of the word vector in the input tensor to the ConvNet. The word vectors from the beginning of the sample are more likely to be a question and from the ending are more likely to be an answer. Therefore model most likely is learning to match symmetrical distributional similarities. Here it is more likely that the model is using strong representations of the words *rank, reach, nimitz, bush* in the beginning of the sample and matching with the words *admiral, nimitz, museum, historical* at the end of the sample. Specially when the question contains only a small number of words compared to the answers, it is more likely that the model tends to match more of the beginning of the answer with its own ending.

TABLE 5.6: Q68 What rank did Nimitz reach ?

Index	Score	Label	Answer
0	0.4551	1	"Bush cut the ribbon Friday on the George Bush Gallery of the National Museum of the Pacific War , part of the larger Admiral Nimitz Museum and Historical Center ."
4	0.1936	0	"At the far end is the serene Japanese Garden of Peace , a gift from the people of Japan to symbolize the friendship between Nimitz and their national hero Adm . Heihachiro Togo ."
2	0.1594	1	"To commemorate Nimitz ' illustrious career , his grandfather 's Nimitz Steamboat Hotel at Main and Washington streets was turned into the Admiral Nimitz Museum and Historical Center , a state historical park managed by Texas Parks and Wildlife ."
6	0.1348	0	"Bush explains that he served under Adm . Chester Nimitz , the Fredericksburg native for whom the complex is named ."
5	0.1302	0	The two did n't get a chance to meet .
1	0.1208	1	"The museum is named for World War II Pacific fleet commander and Fredericksburg native Admiral Chester Nimitz , under whom Bush served ."
3	0.1178	1	It is part of the Admiral Nimitz Historical Center in Fredericksburg and is under the Texas Parks and Wildlife Department .
7	0.0781	0	"The primary purpose , she said , is to reach a younger audience and give them a better understanding of the contributions that were made by the generation of World War II ."

5.1.3 Questions yielding worst Average Precision

The worst performing question like Q14 *Where are the company Conde Nast 's headquarters ?* had the following answer ranking as shown in Table 5.7. Word embeddings contain the words *company, conde, nast, headquaters, wired, subscribers*. Looking at the higher ranked answers, large parts of each answer are highly correlate with the words in the questions. The one correct answer should also be highly correlate with the question, but having a smaller length seem to have given it a lower likelihood. A word overlap feature like devised by Severyn et al [43] could have given this answer a higher ranking, but having too little data to learn a such feature hampers the model. In retrospect, this was the very reason they introduced the word overlap feature manually into the model.

TABLE 5.7: Q14 Where are the company Conde Nast 's headquarters ?

Index	Score	Label	Answer
17	0.4269	0	"He has some other tricks up his sleeve as well : In July , Digimarc , an Oregon company , will give special PC devices to <num> Wired subscribers ."
08	0.4175	0	"Drew Schutte , who 's been with the magazine since <num> , ascended to the publisher 's post but is based in New York , home of Wired parent Conde Nast ."
02	0.3122	0	"NEW YORK _ With the coordination of an invading army , Conde Nast has begun moving its glossy fashion magazines into a new home at <num> Times Square , the first skyscraper built in New York City since <num> ."
14	0.3094	0	"The company had just been bought by Walt Disney Co . as part of the Capital Cities acquisition , and Disney did not seem a natural fit ."
13	0.3052	0	"Instead of giving all the window space to ranking executives , the company has decided to let many employees labor at metallic blue workstations near the windows ."
05	0.2748	0	"Conde Nast will be moving from its old-money environs on the East Side , within easy reach of Brooks Brothers , Paul Stuart and Patroon , to the hurly-burly of Times Square ."
18	0.2437	0	"In October , a company called Digital Convergence will send similarly equipped laser " pens " that link to the PC to every Wired subscriber , so that they can read bar codes and launch Web sites ."
03	0.2284	0	"Since Conde Nast signed a deal for the building nearly three years ago , Times Square has become one of the most sought-after commercial neighborhoods in the city ."
19	0.2041	0	"Meanwhile , Schutte 's old boss Lyon is working feverishly to start her new company , One Ventures , which will produce a magazine and Web site devoted to all things design this fall ."
16	0.1981	0	"Advertisers such as Ralph Lauren and Kenneth Cole have found it " easy to add Wired to the plan " of other Conde Nast buys , such as Vanity Fair , the New Yorker , Vogue and GQ ."
09	0.1954	0	""" The whole thing was to engage women and get them to take action , " said Cara Deoul Perl , the vice president for creative marketing at Conde Nast in New York , a unit of Advance Publications ."
15	0.1766	0	"Kenneth Cole , whose footwear and clothing company earned $ <num> million in revenue in <num> , said the situation resembled a kind of monopoly on style ."
07	0.1471	0	"Truman , the editorial director , said Conde Nast had a choice of either leaving its longtime home on Madison Avenue for a year while the space there was renovated , or moving permanently to a new building ."
00	0.1404	1	Other issues were bandied about in the brand-new Conde Nast headquarters in Times Square in Manhattan .
04	0.1290	0	"Reuters recently started work on its new headquarters across from <num> Times Square , and Ernst & Young , the accounting firm , is close to a deal to build its own tower nearby ."
01	0.1169	0	"Magazine , will be editor in chief of the San Francisco company ."
20	0.1016	0	"Three of the company 's publications , Wired , Architectural Digest and Bon Appetit , are based in California but have business offices at <num> Times Square , Ms . Perl said ."
11	0.0950	0	"Newhouse averaged half that figure , although he pulled out all the stops when it came to the company 's cafeteria , where , executives said , spending exceeded $ <num> a square foot ."
06	0.0771	0	"It was Douglas Durst , the developer of <num> Times Square , who persuaded Conde Nast to make the move ."
10	0.0691	0	" And the company was not madly for Adlai .
12	0.0681	0	"The company chose to bring the magazines , scattered among six buildings , into the same tower ."

The Q44 Table 5.8 asks a question that require a number as an answer. The top ranking answers mostly contain <*num*> token in them or terms that correlates well with some notion of time. The model can be said to have learnt to associate *When* with numbers, but selecting the correct answer from this list is extremely difficult, without being able to distinguish between the type of numbers that associates with time and the types of numbers that associates a financial quantity, etc..

Figure 5.1 provides an overview of the distribution of average precision of the questions based on their classes. There is a higher number of *What* questions very low number of *How* questions. Model seem to perform well for questions expecting a numerical answer (HowMuch and When) and better for *What* questions.

TABLE 5.8: Q44 When did Jack Welch become chairman of General Electric ?

Index	Score	Label	Answer
25	0.3044	0	"Welch , who announced the date on CNBC on Monday evening , will turn <num> on Nov . <num> , <num> , and had always indicated he would step down then ."
24	0.2957	0	"Everyone knew it was coming , but now they know when : John F . Welch Jr . , the chairman of General Electric , will retire after the company 's annual meeting in April <num> ."
11	0.2282	0	"Bill Gates may be America 's best known corporate leader , but in business circles , Jack Welch is probably more admired ."
15	0.2269	0	"More than <num> GE jobs have been axed under Welch , in a series of downsizings that have devastated old industrial towns like Pittsfield and Schenectady , N.Y . Did Welch need to be so hard-nosed ?"
04	0.1966	1	"When Welch was named chief executive in <num> , most of the also- rans quit ."
05	0.1936	0	"In his author 's note , he points out that the subject of his book , Jack Welch , chairman of General Electric Co . , refused to cooperate with the project ."
09	0.1734	0	Welch rolled up his sleeves and went to work .
29	0.1502	0	"He whittled away at GE 's bloated staff long before downsizing became a buzzword , earning him the rubric of Neutron Jack _ someone who kills off the people but leaves the physical assets intact ."
03	0.1493	1	Welch has turned what had been a $ <num> billion manufacturing company in <num> into a $ <num> billion behemoth that derives huge portions of its revenues from more profitable services .
19	0.1480	0	"Someday , a book will come along that provides a balanced account of Jack Welch ."
22	0.1445	0	"" The Internet is such a new phenomenon that it would be too limiting to mandate hands-on experience , " said Thomas J . Neff , chairman of U.S . operations for the search firm Spencer Stuart ."
34	0.1409	0	"Welch has often said he thinks chief executives should stay for <num> , even <num> years , to live with the results of their long-term decisions ."
14	0.1310	0	O'Boyle has plenty to say about the problems of the Welch years .
12	0.1289	0	Welch has created winners in at least half a dozen separate fields .
10	0.1246	0	Welch did n't invent the forces of modern capitalism .
00	0.1182	1	"When Jack Welch took over GE in <num> , he inherited a slow-moving , bureaucratic company ."
23	0.1121	0	"That worries Kelly , <num> , who became chairman of LTV in February ."
32	0.1076	0	"" The businesses have a momentum in market share and profitability that will persist when Jack is retired , " said Russell A . Leavitt , an analyst at Banc of America Securities ."
01	0.1060	1	"For that matter , the World Wide Web did not even exist when John F . Welch Jr . took over the General Electric Co . in <num> ."
20	0.1053	0	Nor did anyone at the LTV Corp . question Peter E . Kelly about the marketing potential of the Internet when he was named president of the company in <num> and became a front-runner for the top job .
17	0.0941	0	"While Welch has never told his employees to break the law , the high-pressure culture he has created may have encouraged some of his underlings to cross the line in pursuit of greater profits ."
33	0.0925	0	" And Jack will personally introduce him to the major customers . "
27	0.0910	0	"In a sense , Welch will be both a difficult and an easy act to follow ."
02	0.0879	1	"Welch became GE 's chief executive in April <num> , so the date will mark his 20th anniversary ."
07	0.0876	0	"O'Boyle has made Welch the poster boy for modern capitalism and all the ills that accompany it : downsizing , deal-making , ethical violations , pollution ."
26	0.0836	0	"" Jack still plans to announce his successor in mid-2000 , so this is business as usual for us , " said Beth Comstock , a GE spokeswoman ."
18	0.0830	0	"In the business press , Jack Welch receives very different treatment ."
08	0.0811	0	"If the book had come out a few months later , O'Boyle probably would have blamed Welch for the Asian financial crisis and the collapse of the ruble ."
06	0.0742	0	" At Any Cost " is actually a remarkably one-sided book about Welch 's tenure at GE .
35	0.0682	0	"At <num> , Immelt could conceivably serve as long as Welch has ."
31	0.0658	0	Only people whom Welch deemed to be team players were propelled up the ranks .
16	0.0620	0	"It is also fair to ask how much blame Welch should bear for some of the ethical lapses that have happened on his watch , especially in the defense business ."
30	0.0618	0	"Welch has also turned GE into what he calls a " boundary-less " organization , a place where executives are rewarded for sharing insights and customer contacts and castigated for keeping good ideas to themselves ."
13	0.0520	0	He dismisses Welch 's foray into finance _ probably his greatest triumph _ in a few paragraphs .
28	0.0490	0	Corporate wags have long referred to GE as " the House that Jack built. "
21	0.0354	0	"" Even if they do n't specify Internet savvy by name , they are asking for it , " said Gerard R . Roche , chairman of Heidrick & Struggles , a search firm ."

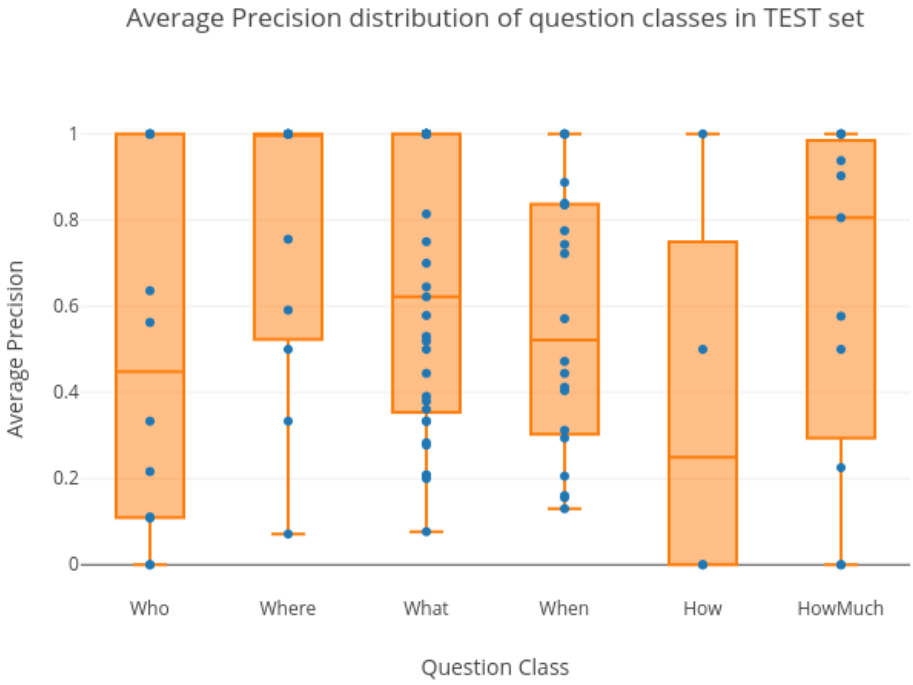


FIGURE 5.1: Average Precision distribution of question classes of TEST set

Chapter 6

Conclusion

We created a ConvNet classifier taking word embeddings as the distributional semantics model for question answer pair ranking. Compared to the comparable methods used by Severyn et al. [43] and by L. Yu et al [44] we attained similar performance on the TRAIN-ALL dataset, but with variable size of convolutional kernels allowing the model to capture wider dependencies, and using wider word embedding dimensions allowing a richer distributional representation. Our method does not require any additional feature engineering (compared to [43]), and simpler to implement. The dimensionality of the word embeddings play a major role in achieving better MAP score coupled with a deeper architecture.

Coupled with the evolutionary hyper parameter optimiser, the experimentation can be massively parallelised to use the best combination of hyper parameters that improves the measured model performance. This is in contrast to the usual grid search and can discover the best set of hyper parameters in shorter time due to the search space culling generally performed by evolutionary algorithms.

The ConveNet learns a function using the distributional semantics of combined question and answer, which can be used for ranking the answers. The datasets that were used is orders of magnitude smaller, compared to the usual deep learning experiments in other domains such as vision (where several millions of images used). ConvNets are capable of learning from massive amounts of data but what is holding back the improvement of the reranking tasks therefore seem to be the lack of large enough labelled datasets. Therefore it would be interesting to experiment on the relatively new Stanford Question Answer dataset (SQuAD)[1] which contains over 100,000 question answer pairs from wide variety of articles that were human annotated to the correct span of text in a contending paragraph.

Factoid answering has become somewhat mainstream due to the widespread use of smart speaker devices. But these devices need a robust and scalable end to end systems. Therefore to be a real world deployable QA system, this method (and the similar methods) needs more work in terms of extracting the exact answer from the

[1] https://rajpurkar.github.io/SQuAD-explorer/

highest ranked answer candidates. This require a model that can operate on the word level rather than the sentence level we've looked at.

ConvNet architectures and word embeddings show a great promise in the question answer reranking tasks. But recursive neural networks capture the hierarchical nature of a natural language better. Socher et al. [45] had great success in a number of NLP tasks. It would be interesting to apply these ideas in a question answering context.

Appendix A

Appendix A

A.1 Code and experimentation results

Due to the size of the code and results and typesetting issues, the code used and the results of the experimentation is located at https://github.com/Indy9000/qa-experiments and relevant files are listed below.

```
The main experimentation files are:
v2/exp-11.py

Hyper parameter optimiser:
v2/ea11.nim

Results output:
v2/output/20170820-2208
v2/output/20170821-1524
v2/output/20170822-1821
v2/output/20170822-0109
```

Result outputs were processed through a filter (filter3.nim) for publishing, which computed the correct MAP and MRR values and returned the top n results. The experimentation was on Linux platform using Python 2.7, Keras and Tensorflow. Installation instructions are in their respective websites (Installations can be done by trivially following the instructions on their respective websites). Hyper parameter optimiser was written in a language called nim-lang https://nim-lang.org/ which is python like but with static typing and extremely fast binaries (comparable to C). This type of a language is better suited for machine learning in my opinion as it relieves the coder from discovering mistakes several days into a long running experiment.

A.2 Hyper parameters of the models reported

TABLE A.1: Hyper parameters of 50d L1

Model	Filter	Kernel	Pooling	Dense 1	Dropout 1	Dense 2	Dropout 2	Batch	Filename
A1	18	2	08	34	0.6175	0	0.0	42	output/20170820-2208//output8-0008-0028.txt
A2	09	4	05	36	0.6927	0	0.0	54	output/20170820-2208//output8-0000-0010.txt
A3	18	2	08	34	0.6175	0	0.0	42	output/20170820-2208//output8-0009-0028.txt
A4	09	4	05	36	0.6901	0	0.0	54	output/20170820-2208//output8-0004-0010.txt
A5	09	4	05	36	0.6978	0	0.0	54	output/20170820-2208//output8-0007-0010.txt

TABLE A.2: Hyper parameters of 100d L1

Model	Filter	Kernel	Pooling	Dense 1	Dropout 1	Dense 2	Dropout 2	Batch	Filename
B1	09	4	05	36	0.6899	0	0.0	54	output/20170821-1524//output8-0005-0011.txt
B2	11	2	20	35	0.5272	0	0.0	27	output/20170821-1524//output8-0001-0012.txt
B3	10	2	09	17	0.5175	0	0.0	36	output/20170821-1524//output8-0000-0003.txt
B4	09	4	05	36	0.6893	0	0.0	54	output/20170821-1524//output8-0007-0010.txt
B5	10	2	09	17	0.2201	0	0.0	36	output/20170821-1524//output8-0008-0003.txt

TABLE A.3: Hyper parameters of 50d L2

Model	Filter	Kernel	Pooling	Dense 1	Dropout 1	Dense 2	Dropout 2	Batch	Filename
C1	37	2	05	44	0.5610	35	0.2743	54	output/20170822-1821/output9-0004-0024.txt
C2	49	2	06	42	0.5298	35	0.3960	33	output/20170822-1821/output9-0002-0011.txt
C3	37	2	05	44	0.5611	35	0.2710	54	output/20170822-1821/output9-0000-0025.txt
C4	37	2	05	45	0.5655	35	0.2786	54	output/20170822-1821/output9-0008-0024.txt
C5	02	2	08	27	0.0869	28	0.1049	05	output/20170822-1821/output9-0002-0023.txt

TABLE A.4: Hyper parameters of 100d L2

Model	Filter	Kernel	Pooling	Dense 1	Dropout 1	Dense 2	Dropout 2	Batch	Filename
D1	45	5	05	19	0.7587	29	0.0735	11	output/20170822-0109/output9-0007-0003.txt
D2	43	3	11	23	0.8090	31	0.0007	56	output/20170822-0109/output9-0005-0015.txt
D3	43	3	11	23	0.8083	31	0.0016	56	output/20170822-0109/output9-0008-0014.txt
D4	43	3	11	24	0.8095	31	0.0056	56	output/20170822-0109/output9-0005-0014.txt
D5	43	3	11	24	0.8062	31	0.0022	56	output/20170822-0109/output9-0000-0015.txt

A.3 Performance on questions

TABLE A.5: Top 10 Average Precision Questions

QID	Correct Answer/Total	Predicted (index,score)
QID 1	[0]/1	[(0, 0.080708578)]
QID 13	[0,1]/3	[(0, 0.47971076), (1, 0.26769838), (2, 0.094062768)]
QID 15	[0]/16	[(0, 0.49426109), (3, 0.43904462), (6, 0.42319012), (13, 0.32338208), (1, 0.26975623), (5, 0.26490003), (14, 0.19977526), (9, 0.18029062), (8, 0.16498362), (15, 0.16398801), (11, 0.15920295), (2, 0.1548472), (12, 0.12960735), (10, 0.12731576), (4, 0.1043046), (7, 0.079039559)]
QID 17	[0,1]/2	[(0, 0.095745072), (1, 0.051032498)]
QID 19	[0,1,2,3,4]/5	[(0, 0.28008199), (3, 0.2604003), (1, 0.24749805), (2, 0.24299787), (4, 0.19781223)]
QID 25	[0]/1	[(0, 0.23515154)]
QID 26	[0]/3	[(0, 0.1391563), (2, 0.05178307), (1, 0.039683454)]
QID 27	[0,1]/4	[(0, 0.33593705), (1, 0.1955101), (2, 0.1391733), (3, 0.13878661)]
QID 29	[0,1,2]/3	[(0, 0.41168875), (1, 0.23586008), (2, 0.12134192)]
QID 30	[0]/1	[(0, 0.10771196)]
QID 31	[0]/5	[(0, 0.17207976), (1, 0.12766305), (4, 0.10416951), (3, 0.096948653), (2, 0.080444783)]

QID QUESTION
01 Whom did Ramirez marry ?
13 Where was the first Burger King restaurant opened ?
15 What years did Sacajawea accompany Lewis and Clark on their expedition ?
17 How many employees does Amtrak have ?
19 When was the Muslim Brotherhood formed ?
25 Who is Public Citizen 's current head ?
26 Whom did Eileen Marie Collins marry ?
27 Where did the mass suicide of Heaven 's Gate occur ?
29 Where was Carlos –LRB– Ramirez –RRB– captured ?
30 What is Muslim Brotherhood 's goal ?
31 When was the USS Constitution commissioned ?

TABLE A.6: Median performing Average Precision Questions

QID	Correct Answer/Total	Predicted (index, score)
QID 77	[0, 1] / 7	[(0, 0.42744783), (4, 0.25926051), (2, 0.21129765), (5, 0.2021139), (1, 0.13861111), (3, 0.1061502), (6, 0.042657226)]
QID 68	[0, 1, 2, 3] / 8	[(0, 0.45517927), (4, 0.19366612), (2, 0.15949067), (6, 0.13481943), (5, 0.13020456), (1, 0.12083364), (3, 0.11789753), (7, 0.078111053)]
QID 00	[0, 1, 2, 3, 4, 5] / 20	[(8, 0.51463199), (3, 0.50260901), (2, 0.47356001), (4, 0.46648377), (1, 0.4376713), (14, 0.3996841), (18, 0.38790211), (10, 0.38304693), (5, 0.37477574), (19, 0.33108979), (0, 0.32431364), (11, 0.27157751), (7, 0.1897123), (15, 0.1824557), (12, 0.10650191), (6, 0.092331626), (9, 0.088399075), (16, 0.084870987), (17, 0.079057015), (13, 0.03162507)]
QID 63	[0, 1, 2, 3] / 7	[(6, 0.44438973), (1, 0.30231044), (2, 0.23263942), (0, 0.22382495), (4, 0.20308091), (5, 0.18402275), (3, 0.18197143)]
QID 07	[0, 1, 2, 3, 4] / 9	[(4, 0.34921351), (7, 0.31277677), (6, 0.20195794), (8, 0.18318775), (1, 0.13466378), (2, 0.12012611), (5, 0.10475385), (0, 0.10016962), (3, 0.093827359)]
QID 34	[0, 1, 2, 3, 4, 5] / 40	[(19, 0.48262581), (3, 0.36977518), (5, 0.32495412), (0, 0.2517927), (16, 0.2035881), (39, 0.15892276), (1, 0.15860835), (2, 0.15319394), (23, 0.1492914), (28, 0.13930418), (7, 0.13395755), (38, 0.1306662), (21, 0.12759909), (11, 0.12671755), (10, 0.11984646), (4, 0.11808879), (12, 0.11706242), (36, 0.11225772), (32, 0.11185029), (22, 0.1102775), (18, 0.11000852), (9, 0.10819475), (20, 0.098557636), (31, 0.097484015), (17, 0.097304612), (6, 0.097165346), (14, 0.093365625), (24, 0.089690089), (15, 0.074581966), (37, 0.074058898), (13, 0.072470389), (26, 0.070574328), (8, 0.07048057), (34, 0.068108737), (35, 0.063825652), (30, 0.060505763), (27, 0.060231414), (33, 0.058656305), (25, 0.056212574), (29, 0.021610398)]
QID 08	[0, 1, 2, 3, 4] / 10	[(9, 0.49663076), (1, 0.48820069), (3, 0.44288301), (8, 0.3661944), (4, 0.29680485), (7, 0.25400308), (2, 0.20772347), (6, 0.17150182), (0, 0.14081953), (5, 0.14036082)]
QID 42	[0, 1] / 17	[(0, 0.2033841), (5, 0.15258984), (14, 0.11387057), (7, 0.11201076), (8, 0.10854912), (3, 0.10571022), (15, 0.10138006), (9, 0.092080407), (10, 0.086255223), (6, 0.085732765), (4, 0.070257537), (2, 0.070151441), (1, 0.066130333), (12, 0.065770842), (13, 0.05880022), (11, 0.055507738), (16, 0.04612254)]
QID 72	[0, 1, 2, 3, 4, 5, 6] / 27	[(1, 0.17679131), (19, 0.17151187), (8, 0.1653154), (0, 0.14972456), (3, 0.13887556), (15, 0.11316334), (10, 0.11158201), (25, 0.099444613), (2, 0.095285937), (23, 0.094013408), (4, 0.089600556), (6, 0.084473729), (21, 0.079430752), (5, 0.075068399), (7, 0.073358811), (24, 0.07202708), (12, 0.066801175), (11, 0.054601047), (13, 0.052626666), (16, 0.049960222), (20, 0.042469479), (17, 0.042257264), (18, 0.041249175), (9, 0.033214726), (26, 0.032299262), (14, 0.02982882), (22, 0.019049376)]
QID 47	[0, 1, 2, 3, 4] / 25	[(3, 0.22970209), (9, 0.20223591), (17, 0.17326722), (1, 0.16343944), (0, 0.15195489), (15, 0.14825673), (7, 0.13658166), (13, 0.12968345), (12, 0.12456912), (2, 0.12170098), (8, 0.097534224), (10, 0.088660985), (21, 0.085786529), (19, 0.082761407), (18, 0.071183316), (4, 0.069002368), (24, 0.067095183), (16, 0.065536283), (11, 0.056499857), (6, 0.05196701), (22, 0.047492556), (23, 0.044300023), (5, 0.042080328), (14, 0.03628223), (20, 0.026847236)]

QID	QUESTION
77	What is the name of the first space shuttle ?
68	What rank did Nimitz reach ?
00	Who established the Nobel prize awards ?
63	What is Florence Nightingale famous for ?
07	Where was Carlos the Jackal born ?
34	During what war did Nimitz serve ?
08	What is Cassini 's destination ?
42	How many people did Jack Welch fire from GE ?
72	When was the Good Friday agreement made ?
47	Who founded Public Citizen ?

TABLE A.7: Worst performing Average Precision Questions

QID	Correct Answer/Total	Predicted (index,score)
QID 91	[0, 1]/12	[(6, 0.51424253), (7, 0.26575568), (9, 0.22294578), (8, 0.2224436), (5, 0.18405184), (0, 0.15018585), (4, 0.12420429), (1, 0.10463461), (3, 0.095704749), (10, 0.086780258), (2, 0.084115393), (11, 0.062696956)]
QID 44	[0, 1, 2, 3, 4]/36	[(25, 0.30446875), (24, 0.2957021), (11, 0.22827786), (15, 0.22697222), (4, 0.19664629), (5, 0.19368051), (9, 0.17344275), (29, 0.15023147), (3, 0.14939511), (19, 0.14804108), (22, 0.14453492), (34, 0.14099748), (14, 0.13100594), (12, 0.12891552), (10, 0.12464461), (0, 0.11827713), (23, 0.11211324), (32, 0.10767436), (1, 0.10608303), (20, 0.10539795), (17, 0.094127052), (33, 0.092531189), (27, 0.091022454), (2, 0.087970681), (7, 0.08769764), (26, 0.083654083), (18, 0.083088256), (8, 0.081103027), (6, 0.074284948), (35, 0.068236388), (31, 0.065872028), (16, 0.06203429), (30, 0.061884467), (13, 0.052000344), (28, 0.049019828), (21, 0.035415787)]
QID 64	[0]/6	[(4, 0.1513934), (1, 0.12041572), (2, 0.10370769), (5, 0.090300187), (0, 0.087630019), (3, 0.078876071)]
QID 88	[0, 1, 2]/40	[(4, 0.43096325), (7, 0.41627389), (29, 0.4155153), (16, 0.31130356), (26, 0.30271634), (31, 0.2872816), (17, 0.26988119), (3, 0.2546933), (11, 0.24681313), (2, 0.19581246), (1, 0.18925819), (34, 0.18546687), (33, 0.17164603), (21, 0.16949612), (0, 0.16341184), (37, 0.15911512), (38, 0.15485841), (24, 0.14181215), (6, 0.1354299), (32, 0.13378961), (15, 0.11750252), (10, 0.11707997), (20, 0.11633051), (14, 0.11479139), (8, 0.10855308), (22, 0.10152553), (19, 0.10122345), (30, 0.09829212), (27, 0.096361682), (9, 0.096263379), (35, 0.091147907), (13, 0.090090372), (39, 0.079035774), (25, 0.071828008), (36, 0.066080995), (5, 0.064771041), (28, 0.037373077), (23, 0.033515126), (18, 0.026611172), (12, 0.022841142)]
QID 22	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]/112	[(111, 0.39622539), (94, 0.35735852), (76, 0.33942282), (105, 0.32984677), (73, 0.31931758), (84, 0.30971786), (101, 0.30689132), (100, 0.29772165), (31, 0.29534575), (108, 0.28817537), (6, 0.27833399), (72, 0.27823901), (61, 0.26362342), (0, 0.25617996), (95, 0.24855058), (74, 0.24603605), (71, 0.23666468), (81, 0.22848758), (98, 0.22749893), (87, 0.22452971), (9, 0.21198091), (49, 0.20700419), (32, 0.20038633), (48, 0.19959731), (69, 0.19895539), (2, 0.19525181), (15, 0.19215223), (22, 0.19183397), (43, 0.18745902), (8, 0.18054263), (60, 0.17660789), (4, 0.17648721), (3, 0.17516153), (27, 0.17468293), (64, 0.17456762), (40, 0.1723592), (13, 0.17173445), (109, 0.16911267), (107, 0.16835012), (45, 0.16264978), (86, 0.16112128), (96, 0.15916461), (7, 0.15644628), (12, 0.15373747), (89, 0.15069002), (10, 0.14867908), (39, 0.14139457), (110, 0.14099406), (41, 0.13679875), (46, 0.13309154), (28, 0.13224714), (11, 0.12963402), (92, 0.12877116), (51, 0.12744038), (91, 0.12733428), (82, 0.12655878), (34, 0.12423309), (75, 0.1237492), (59, 0.11983083), (36, 0.11896742), (30, 0.11667002), (58, 0.11389365), (93, 0.11374056), (63, 0.1119544), (37, 0.11183269), (20, 0.10981183), (50, 0.10938602), (65, 0.1087577), (44, 0.10849365), (23, 0.10794173), (77, 0.10676094), (55, 0.10508985), (66, 0.10117453), (26, 0.10082562), (54, 0.097826906), (24, 0.095744066), (53, 0.088575624), (1, 0.088462554), (68, 0.088110864), (62, 0.085427955), (18, 0.085408345), (38, 0.082858488), (16, 0.081986159), (56, 0.08058726), (57, 0.080240287), (21, 0.079425238), (103, 0.07936693), (83, 0.078951627), (19, 0.077677965), (104, 0.077344581), (99, 0.074456297), (14, 0.072997674), (25, 0.072701871), (80, 0.072656281), (106, 0.071990103), (67, 0.069955692), (79, 0.066650078), (85, 0.066201776), (29, 0.066041529), (78, 0.062945627), (42, 0.059435192), (5, 0.058825165), (97, 0.055165652), (88, 0.051920492), (70, 0.049155153), (52, 0.048914868), (33, 0.046777032), (47, 0.04468954), (90, 0.044153158), (102, 0.042775515), (17, 0.039051138), (35, 0.036698818)]
QID 12	[0, 1, 2, 3]/41	[(38, 0.44437259), (5, 0.23716372), (7, 0.21615462), (39, 0.20574072), (31, 0.16477677), (14, 0.1641039), (2, 0.13165942), (21, 0.12525231), (4, 0.11812441), (15, 0.11580469), (13, 0.11336216), (1, 0.10983095), (37, 0.10100541), (26, 0.097956076), (10, 0.097373381), (17, 0.091710202), (20, 0.089369372), (22, 0.083839424), (25, 0.079643317), (33, 0.071176745), (12, 0.066736452), (29, 0.065657385), (30, 0.064869717), (28, 0.061642013), (11, 0.061559308), (24, 0.060924977), (16, 0.054890841), (34, 0.053100016), (6, 0.051927462), (3, 0.046175677), (32, 0.045483608), (8, 0.044654153), (35, 0.043433312), (9, 0.042459227), (27, 0.039555073), (0, 0.038767412), (19, 0.037157439), (40, 0.036933023), (23, 0.033678461), (36, 0.031973038), (18, 0.030539772)]
QID 75	[0]/13	[(10, 0.42372677), (11, 0.27831373), (6, 0.27294871), (2, 0.23799859), (12, 0.17963657), (8, 0.1752606), (9, 0.1648623), (4, 0.13905936), (0, 0.12569158), (1, 0.11205045), (3, 0.10288331), (7, 0.06407208), (5, 0.063856177)]
QID 80	[0, 1, 2]/49	[(45, 0.31531617), (41, 0.31136137), (23, 0.29528001), (13, 0.27643901), (20, 0.2715252), (0, 0.2710571), (21, 0.26125664), (8, 0.25823587), (25, 0.2519505), (18, 0.2468344), (26, 0.24464528), (38, 0.22642632), (42, 0.22531241), (43, 0.22169477), (3, 0.19858271), (31, 0.19430687), (28, 0.18824796), (24, 0.1860732), (36, 0.17430586), (37, 0.17385294), (27, 0.14743081), (33, 0.14590468), (5, 0.1436116), (32, 0.1377226), (22, 0.13686672), (2, 0.13664442), (40, 0.13632286), (12, 0.13581684), (30, 0.1284588), (48, 0.11442585), (46, 0.11404914), (47, 0.10718688), (16, 0.1054408), (4, 0.10060015), (19, 0.10058251), (6, 0.099156946), (1, 0.097464934), (34, 0.096908793), (15, 0.09481246), (7, 0.091784596), (17, 0.080135725), (44, 0.06032161), (11, 0.058391247), (14, 0.057896014), (39, 0.057734959), (9, 0.057001449), (10, 0.056342885), (35, 0.053693946), (29, 0.031570993)]
QID 39	[0]/15	[(7, 0.39046669), (4, 0.32164842), (2, 0.29179454), (3, 0.25334534), (10, 0.23921233), (5, 0.15645832), (6, 0.13916998), (11, 0.12959106), (9, 0.12855716), (8, 0.10962581), (1, 0.097616412), (13, 0.052610945), (0, 0.04536406), (12, 0.041479435), (14, 0.018947836)]
QID 14	[0]/21	[(17, 0.42696771), (8, 0.41752681), (2, 0.31228122), (14, 0.30945441), (13, 0.3052133), (5, 0.27481681), (18, 0.24372944), (3, 0.2284424), (19, 0.20411402), (16, 0.19819871), (9, 0.19544232), (15, 0.17665899), (7, 0.14718522), (0, 0.14046802), (4, 0.12906809), (1, 0.11694164), (20, 0.10161125), (11, 0.095066614), (6, 0.077121213), (10, 0.069124661), (12, 0.068101324)]

QID	QUESTION
91	What is the name of the company Vilar founded ?
44	When did Jack Welch become chairman of General Electric ?
64	What rank did Eileen Marie Collins reach ?
88	What year did the Teapot Dome scandal take place ?
22	When did the Khmer Rouge come into power ?
12	When did Amtrak begin operations ?
75	Who founded the Muslim Brotherhood ?
80	Who was President of the United States in <num> ?
39	What is the religious affiliation of the Kurds ?
14	Where are the company Conde Nast 's headquarters ?

Bibliography

[1] Ferrucci, David. "Build Watson: an overview of DeepQA for the Jeopardy! challenge." Parallel Architectures and Compilation Techniques (PACT), 2010 19th International Conference on. IEEE, 2010.

[2] Fan, James, et al. "Automatic knowledge extraction from documents." IBM Journal of Research and Development 56.3.4 (2012): 5-1.

[3] Ferrucci, David, et al. "Building Watson: An overview of the DeepQA project." AI Magazine 31.3 (2010): 59-79.

[4] Bilotti, Matthew W., Boris Katz, and Jimmy Lin. "What works better for question answering: Stemming or morphological query expansion." Proceedings of the Information Retrieval for Question Answering (IR4QA) Workshop at SIGIR. Vol. 2004. No. 1.2. 2004.

[5] Hirschman, Lynette, and Robert Gaizauskas. "Natural language question answering: the view from here." natural language engineering 7.4 (2001): 275-300.

[6] Porter, Martin F. "An algorithm for suffix stripping." Program 14.3 (1980): 130-137.

[7] Popovic, Mirko, and Peter Willett. "The effectiveness of stemming for natural-language access to Slovene textual data." Journal of the American Society for Information Science 43.5 (1992): 384.

[8] Krovetz, Robert. "Viewing morphology as an inference process." Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1993.

[9] Hull, David A. "Stemming algorithms: A case study for detailed evaluation." JASIS 47.1 (1996): 70-84.

[10] Harman, Donna. "How effective is suffixing?." Journal of the American society for information science 42.1 (1991): 7.

[11] Berger, Adam, et al. "Bridging the lexical chasm: statistical approaches to answer-finding." Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2000.

[12] Colmerauer, Alain, and Philippe Roussel. "The birth of Prolog." History of programming languages—II. ACM, 1996.

[13] Lehmann, Jens, et al. "DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia." Semantic Web 6.2 (2015): 167-195.

[14] Prud, Eric, and Andy Seaborne. "SPARQL query language for RDF." (2006).

[15] Bao, Junwei, et al. "Knowledge-based question answering as machine translation." Cell 2.6 (2014).

[16] Rosenblatt, Frank. x. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan Books, Washington DC, 1961

[17] Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation". David E. Rumelhart, James L. McClelland, and the PDP research group. (editors), Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundation. MIT Press, 1986.

[18] Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." Neural networks 2.5 (1989): 359-366.

[19] Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." Neural computation 18.7 (2006): 1527-1554.

[20] Smolensky, Paul. Information processing in dynamical systems: Foundations of harmony theory. No. CU-CS-321-86. COLORADO UNIV AT BOULDER DEPT OF COMPUTER SCIENCE, 1986.

[21] Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." science 313.5786 (2006): 504-507.

[22] Coates, Adam, Andrew Ng, and Honglak Lee. "An analysis of single-layer networks in unsupervised feature learning." Proceedings of the fourteenth international conference on artificial intelligence and statistics. 2011.

[23] Bengio, Yoshua. "Learning deep architectures for AI." Foundations and trends® in Machine Learning 2.1 (2009): 1-127.

[24] LeCun, Yann, and Yoshua Bengio. "Convolutional networks for images, speech, and time series." The handbook of brain theory and neural networks 3361.10 (1995): 1995.

[25] Hubel, David H., and Torsten N. Wiesel. "Receptive fields and functional architecture of monkey striate cortex." The Journal of physiology 195.1 (1968): 215-243.

[26] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

[27] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[28] Zhang, Xiang, Junbo Zhao, and Yann LeCun. "Character-level convolutional networks for text classification." Advances in neural information processing systems. 2015.

[29] Kalchbrenner, Nal, Edward Grefenstette, and Phil Blunsom. "A convolutional neural network for modelling sentences." arXiv preprint arXiv:1404.2188 (2014).

[30] Lee, Ji Young, and Franck Dernoncourt. "Sequential short-text classification with recurrent and convolutional neural networks." arXiv preprint arXiv:1603.03827 (2016).

[31] Collobert, Ronan, et al. "Natural language processing (almost) from scratch." Journal of Machine Learning Research 12.Aug (2011): 2493-2537.

[32] Bottou, Léon. "Online learning and stochastic approximations." On-line learning in neural networks 17.9 (1998): 142.

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

[34] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." EMNLP. Vol. 14. 2014.

[35] Dean, Jeffrey, et al. "Large scale distributed deep networks." Advances in neural information processing systems. 2012.

[36] Kingma, Diederik, and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[37] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747 (2016).

[38] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[39] LeCun, Yann, Ido Kanter, and Sara A. Solla. "Second order properties of error surfaces: Learning time and generalization." Advances in neural information processing systems. 1991.

[40] Hinton, Geoffrey. "Distributed Representations" Department of Computer Science, University of Toronto, http://www.cs.toronto.edu/ bonner/courses/2014s/csc321/lectures/lec5.pdf. 2014.

[41] Aphex34 (Own work) [CC BY-SA 4.0 (http://creativecommons.org/licenses/by-sa/4.0)], via Wikimedia Commons

[42] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of machine learning research 15.1 (2014): 1929-1958.

[43] Severyn, Aliaksei, and Alessandro Moschitti. "Learning to rank short text pairs with convolutional deep neural networks." Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2015.

[44] L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman. Deep learning for answer sentence selection. CoRR, 2014

[45] Socher, Richard, et al. "Recursive deep models for semantic compositionality over a sentiment treebank." Proceedings of the 2013 conference on empirical methods in natural language processing. 2013.

[46] Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, chap2, 2015

[47] Rubenstein, Herbert, and John B. Goodenough. "Contextual correlates of synonymy." Communications of the ACM 8.10 (1965): 627-633.

[48] Sahlgren, Magnus. "The distributional hypothesis." Italian Journal of Disability Studies 20 (2008): 33-53.

[49] Hofmann, Thomas. "Probabilistic latent semantic analysis." Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1999.

[50] Golub, Gene H., and Christian Reinsch. "Singular value decomposition and least squares solutions." Numerische mathematik 14.5 (1970): 403-420.

[51] Thomas K Landauer and Susan Dumais (2008) Latent semantic analysis. Scholarpedia, 3(11):4356.

[52] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

[53] Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig. "Linguistic regularities in continuous space word representations." hlt-Naacl. Vol. 13. 2013.

[54] Collobert, Ronan, and Jason Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning." Proceedings of the 25th international conference on Machine learning. ACM, 2008.

[55] Zhila, Alisa, et al. "Combining Heterogeneous Models for Measuring Relational Similarity." HLT-NAACL. 2013.

[56] Bengio, Yoshua, et al. "A neural probabilistic language model." Journal of machine learning research 3.Feb (2003): 1137-1155.

[57] Hinton, Geoffrey E. "Learning distributed representations of concepts." Proceedings of the eighth annual conference of the cognitive science society. Vol. 1. 1986.

[58] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.

[59] Morin, Frederic, and Yoshua Bengio. "Hierarchical Probabilistic Neural Network Language Model." Aistats. Vol. 5. 2005.

[60] Molino, Piero, et al. "Distributional Semantics for Answer Re-ranking in Question Answering." IIR. 2013.

[61] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." EMNLP. Vol. 14. 2014.

[62] Lund, Kevin, and Curt Burgess. "Producing high-dimensional semantic spaces from lexical co-occurrence." Behavior Research Methods, Instruments, & Computers 28.2 (1996): 203-208.

[63] Bai, Jing, et al. "Query expansion using term relationships in language models for information retrieval." Proceedings of the 14th ACM international conference on Information and knowledge management. ACM, 2005.

[64] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." Journal of machine Learning research 3.Jan (2003): 993-1022.

[65] Severyn, Aliaksei, and Alessandro Moschitti. "Learning to rank short text pairs with convolutional deep neural networks." Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2015.

[66] Chollet, François. "Keras (2015)." URL http://keras. io (2017).

[67] Abadi, Martín, et al. "TensorFlow: A System for Large-Scale Machine Learning." OSDI. Vol. 16. 2016.