Question Answering System based on Sentence Similarity

Mohammed Kashif
IIIT-D-MTech-CS

Submitted in partial fulfillment of the requirements
for the Degree of M.Tech. in Computer Science

to

Indraprastha Institute of Information Technology Delhi

June, 2017

Certificate

This is to certify that the thesis titled **"Question Answering System based on Sentence Similarity"** submitted by **Mohammed Kashif** to the Indraprastha Institute of Information Technology Delhi, for the award of the Master of Technology, is an original research work carried out by him under our supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree. The results contained in this thesis have not been submitted in part or full to any other university or Institute for the award of any degree/diploma.

Dr. Chetan Arora

Dept. of Computer Science

IIIT Delhi

Acknowledgements

I sincerely thank my supervisor Dr. Chetan Arora, IIIT Delhi for giving me the opportunity to work on this project and also his constant supervision and valuable guidance during the course of thesis.
A special thanks to my fellow, Sumit for making my thesis journey fun and memorable.
I would also like to thank my family for providing me the motivation, support and encouragement during the entire duration of my thesis.

Abstract

Question-Answering systems are becoming increasingly popular, especially after the advent of Machine Learning. Such systems can be used in a wide variety of applications ranging from Open domain question answering or closed domain question answering which involves searching for answering within a fixed domain. This thesis explores the idea of using sentence matching as a technique to answer question based on a corpus of text provided to us, (similar to that of machine comprehension).

In this thesis, we mainly focus on facts about Dr. A.P.J. Abdul Kalam as the corpus given to the system as input. Based on the Questions asked by the user, the system searches for correct facts that contain the relevant informaition. This work first explores corpus statistics as measure of sentence matching, followed by two different representation of word vectors to represent the sentences in the vector space,viz word2vec and fastText. Next, we explore Siamese Deep Learning Network to perform sentence matching. Finally we compare the results with sentence vectors captured using sentence2vec.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

The main objective of my thesis was that given a dataset of facts associated with Dr. A.P.J. Abdul Kalam, and given a question entered by the user into the system, the Top 3 related facts should be returned to the user. The Top 3 sentences from the corpus of related facts are decided based on their sentence similarity with the question itself.

1.2 Challenges

1.2.1 Current Machine Learning Systems

The intial challenge with developing this type of system was that the current machine learning techniques employed for question answering systems fall broadly into the two categories :

1. In the first type of system[2][10], given a given a corpus of atmost 500 characters and a question related to the corpus, the system would return a one word answer from the text for the question. The reason, we cannot use these kinds of systems for our problems are as follows:
    - The length of the corpus is limited to atmost 500 characters. Thus in effect, the corpus need to be modified according to the question in order to ensure that we get the most relevant fact as it may not even be present in the 500 characters.
    - These kinds of systems return only the relevant term from the corpus associated with the question and not the whole sentence.
    - It is difficult to answer questions like 'How did xyz happen' or 'What did he do to steal this' with single term as the answer, as the correct answer would require multiple sentences as the answer.
2. In the second type of system[11], the sentences need to be broken into very simple parts of the form '<subject> <action> <object>' . The system was correctly able to identify the answer as well as trace the exact sentence for the correct answer as well. It suffers from the following limiations :

- The sentence all follow the format '<subject> <action> <object>'. For example, Bob took the ball, Bob went to the kitchen, etc. Real world facts cannot always follow the same format.
- Since the whole system was based on the assurance that sentences follow a certain format, we cannot use any of the modules for own problem (even the sentence similarity one) , since the sentences in our corpus did not follow this format.

These systems are discussed in more detail in Section 2. Now since the existing question answering systems were tailor made for specific use cases, we could not use them for solving our problem, as our system required the following things:

- The corpus could be of any size
- The sentence in the corpus can follow any format
- It may be the case that the answer may span across more than one sentence, so the multiple sentences must be returned.
- The answer to a question cannot be always answered by a single term, and it might require the whole sentence to understand the context.

1.2.2 Dataset

There are multiple datasets available[13][8] for question answering system availble for use. However, all these are made specifically for the systems described above. So, in order to ensure the optimal performance of our system, we had to develop our own dataset. We collected text related to Dr. A. P. J. Abdul Kalam from multiple sources and re-organised the corpus into 300 sentences. A snapshot of the dataset is attached below:

1. abdul kalam was the 11th president of india
2. Abdul kalam was president of India from 2002 to 2007.
3. abdul kalam was born in rameswaram, tamil nadu.
4. abdul Kalam was raised in rameswaram, tamil nadu.
5. abdul Kalam was a career scientist turned statesman.
6. abdul kalam studied physics and aerospace engineering.
7. abdul kalam spent four decades as a scientist and science administrator.
8. Abdul kalam worked at the defence research and development organisation (drdo).
9. Abdul kalam worked at indian space research organisation (isro) .
10. abdul kalam was intimately involved in india's civilian space programme .
11. Abdul kalam was also involved in Indian military missile development efforts.
12. abdul kalam is also known as the missile man of india.
13. For his work on launch vehicle technology, abdul kalam was nicknamed missile man of india.
14. abdul kalam played a pivotal organisational, technical, and political role in india's pokhran-ii nuclear tests in 1998
15. Abdul kalam was elected as the 11th president of india in 2002.
16. Both the ruling BJP and the then opposition congress supported Kalam

Figure 1.1: Dataset

1.3 Outline

The rest of the thesis is organized as follows:

- In Section 2, we provide the related work and background to the problem. This section primarily discusses the literature review and current systems associated with the problem.
- In section 3, we discuss a technique which uses corpus statistics and WordNet to solve the problem.
- In section 4 and 5, we discuss the different word embeddings for text and how they can be used to solve the problem of sentence matching.
- In section 6, we discuss the application of Siamese neural network to our problem.
- Section 7 provides the approach for sentence embedding.
- Section 8 provides experimental results and analysis.
- Finally in Section 9, we conclude the report.

Chapter 2

Related Work

In this chapter we will discuss the various systems used for Question Answering

2.1 QA systems

2.2 Questions with Supporting Facts

Recently Facebook released the babi dataset[13], which consists of 20 toy tasks. All tasks are in the form of context, question and answer triplets. The main purpose of this dataset is to ensure that the system learns as many different tasks as possible. It is a synthetic dataset, consisting of an actor and simulated environment.

```
1 Mary moved to the bathroom.
2 John went to the hallway.
3 Where is Mary?        bathroom        1
4 Daniel went back to the hallway.
5 Sandra moved to the garden.
6 Where is Daniel?      hallway 4
7 John moved to the office.
8 Sandra journeyed to the bathroom.
9 Where is Daniel?      hallway 4
10 Mary moved to the hallway.
11 Daniel travelled to the office.
12 Where is Daniel?     office  11
13 John went back to the garden.
14 John moved to the bedroom.
15 Where is Sandra?     bathroom        8
1 Sandra travelled to the office.
2 Sandra went to the bathroom.
3 Where is Sandra?      bathroom        2
```

Figure 2.1: baBi Dataset

End-to-End Memory networks [11]achieved state of the art results on the baBi dataset, however it did not work well in real world environment. The reason for this was, any network trained over this dataset generalised itself over the simplified set of tasks. Since the sentences were very simple in nature, mainly consisting of the form <actor> <action> <object>, any system trained over this dataset could not handle real world sentences. Hence, models trained over this dataset cannot be used for our problem.

2.3 SQuAD Dataset

The Stanford Question Answering Dataset (SQuAD)[9] is one the largest datasets available for QA systems. It contains over 100,000 questions based on over 500+ Wikipedia articles. The dataset is of the following format : Currently the systems that achieve state of the art results

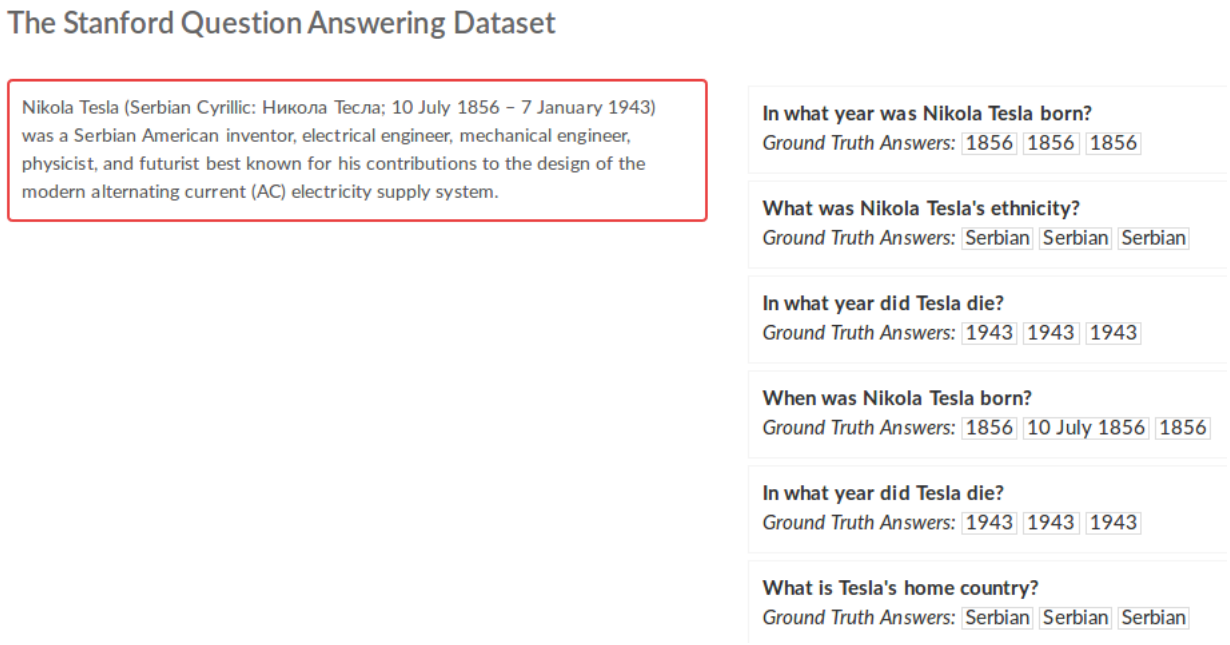


Figure 2.2: SQuAD Dataset

on this dataset include r-net[12], Resonet[10] and Neumonic Reader[2]. However, the systems trained on this dataset cannot be used for our problem because of the following reasons:

- The corpus is limited to at max 500 characters.
- The answer to a question can span across multiple lines. Hence, all those line should be included in the results, which is not the case in these systems.

Chapter 3

Semantic Nets and Corpus Statistics

3.1 Introduction

Sentence Similarity measures are widely used in text-related research, text-mining, Web page retrieval and dialogue systems. Earlier methods uses long text documents to calculate sentence similarity. It processes the sentence in very high-dimensional space thus these methods are inefficient and cannot be adopted for many application domains. This approach[3] focuses on the sentence similarity between short texts with text of sentence length. It makes use of lexical database and corpus statistics in order to find out the semantic similarity between the two sentences. The purpose of lexical database is to incorporate human common sense knowledge while corpus statistics makes the model to be adaptable for different domains.

3.2 Proposed Text Similarity Method

The proposed method makes use of both semantic as well as syntactic information from the compared texts. Here, the text is considered of sentence length. The above method performs following steps in order to calculate sentence similarity:

- Dynamically form a joint word set which consist of distinct words from both the sentences.
- A raw semantic vector is calculated for both the sentences using Lexical database.
- A word order vector is calculated using the Lexical database.
- As each word contributes differently in the meaning of sentence, each word is weighted using the corpus statistics.
- Combine the raw semantic vector along with the information from corpus calculate the semantic vector for both the sentences.
- Semantic similarity is then calculated by using both the semantic vectors calculated in above step.

- Order vectors are used to calculate the order similarity and finally sentence similarity is calculated by combining both the order similarity and semantic similarity.

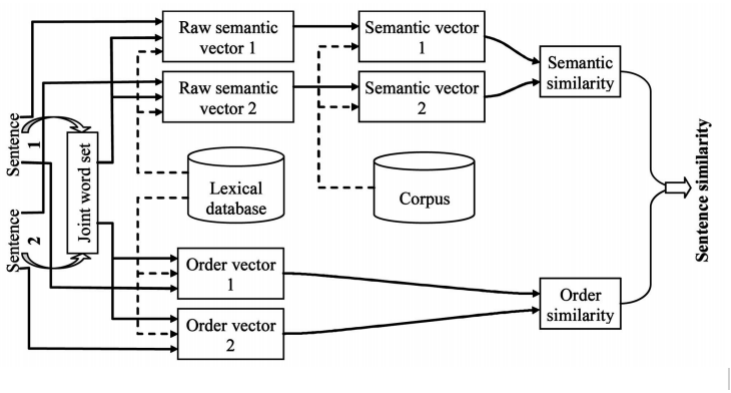


Figure 3.1: Text similarity using Corpus statistics

3.3 Semantic Similarity between Words

As Semantic Similarity is used in both the sentence semantic similarity and word order similarity this section will discuss semantic similarity is calculated.

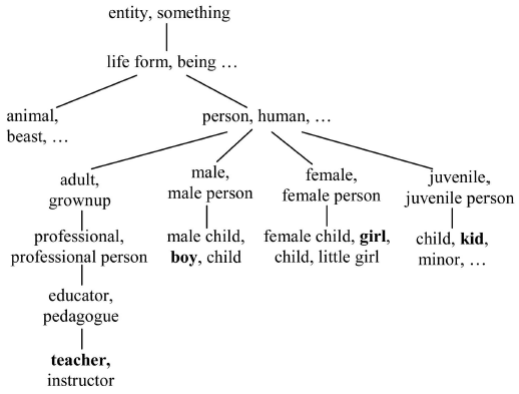


Figure 3.2: Semantic Similarity between Words

The hierarchical structure for knowledge base is used to determine the semantic distance between words. WordNet[5] is the lexical knowledge base used here. In order to calculate the semantic similarity between two words w1 and w2 is s(w1,w2). The similarity s(w1,w2) between words is consider as the function of path length as well as depth:

$$S(w1, w2) = f(l, h) \tag{3.1}$$

Where, l is the shortest path between the words w1 and w2 and h is the depth in the hierarchical semantic nets. Here, the above equation can be written two independent functions

as:

$$S(w1, w2) = f1(l)f2(h) \tag{3.2}$$

Here f1 and f2 are called transfer functions of path length and depth respectively.

3.3.1 Contribution of Path Length

The path length between the two words can be calculated by one of the following cases:

- If w1 and w2 belong to the same synset, which means they convey the same meaning and hence path length is 0.
- If w1 and w2 does not belong to the same synset but share some common words, for example in fig 2, the boy and girl share a common word child. In this case we set the path length to 1.
- If w1 and w2 does not belong to the same system and neither shares a common word then we calculate the actual path length between w1 and w2.

Considering above cases we define f1(l) as follows:

$$f1(l) = e^(\alpha l) \tag{3.3}$$

Where α is a constant. This makes f1(l) to be a monotonically decreasing function of l.

3.4 Scaling Depth Effect

The words at top layer of hierarchical semantic nets have less semantic similarity than words at lower layer. In order to incorporate this fact in s(w1,w2), we scale down s(w1,w2) for words at upper layer and scale up for the words at lower layers. Thus we can write f2(h) as:

$$f2(h) = \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \tag{3.4}$$

Where β>0 is a smoothing factor. Thus the formula for the word similarity can be written as:

$$s(w1, w2) = e^{-\alpha l} . \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \tag{3.5}$$

The optimum values of and are dependent on the knowledge based used, here the knowledge base is WordNet and the optimum values for this measure is $\alpha = 0.2$ and $\beta = 0.45$.

3.5 Semantic Similarity between Sentences

For sentences T1 and T2 we calculate the joint word set as follows:

$$T = T1 \bigcup T2 \tag{3.6}$$

T contains all the distinct words from T1 and T2. As joint word set is derived from the T1 and T2 it does not contain any redundant information. Lexical semantic vector is calculated from the joint word set and is represented by $\hat{s}$ and size is equal to the size of joint word set. The value at any index of the lexica semantic vector is calculated as follows:

- If w_i is present in T1 then $\hat{s_i}$ is set to 1.
- If w_i is not present in T1 then we calculate semantic similarity score between w_i and every word in T1. So the most similar word in T1 to w_i is the one with highest similarity score $\hat{s}$. We set a threshold then we set $\hat{s_i} = \hat{s}$ otherwise set to 0.

Different words contribute to the meaning of the sentence in different degrees. In order to associate weight to every word we use information content. It has been showed that the word with higher frequency in the corpus contain less information than the words occur with less frequencies. Each word is associated with the information $l(w_i)$ and $l(\hat{w_i})$. So the value of semantic vector can be written as:

$$S_i = \hat{s}.l(w_i).l(\hat{w_i}) \tag{3.7}$$

Where w_i is the word in the joint word set, $\hat{w_i}$ is the its associated word in the sentence. Thus the semantic similarity between two sentences is defined as cosine coefficient between two vectors as follows:

$$S_s = \frac{s_1.s_2}{\|s_1\|.\|s_2\|} \tag{3.8}$$

3.6 Word Order Similarity between Sentences

In order to capture the dissimilarity between the sentences on the basis of the ordering of words in the sentences Word Order vector is calculated. In order to find out the Word Order Similarity, word order vector r is calculated for each sentence T1 and T2 respectively, by using the joint word set T. For example in order to calculate r1 for T1 we perform the following steps:

- If the word is present in the T1 we fill the index in r1 with the index of the word in T1.
- If not present then we try to find out the most similar word in T1, if the similarity is greater than a threshold than we fill the index in r1 with that word.
- Otherwise we fill the entry in r1 with 0.

$$S_r = 1 - \frac{\|r_1 - r_2\|}{\|r_1 + r_2\|} \tag{3.9}$$

$$S_r = 1 - \frac{k}{\sqrt{2\|r\|^2 - k^2}} \tag{3.10}$$

3.7 Overall Sentence Similarity

Overall Sentence Similarity consists of both the sentence similarity as well as word order similarity. Thus we can calculate the sentence similarity between two sentences T1 and T2 as follows:

$$S(T_1, T_2) = \delta.S_s + (1 - \delta)S_r \tag{3.11}$$

$$S(T_1, T_2) = \delta.\frac{s_1.s_2}{\|s_1\|.\|s_2\|} + (1 - \delta).\frac{\|r_1 - r_2\|}{\|r_1 + r_2\|} \tag{3.12}$$

Chapter 4

word2vec

4.1 Introduction

Word2Vec[4] is a neural network model that provides us a vector for each word in the corpus that encodes the semantic information of the word. It provides the vectors such that similar (synonyms) words tend to have similar vectors while dissimilar (antonyms) have dissimilar vectors. Word2Vec has two main models:

1. Skip-gram Model
2. CBOW Model

CBOW works on the model that predict a word given a context, a context can be a sentence or group of words. While Skip-Gram does the opposite of CBOW, it gives us the context of the word.

4.2 Skip-gram Model

The input of this model is a word w, which provides us with the output of the form w1....wc, where C is the word window size. For example, given a sentence "I play basketball every day in evening" and the input word is "basketball", which will give "I","play","every","day","in","evening" as the output. The output words are represented as one-hot encoding, they are vectors of length V where V is the size of our vocabulary. The vector has one for the index corresponding the word and zeros at other indices.

4.3 Forward Propogation

The skip-gram model consist of three layers namely, Input Layer which is represented by x in the one-hot encoded vector corresponds to input word. The output layer consist of vectors y1.....yC, these are the one-hot encoded vector for the output words from the training instance. Between the input layer and hidden layer there is a matrix W, which is the weight matrix of size V X N. The ith row of W represents the weight of the ith word in the vocabulary.

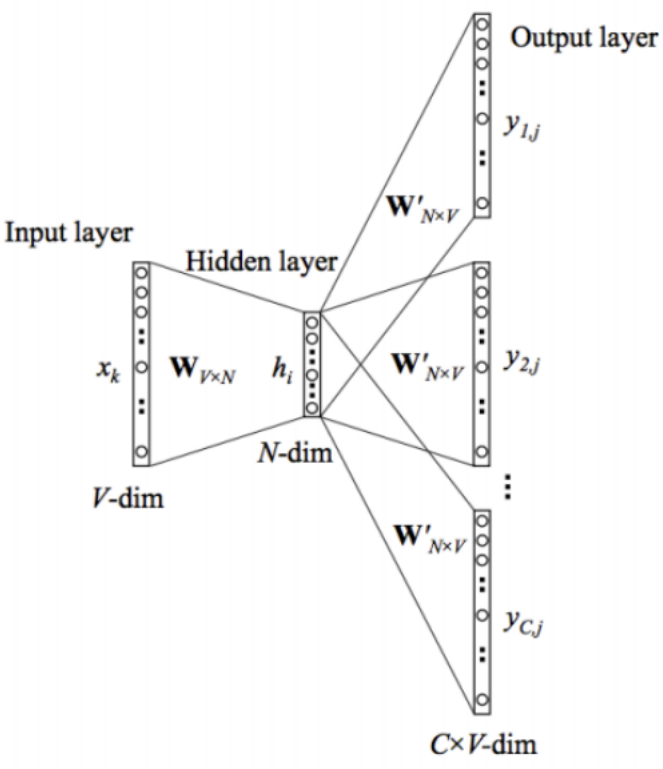


Figure 4.1: word2vec : Skip-gram Model

There is a matrix associated with the output word also represented as W' of size NXV. The hidden layer consists of N nodes where N is the training parameter. Let's consider the input layer and hidden layer. The input vector x is the one-hot encoded vector which has value either one or zero at the indices. Therefore, only the non zero values will contribute to the hidden layer. If kth index has value one in x then the output of the hidden layer will be kth row of matrix W. mathematically it can be represented as:

$$h = x^T W \tag{4.1}$$

The input to the second layer which is of dimension C X V is calculated by the weighted sum of its input. So, the input to the jth node with cth output word can be written as:

$$u_c, j = v'^T h \tag{4.2}$$

where v' , is the output vector for the jth word in the vocabulary.

In order to calculate the output of the jth node of the cth output word we uses a softmax function which can be mathematically represented as follows:

$$p(w_c, j = w_O, c|w_I) = y_c, j = \frac{\exp u_c, j}{\sum_{j=1}^{V} \exp u_j} \tag{4.3}$$

It represents the probability that our prediction for the j^{th} word in the c^{th} panel $w_{\text{c,j}}$ is equal to the actual output word at c^{th} location given the input word w_I.

4.4 Learning Weights with Backpropagation and Stochastic Gradient Descent

As we discussed in the above section how input is propagated forward in the network, now we can derive the gradient descent using Backpropagation to calculate w and w'.

$$E = -\log p(w_{O,1}, w_{O,2}, .., w_{O,C}|w_I) \tag{4.4}$$

$$E = -\log \prod_{c=1}^{C} \frac{\exp\left(u_{c,j}\right)}{\sum_{j'=1}^{V} \exp\left(u_{j'}\right)} \tag{4.5}$$

$$E = -\sum_{c=1}^{C} u_{j_c} + C.\log \sum_{j'=1}^{V} \exp\left(u_{j'}\right) \tag{4.6}$$

The above equation represents the loss function, which implies the probability of the words in the output vector corresponds to the input word. Taking the partial derivative w.r.t $u_{c,j}$

$$\frac{\partial E}{\partial u_{c,j}} = y_{c,j} - t_{c,j} \tag{4.7}$$

After this we can find out the partial derivative with respect to output matrix $W^{'}$

$$\frac{\partial E}{\partial w_{ij}^{'}} = \sum_{c=1}^{C} \frac{\partial E}{\partial u_{c,j}} . \frac{\partial u_{c,j}}{\partial w_{ij}^{'}} \tag{4.8}$$

$$\frac{\partial E}{\partial w_{ij}^{'}} = \sum_{c=1}^{C} (y_{c,j} - t_{c,j}).h_i \tag{4.9}$$

So, gradient descent for the output matrix can be written as follows:

$$w_{ji}^{'(new)} = w_{ji}^{'(old)} - \eta . \sum_{j=1}^{V} (y_{c,j} - t_{c,j}).h_i \tag{4.10}$$

Now we can compute the partial derivative with respect to hidden layer as follows:

$$\frac{\partial E}{\partial h_i} = \sum_{c=1}^{C} \frac{\partial E}{\partial u_j} . \frac{\partial u_j}{\partial h_i} \tag{4.11}$$

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \sum_{c=1}^{C} (y_{c,j} - t_{c,j}) . w_{ij}^{'} \tag{4.12}$$

After, calculating the partial derivative with respect to hidden layer we can calculate the derivative with respect to W.

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial e}{\partial h_i} . \frac{\partial h}{\partial w_{ki}} \tag{4.13}$$

$$\frac{\partial E}{\partial w_{ki}} = \sum_{j=1}^{V} \sum_{c=1}^{C} (y_{c,j} - t_{c,j}) . w_{i,j}^{'} . x_k \tag{4.14}$$

Lastly we can calculate the gradient descent for the input weights in the first layer of the model as follows:

$$w_{ji}^{'(new)} = w_{ji}^{'(old)} - \eta . \sum_{j=1}^{V} \sum_{c=1}^{C} (y_{c,j} - t_{c,j}) . w_{ij}^{'} . x_j \tag{4.15}$$

4.5 Word Mover's Distance

This technique is used to calculate the dissimilarity between the two text documents by calculating the minimum distance needs to travel to reach from embedded words of one document to the other. The BOW and TF-IDF are the common ways of representing the documents. These features are not appropriate to calculate the document distances as they do not capture the individual words. For example the sentences below:

1. Obama speaks to the media in Illinois.
2. The president greets the press in Chicago.

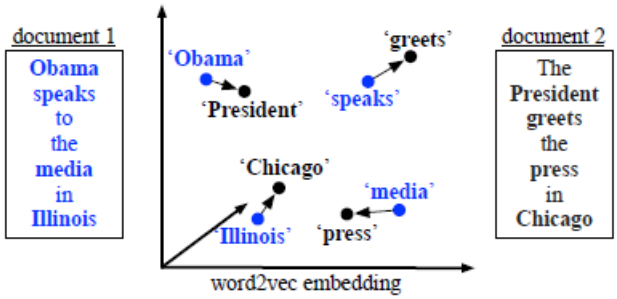


Figure 4.2: Word2Vec embeddings

As both the above sentences convey the same meaning but BOW distance model does not capture closeness between the sentences. The closeness between the word pairs like Obama-President , Illinois-Chicago which is not captured by the BOW-based distance.

In this method a new metric for the distance between text document is introduced. It uses word2Vec word embeddings and then calculate the minimum cumulative distance need to travel to match the words in each document.

The Figure 4.2 shows the distance between the word embeddings of two sentenes and thats how minimum cumulative distance is calculated.

Advantages of WMD distance measure are as follows:

1. It is hyper-parameter free.
2. It calculates the distances between the few words in the documents.
3. It makes use of word2vec embeddings which gives very high accuracy.

To calculate the WMD, word2vec embeddings are used. For a n words vocabulary, Matrix X $\varepsilon \mathrm{R}^{dxn}$ represents the word vector in d dimensional space.The vector is represented as a normalized Bag Of Words(nBOW) vectors, d εR^n.d_i is calculated as $\frac{c_i}{\sum_{j=1}^{n} c_j}$, where c_i is the number of times word i appears in the document. The distance between two word is calculated using Euclidean distance, c(i,j)= $||x_i - x_j||_2$.

In order to calculate the document distance, the documents are represent in nBOW vectors as d and $d^{'}$. Now, form a matrix T $\varepsilon R^{nxn} where \mathrm{T}_{ij}$ represent the distance between the word i and word j and $T_{ij} \geq 0$.Now to transform d entirely to $d^{'}$, the outgoing flow should be equal to the d_i for the word i, i.e $\sum_j T_{ij} = d_i$. Similarly for incoming flow for the word j should be $d_j^{'}$ i.e $\sum_i T_{ij} = d_j^{'}$. So the distance between the two documents can be formulated as $\sum_{i,j} T_{ij} c(i,j)$.

The transportation problem can be formed as a linear program as follows:

$$\min_{T \geq 0} \sum_{i,j=1}^{n} T_{ij} c(i,j)$$

$$\text{subject to: } \sum_{j=1}^{n} T_{ij} = d_i \quad \forall i \quad \varepsilon \quad \{1, ..., n\}$$

$$\sum_{i=1}^{n} T_{ij} = d_i^{'} \forall j \quad \varepsilon \quad \{1, ..., n\} \tag{4.17}$$

(4.17)

Chapter 5

FastText

5.1 Introduction

Recently there are many text classification models based on neural networks become increasingly popular. But these models are limited with their time to train and test on large datasets. FastText[1] works efficiently on large dataset with less train and test time. Most of the techniques do not take into consideration the internal structure of the words. There are many words that occur rarely so we can't make a good word level representation. FastText is the extension of the skip-gram model including the subword information.

5.2 Model

This model is derived from the skip gram model. The objective function of the skip gram model is represented as follows:

$$\sum_{t=1}^{T} \sum_{c \in \mathcal{C}_t} \log p(w_c \mid w_t) \tag{5.1}$$

where w_t is the input word, w_c is the context word for the given word and the context $\mathcal{C}_t$ is the set of indices of words surrounding w_t.

5.3 Subword Model

As discussed earlier skip gram do not take into consideration the internal structure of the words, FastText introduced a different scoring function s which take into consideration the internal structure of the words. The difference between skip gram and this model is that here each word also has the information of the n-grams along with the word itself. Let G_w denotes the set of n-grams for word w and Z_g represents the vectors associated with each n-gram g. So the scoring function can be represents as:

$$s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c. \tag{5.2}$$

One important point to note is that different vectors are assigned to a word and a n-gram that share the same sequence of characters. For intance consider the word *to* and the bigram *to*, appearing in the word *total*, will be assigned to different vectors. This simple model allows sharing the representations across words, thus allowing to learn reliable representation for rare words.

After calculating the scoring function we can find out the log likelihood for the for every word as follows:

$$\log(1 + e^{-s(w_t,\ w_c)}) + \sum_{n \in \mathcal{N}_{t,c}} \log\left(1 + e^{s(w_t,\ n)}\right), \tag{5.3}$$

where $\mathcal{N}_{t,c}$ is a set of negative examples sampled from the vocabulary.

For the purpose of the experiments, we use the pretrained vectors trained over Wikipedia (English)[7].

Chapter 6

RNN, LSTM and Siamese NN

6.1 Background

The problem of text understanding and information retrieval can be enhanced by finding the sentence similarity between sentences. Given two sentences, how likely they are similar or dissimilar is not a trivial task. A major challenge in the task of finding sentence similarity is finding semantic textual similarity. Other challenges are finding labelled data which scarce, sentences have complex structure and variable length. Method like bag of words/tf-IDF are limited in their cpntext. (c.f. Mihalcea, Corley, and Strapparava 2006).

Literature has proposed several methods for finding sentence similarity (in general). And naturally suited for sentences with variable length are Recurrent neural networks (RNN). A specific type of RNN, Long Short-Term Memory (LSTM) is well suited to learn from experience when there are unknown size and bound between the events (sentences in our case). In relation to neural networks, siamese neural network have been recently (2011) proposed for a number of metric learning tasks. They are a class of neural network architectures that contain two or more identical sub-networks.

In order to solve the problem of finding semantic sentence similarity, J. Mueller and A. Thyagarajan [6] proposed a siamese adaptation of Long Short-Term Memory (LSTM). We explore each of these techniques in the following sections to get an idea of their functionalities.

6.2 Recurrent Neural Networks

RNNS are a class of artificial neural network where connections between units form a directed cycle (source: Wiki). RNNs are used to make use of sequential information. In a traditional neural network, the assumption that all inputs (and outputs) are independent of each other, is not applicable to all applications. The main reason why recurrent nets are so special is that can work on sequence of vectors. Sequences could be in input, output, or both as well.

RNNs adapt standard feed-forward neural networks for sequence data $(x_1, \ldots, x_T)$, where at each t in $\{ 1, \ldots, T \}$, updates to a hidden-state vector h_t are performed via

$$h_t = sigmoid\ (W_{x_t} + U_{h_{t-1}}) \tag{6.1}$$

Here t represents a particular word in a text based input sequence. The input x_i in an RNN is one-hot vector representation of a word in a sequence.

$$x_i(\text{"}kalam\text{"}) = \begin{bmatrix} 0 \\ : \\ 1 \\ : \\ 0 \end{bmatrix} \quad where\ i\ in\ \{1, .., t\} \tag{6.2}$$

In equation 6.1, W represent the weight matrix for each input word and U represents the weight parameters for each hidden state vectors. One may notice that next hidden state vector in h_t is dependent on previous hidden state vector (h_{t-1}).

An output y_t for an RNN is represented by the equation 6.3. Here function f is also called activation functions and b represents the bias.

$$y_t = f\ (W_{y_{h_t}} + b_y) \tag{6.3}$$

Next we show some examples where RNNs can be used. Figure 6.1 shows various sequence types. A one-to-one mapping shown is a vanilla mode of processing. They have fixed-sized input to fixed-sized output. These function can work even without RNNS. An example of such type of sequences could be image classification.

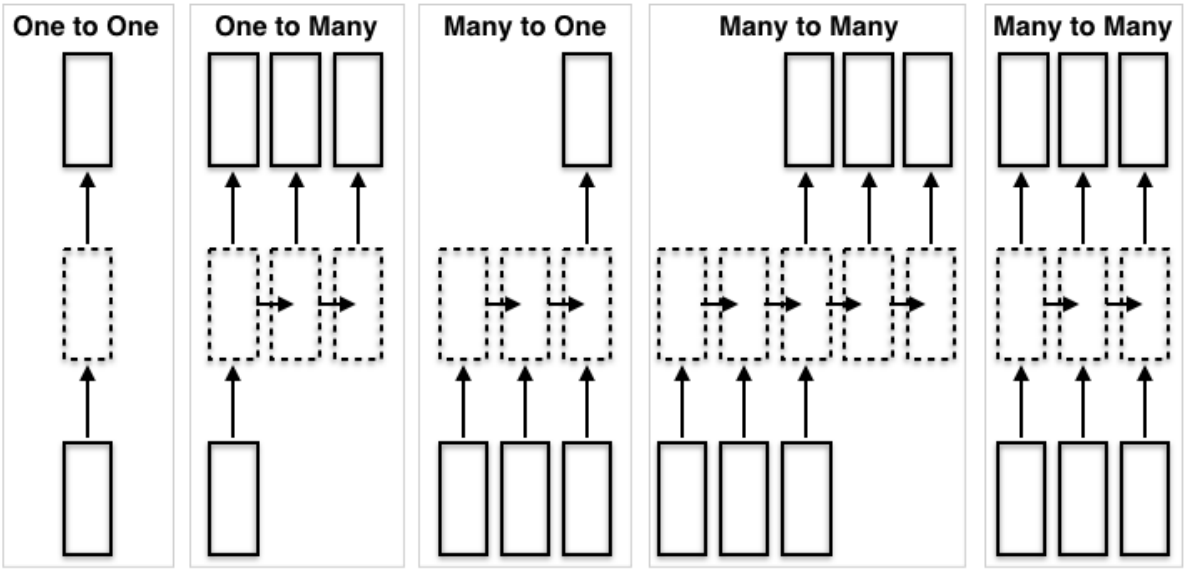
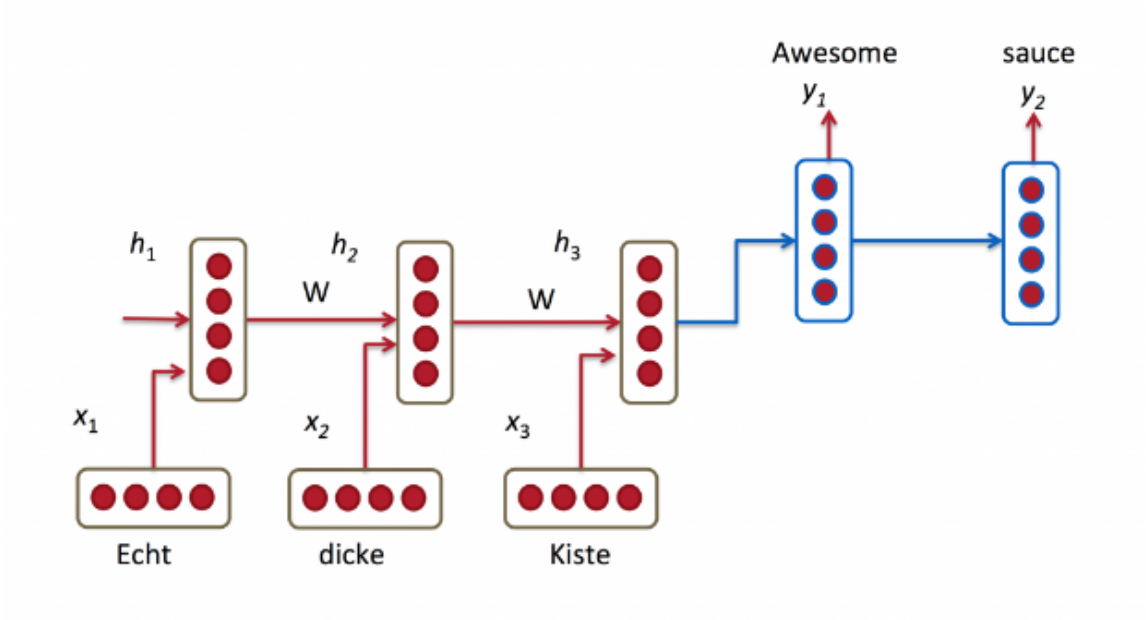


Figure 6.1: Types of sequences.

A one-to-many mapping shown is an example of sequenced output. These could be used in applications where image input could be a captioned image and output is a sentence of words. The many-to-one mapping shown is an example of sequenced input. We can consider an example of sentiment analysis where given a sentence, we have to classify weather this sentence is expressing positive or negative sentiments.



RNN for Machine Translation. Image Source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

Figure 6.2: An example of sequenced input and sequenced output.

There could be multiple types of many-to-many sequences. The many-to-one sequence shown on the left in the figure is an example of sequenced input and sequenced output. An application of such could be machine translation (shown in figure 6.2), where an input is in German language and output is in English language. Shown on the extreme right is an example of sequenced input and sequenced output. An application could be video classification, where we need to classify each frame of a video.

6.3 Long Short-Term Memory (LSTM)

Traditional RNNs have shown to be suffering from vanishing back-propagated gradients over long sequences. And this is where a type of RNN, Long short-term memory (LSTM) comes in place. An LSTM (ref:Wiki) is an artificial neural network containing LSTM units (there could additional other units as well in variants types of LSTM). An LSTM unit is recurrent network unit which uses no activation function within its recurrent components and this makes it excellent in remembering values for longer duration. Since there is no activation function, the stored value does not get squashed over time. Hence LSTM perform well in case of longer sequences when compared to traditional RNNs.

LSTMs are often implemented in blocks and each block contains 3 or 4 gates. the extent to which a new value flows into the memory and the extent to which it remains in the memory is controlled by an input and a forget gate. The output gate is for controlling the extent to which a a value in memory us used to compute the output activation of block. A memory state determines how it affects other states.

The updates in LSTM are performed in the following manner:

$$i_t = sigmoid\ (W_{ix_t} + U_{ih_{t-1}} + b_i) \tag{6.4}$$

$$f_t = sigmoid\ (W_{fx_t} + U_{fh_{t-1}} + b_f) \tag{6.5}$$

$$\hat{c_t} = tanh\ (W_{cx_t} + U_{ch_{t-1}} + b_c) \tag{6.6}$$

$$c_t = i_t \odot \hat{c_t} + f_t \odot c_{t-1} \tag{6.7}$$

$$o_t = sigmoid\ (W_{ox_t} + U_{oh_{t-1}} + b_o) \tag{6.8}$$

$$h_t = o_t \odot tanh\ (c_t) \tag{6.9}$$

Equations 6.4-6.9 represents the updates in each of the gate in an LSTM block. i_t represents the input gate, f_t represents the forget gate, c_t represents the memory state, o_t and h_t represents the output and the hidden state at t sequence. Here, W_k, U_k are respective weight parameters and b_k represents the bias for each k in $\{i, f, c, o\}$.

6.4 Siamese Networks

One can notice that an LSTM (or a traditional RNN) takes in a single sequence of an event (a sentence in our case) as input and produces some output. However sentence similarity require us to compare two sentences or sequence of words in order to find how similar or dissimilar the two set of sentences are to each other. Thus having just one sequence of input does not solve our problem. This is where Siamese Network comes in place. Siamese NN presents an architecture to compare two inputs or a pair of input.

Siamese neural network is a class of neural network architectures that contain two or more sub-networks having similar weight and parameters. Figure 6.3 shows a general architecture of siamese network. Given two inputs, a siamese network processes each of the input with a function with similar (or same) weight and other parameters. This could be any neural network system. Later, it process the outputs of each of these functions with some similarity (distance) measure to produce a target score or result.

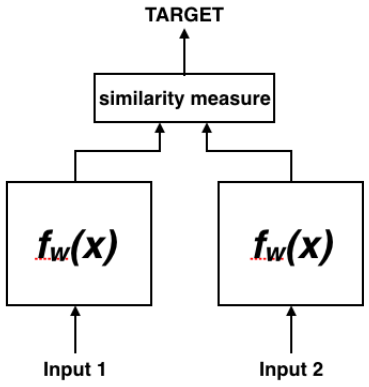


Figure 6.3: A high level architecture of siamese network.

6.5 Siamese Adaptation of LSTM

This idea is proposed in [6]. Figure 6.4 shows the architecture of Manhattan LSTM as proposed in [6]. Given two sentences

a) *He is smart*, and

b) *A truly wise man*;

Each of the words are converted into one hot vectors represented by x_i and feed forwarded into the hidden layer h_i of two LSTMs (a and b). Each of these two LSTM with tied weights produce output y_a (represented by $h_3^{(a)}$ in case of a) and y_b (represented by $h_4^{(b)}$ in case of b).

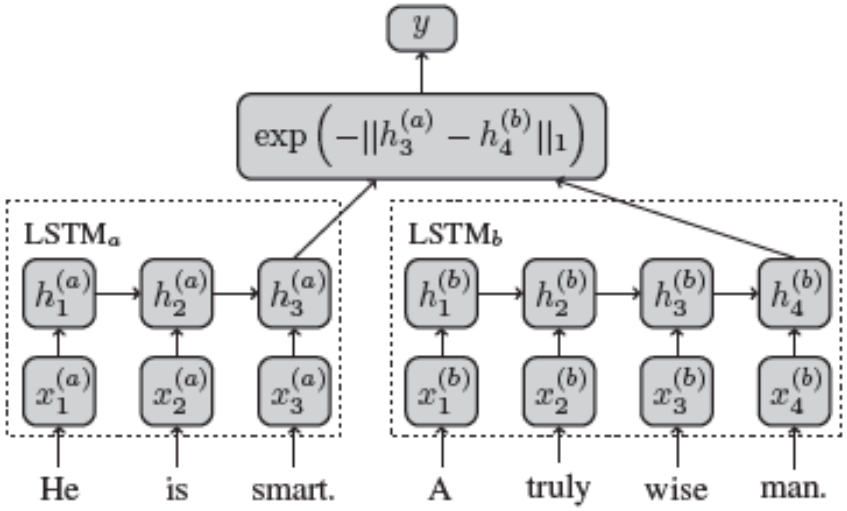


Figure 6.4: Manhattan LSTM - a siamese adaptation of LSTM. Source: []

A distance measure $exp(-||h_3^{(a)} - h_4^{(b)}||1)$ is used to get a similarity score. This score y represents the similarity or dissimilarity between the two sentences.

Chapter 7

Sentence2vec

7.1 Introduction

The motivation behind this came from the studies on word embedding using neural networks. The sentence embedding involves large piece of text as sentence or paragraph. This method involves unsupervised learning for calculating sentence embedding. This involves the word embedding on corpus like Wikipedia by using any popular methods for word embedding. Word embedding capture the similarities between the words. Then it makes use of weighted average of the words to represent the sentence. Then it modifies it by using SVD/PCA techniques. The technique used here can be summarized as follows:

1. Firstly, calculate the weighted average of the words that are present in the sentence.
2. After calculating the weighted average, remove the projections of the average vectors from the first principal component.
3. Here, weight of the word is given as a/a+p(w), where 'a' is the parameter and p(w) represents the word frequency. This is called SIF(Smooth Inverse Function).

There are few advantage of this technique over unweighted average and some RNN, LSTM models:

1. It is easily adaptable for different domains, as word vectors can be trained on different corpora for calculating the sentence embeddings.
2. This is a robust technique to the weighting scheme as using the word frequencies p(w) from different corpus does not impact the performance.
3. The parameter 'a' can be adjusted to get the best result.

The SIF reweighting handles the problem of frequent words which causes the average word vectors move towards meaningless direction.

7.2 Method for Sentence Embedding

This method is the improvement of the generative model described in (Arora et al. 2016). It is a dynamic process where t^{th} word is generated at t^{th} step. It uses the discourse vector c_t of dimension d for the random walk. Each word is represented by vector in d dimension. Then inner product is calculated between the word vector v_w of the word w and it is a time-invariant. It captures the correlation among the word and the discourse vector.

In this model it is assumed that the discourse vector ct does not have any change even if words in the sentence s are emitted, So we can replace the this vector by cs which is a single discourse vector for the sentence s in place of all the $c_t's$.

This model handles the following problem by introducing two smoothing term.

1. It handles the word that occurs out of context.
2. It handles the problem of frequent words like "the", "and".

Firstly it introduces the term $\alpha p(w)$ as an additive term in the log linear model where α is a scalar and $p(w)$ is the uni-gram probability of the word in the corpus.It handles the case when the word has very low inner product with c_s so those words can occur too. A common discourse vector c_0 is introduced with dimension d which represents the discourse related to syntax. It increases the probability for the words to co-occur with high component along the discourse vector c_0.

The probability for the word w can be modeled given the discourse vector c_s and the sentence as follows:

$$Pr[\text{ w emitted in sentence s}|c_s] = \alpha p(w) + (1-\alpha)\frac{exp(\tilde{c}_s, v_w)}{Z_{\tilde{c}_s}} \tag{7.1}$$

where $\tilde{c}_s = \beta c_0 + (1-\beta)c_s, c_o \perp c_s \quad and \quad \alpha$ and β are hyperparameters(Scalar)and $Z_{\tilde{c}_s} = \sum_{w \varepsilon V} exp(\tilde{c}_s, v_w)$ is the normalising constant.

Algorithm 1 Algorithm for Sentence Embedding

Input: Word embeddings $v_w : w \in V$, a set of sentences S, a parameter a and estimated marginal probabilities $p(w) : w \in V$ of the words.

Output: Sentence embeddings $v_s : s \in S$

for all *sentences* $\in \mathcal{S}$
    $v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$
end for
Compute the first principal component u of $v_s : s \in S$
for all *sentences* $\in \mathcal{S}$
    $v_s \leftarrow v_s - uu^T v_s$
end for

7.3 Calculation of Sentence Embedding

In order to calculate the sentence embedding, $Z_{\tilde{c}_s}$ is considered to be the same in all directions because it is assumed that word v_{ws} are approximately uniformly dispersed. So, likelihood for the sentence s can be calculated as follows:

$$p[s|c_s] = \prod_{w \varepsilon s} p(w|c_s) = \prod_{w \varepsilon s} [\alpha p(w) + (1-\alpha)\frac{exp(\tilde{c}_s, v_w)}{Z}] \tag{7.2}$$

Now, Assume

$$f_w(\tilde{c}_s) = log[\alpha p(w) + (1 - [\alpha p(w) + (1-\alpha)\frac{exp(\tilde{c}_s, v_w)}{Z} alpha)\frac{exp(\tilde{c}_s, v_w)}{Z}] \tag{7.3}$$

On the above equation we calculate the following operation :

$$\bigtriangledown f_w(\tilde{c}_s) = \frac{1}{\alpha p(w) + (1-\alpha)\frac{exp(\tilde{c}_s, v_w)}{Z}} \frac{1-\alpha}{Z} exp(\tilde{c}_s, v_w) v_w \tag{7.4}$$

Applying taylor series expansion on the above equation we get:

$$f_w(\tilde{c}_s) \approx f_w(0) + \bigtriangledown f_w(0)^T \tilde{c}_s$$

$$= \text{constant} + \frac{(1-\alpha)/\alpha Z}{p(w)+(1-\alpha)/(\alpha Z)} (v_w, \tilde{c}_s) \tag{7.5}$$

Finally, the maximum likehlihood estimator for the c_s can be written as :

$$\arg\max \sum_{w \varepsilon s} f_w(\tilde{c}_s) \propto \sum_{w_s} \frac{a}{p(w)+a} p(w) \text{ ,where } a = \frac{1-\alpha}{\alpha Z} \tag{7.6}$$

It is the approximation of the weighted average of the word vectors in the corresponding sentence s. As we calculate the weight for a word w by a/p(w)+ a, it will give smaller value for the frequent words. So in order to obtain the final sentence embedding, the projection of $\tilde{c}_s$ is subtracted from the first principal component.

Chapter 8

Experiments

8.1 Testing the models

We tested the models mentioned in the previous section with a test set of 50 question. The test consisted of various questions each testing if the model can answer the questions related to :

- Location of an event that occured in Dr. Abdul Kalam's life
- Time of an important event in Dr. Abdul Kalam's life
- Name of an important person related to Dr. Abdul Kalam
- How an event or an important things happened during his lifetime, etc.

A snapshot of the dataset is show if fig 8.1.

```
1,Who was abdul kalams father
2,Who was abdul kalam mother
3,Where did abdul kalam study
4,When was abdul kalam the president of India
5,How did abdul kalam die
6,What did abdul kalam study
7,what was abdul kalam called
8,Where was abdul kalam born
9,At what age did abdul kalam die
10,Where did abdul kalam die
11,Who is known as missile man of india
12,When did abdul kalam complete his graduation
13,where did abdul kalam graduated from
14,when did abdul kalam presidential term end
15,when did abdul kalam presidential term start
16,Where did abdul kalam taught
17,what did abdul kalam taught
18,who was abdul kalam
19,why was abdul kalam criticised
20,which party supported abdul kalam
21,why was abdul kalam called the missile man of India
22,How many siblings did abdul kalam have
```

Figure 8.1: Snapshot of Test Dataset

Table 8.1: Accuracies for different model

	Siamese	**sen2vec**
Accuracy	42%	62%

Table 8.2: Results for Query : "Where Abdul Kalam born ?"

Wordnet and Corpus statistics	**word2vec**	**Siamese**	**sen2vec**
Bhutanese government lit 1000 butter lamps in homage to the abduls kalamm's death	abdul kalam was the 11th president of india	***abdul kalam was born to a tamil muslim family.***	***Abdul Kalam was born in rameswaram, tamil nadu***
abdul kalam was rushed to the nearby bethany hospital in a critical condition	Abdul kalam worked at the defence research and development organisation (drdo).	***abdul Kalam was born and raised in rameswaram, tamil nadu.***	abdul kalam died from a cardiac arrest on 27 july 2015.
abdul kalam was born in rameswaram, tamil nadu.	abdul kalam is also referred as the "people's president".	Abdul kalams mother ashiamma was a housewife.	On 22nd june 2007, abdul kalam decided not to contest the presidential election again.

Table 8.3: Results for Query : "How did Abdul Kalam die?"

Wordnet and Corpus statistics	**word2vec**	**Siamese**	**sen2vec**
The subject of abdul kalam final book was Transcendence: my spiritual experiences with pramukh swamiji.	abdul kalam was the 11th president of india	*abdul kalam returned to his civilian life of education, writing and public service after a single term.*	***abdul kalam died at the age of 83..***
abdul kalam liked to listen carnatic devotional music daily	Abdul kalam worked atindian space research organisation (isro) .	***abdul kalam died from a cardiac arrest on 27 july 2015.***	abdulabdul kalam liked to study Mathematics..
while climbing a flight of stairs, abdul kalam experienced some discomfort, but was able to enter the auditorium	abdul kalam was intimately involved in india's civilian space programme .	abdul kalam is also known as the missile man of india.	abdul kalam learnt sanskrit.

Table 8.4: Results for Query : "When was Abdul Kalam the president of India?"

Wordnet and Corpus statistics	**word2vec**	**Siamese**	**sen2vec**
abdul kalam played a pivotal organisational, technical, and political role in india's pokhran-ii nuclear tests in 1998	For his work on launch vehicle technology, abdul kalam was nicknamed missile man of india.	*abdul kalam was the 11th president of india*	**Abdul kalam was elected as the 11th president of india in 2002.**
Abdul kalams term lasted from 25 july 2002 to 25 july 2007.	abdul kalam died at the age of 83.	*Abdul kalam died while delivering a lecutre at IIM.*	dr sarvepalli radhakrishnan (1954) and dr zakir hussain (1963) were the earlier recipients of bharat ratna who later became the president of india.
Abdul kalam was elected as the 11th president of india in 2002	abdul kalam was buried in rameshwaram.	abdul kalam studied physics and aerospace engineering.	At PGI chandigarh abdul kalam supported the need of uniform civil code in india in september 2003.

The reason for the poor performance of word2vec and 'Wordnet and Corpus Statistics' model can be attributed to the following reasons:

- These models do not account for new words that may occur outside of the corpus that they were trained on.
- The word mover distance based on word2vec is highly sensitive to word ordering.
- It is highly likely that the WordNet does not contain appropriate entries for Indian context.
- The word2vec model used for experiments is trained on the 300 newsgroups dataset provided by Google. It is possible that Indian terms may be excluded fromt this model hence it performs poorly.

Siamese Network and sen2vec are both trained over the fastText word embeddings which accounts for new words that may occur outside of the corpus that they were trained on. That is why there accuracies are comparitively higher. The Siamese network used for experiments in based on the Quora Question Answering dataset because our corpus was comparitively really small for proper training of Siamese Network. The sen2vec model, however was trained on our dataset using the fastText embeddings.

Table 8.5: Results for Query : "What did Abdul Kalam study?"

Wordnet and Corpus statistics	**word2vec**	**Siamese**	**sen2vec**
abdul kalam wanted to become a fighter pilot.	abdul kalam spent four decades as a scientist and science administrator.	abdul Kalam was a career scientist turned statesman.	***Abdul Kalam liked to study Mathematics.***
abdul kalam became an honorary fellow of IISc, bangalore.	abdul kalam was the 11th president of india	***abdul kalam studied physics and aerospace engineering.***	***Abdul kalam learnt sanskrit***
abdul kalam spent four decades as a scientist and science administrator.	Abdul kalam was also involved in Indian military missile development efforts.	abdul kalam was buried in rameshwaram.	abdul kalam was bachelor his whole life.

Table 8.6: Results for Query : "What was Abdul Kalam known as?"

Wordnet and Corpus Statistics	**word2vec**	**Siamese**	**sen2vec**
abdul kalam read the bhagavad gita	abdul kalam was born in rameswaram, tamil nadu.	Abdul kalam was president of India from 2002 to 2007.	***abdul kalam is also referred as the "people's president"***
Abdul kalam was noted for his integrity and his simple lifestyle.	abdul kalam was the 11th president of india	abdul kalam was intimately involved in india's civilian space programme .	***abdul kalam is also known as the missile man of india.***
The subject of abdul kalam final book was Transcendence: my spiritual experiences with pramukh swamiji	abdul kalam returned to his civilian life of education, writing and public service after a single term.	abdul kalam studied physics and aerospace engineering.	***During his term as president, abdul kalam was affectionately known as the people's president.***

Chapter 9

Conclusion and Future Work

The sentence2vec model performs well for most of the questions with an accuracy of about 68%, while the Siamese network achieves an accuracy of 42%. Rest of the models are not able to capture the semantics of the questions and facts properly and hence, perform very poorly, often giving random results. However, there is still room for a lot of improvements.

In order to improve the results, the following needs to be done to improve the performance of the system :

- Some sort of attention mechanism need to introduced into the neural network to correctly pinpoint to the exact answer.
- The Dataset needs to be expanded to more than 1000 lines to properly train and test the models, since most of the deep learning models require large corpus for training and testing.
- the questions should be more diverse in nature so as to ensure that the system does not overfit for a particular style of questions.

Bibliography

[1] Piotr Bojanowski et al. *Enriching Word Vectors with Subword Information*. 2016.

[2] Minghao Hu, Yuxing Peng, and Xipeng Qiu. "Mnemonic Reader for Machine Comprehension". In: *CoRR* abs/1705.02798 (2017). URL: http://arxiv.org/abs/1705.02798.

[3] Yuhua Li et al. *Sentence Similarity Based on Semantic Nets and Corpus Statistics*. Piscataway, NJ, USA, Aug. 2006. DOI: 10.1109/TKDE.2006.130. URL: http://dx.doi.org/10.1109/TKDE.2006.130.

[4] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *CoRR* abs/1301.3781 (2013). URL: http://arxiv.org/abs/1301.3781.

[5] George A. Miller. "WordNet: A Lexical Database for English". In: *Commun. ACM* 38.11 (Nov. 1995), pp. 39–41. ISSN: 0001-0782. DOI: 10.1145/219717.219748. URL: http://doi.acm.org/10.1145/219717.219748.

[6] Jonas Mueller and Aditya Thyagarajan. "Siamese Recurrent Architectures for Learning Sentence Similarity". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16. Phoenix, Arizona: AAAI Press, 2016, pp. 2786–2792. URL: http://dl.acm.org/citation.cfm?id=3016100.3016291.

[7] *Pretrained Fasttext Vectors*. https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md.

[8] Pranav Rajpurkar et al. "SQuAD: 100, 000+ Questions for Machine Comprehension of Text". In: *CoRR* abs/1606.05250 (2016). URL: http://arxiv.org/abs/1606.05250.

[9] Pranav Rajpurkar et al. "SQuAD: 100, 000+ Questions for Machine Comprehension of Text". In: *CoRR* abs/1606.05250 (2016). URL: http://arxiv.org/abs/1606.05250.

[10] Yelong Shen et al. "ReasoNet: Learning to Stop Reading in Machine Comprehension". In: *CoRR* abs/1609.05284 (2016). URL: http://arxiv.org/abs/1609.05284.

[11] Sainbayar Sukhbaatar et al. "Weakly Supervised Memory Networks". In: *CoRR* abs/1503.08895 (2015). URL: http://arxiv.org/abs/1503.08895.

[12] Wenhui Wang et al. "Gated Self-Matching Networks for Reading Comprehension and Question Answering". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. 2017.

[13] Jason Weston et al. "Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks". In: *CoRR* abs/1502.05698 (2015). URL: http://arxiv.org/abs/1502.05698.