Faculty of Science

INTELLIGENT TEXT CLUSTERING OF ISLAMIC RELIGIOUS TEXTS USING DEEP LEARNING TECHNIQUES

June 30, 2019

Shakeel Ahmad Sheikh

Supervisor: Prof. Dr. Sevinç Gülseçen

Department of Informatics

Istanbul University

Master of Science in Informatics

Abstract: Ignore The Abstract For the Moment

While issuing the rulings regarding some social or religious affair, the Islamic Jurists (legal experts) has to go into all the basic but authentic texts to find the stuff which is directly related or having the contextual analogy or has an association with it. It is obvious that the verdict or ruling of a jury or jurist will be strong if he takes into account all the contextually related stuff which can only be made possible by developing such a system which upon query. Infers a context, and produces before the jury all the related corpus. If the decision is taken just at the shallower level, it potentially leads to the rift and conflict in the society and has been historically observed.

Deep learning techniques preferably Deep Convolutional Neural Network and Generative Adversarial Network architectures were used to develop autoencoders for contextual similarity and analogy. Embedding of the corpus were calculated using Google's word2vec, Doc2vec, ELMO, BERT, FastText and Stanford's Glove. In addition, the embeddings were calculated in itself from the text corpus using cosine similarity in the multidimensional context. Google's Tensorflow along with Keras framework have been deployed to carry out the training of the autoencoder.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"O beloved disciple, knowledge without action is vanity and action without knowledge is insanity!

Imam al-Ghazali

Artificial Intelligence (AI) machines are evaluated for their capacity to emulate human behaviour and according to a classic criterion proposed by Alan Turing, they are said to exhibit intelligent behaviour if they pass the Turing test (Turing, 1950). If the judge confronted with the output given by a machine and a human cannot tell the machine from the human, the test is said to have passed. A chat-bot is an example of such type of machine that emulates the intelligent behaviour of humans. Chat-bot investigates the use of automatic programs instead of humans to interact with the end users.

1.1 Natural Language Processing

We can define Natural language processing (NLP) as a sub-field of AI which deals with the interactions between computing devices and human (natural) languages, in particular how to program computing devices to process, retrieve, classify, categorize and analyze huge amounts of natural language corpus.

What is the most important thing that distinguishes language processing applications from the other data processing applications is the use of their knowledge of language. When a Unix **wc** program is used to count the number of bytes or number of lines in a text file, it is an ordinary data processing application. However, when the same program is used to count the number of words in a file, it requires the knowledge about what it means to be a word and thus can be considered as a language processing application

1.2 Motivation

According to (Gantz & Reinsel, 2011), approximately 90% of the digitized data will be unstructured in the upcoming decade. The unstructured data are heavily loaded in particular with texts and these are ubiquitous like in electronic documents, web-pages, social networking sites. For human, the natural language is easily readable and understandable but due to the humongous volumes of textual data, it is near to impossible for humans to process such volumes of textual data and to categorize, classify, retrieve meanings and patterns from such huge volumes. Text analysis can assist in classifying, categorizing and organizing the data and can help in extracting inferences, rules and patterns from the unstructured data. For finding and predicting trends and knowledge patterns from textual data, various approaches have already been implemented (Özyirmidokuz, 2014). Text Processing can also be used to for financial statements to detect the possible fraud cases. Recently medical industries have started using text processing to exploit the medical documents in order to discover more beneficial drugs (Ku, Chiu, Zhang, Chen, & Su, 2014).

Text Processing implements various diverse tools and techniques in order to derive the useful patterns and inferences from the textual data. We can classify these techniques based on their funcationalities as categorization, classification, summarization, exploratory analysis, information retrieval (IR). Exploratory analysis comprises cluster analysis and topic extraction. Therefore we can consider the text processing as the subset of data mining which utilizes the techniques from Deep Learning Neural Networks (DLNNs) on natural language. In this research various deep learning techniques are applied to the Islamic religious texts in order to obtain the relevant clusters based on their contextual similarities. This research work is among the initial efforts that use deep learning techniques for problem solving in the Islamic textual field.

Thematic categorization of textual data dates back to the Seven Epitomes, is the initial classification system which was created in 26 BC. (Lee, 2010). This consists of many different categories: rhapsodies, arts, lyrics military texts, divination and numbers. At that time all the books(10k approximately) were manually categorized.

The thematic categorization of Islamic texts i.e. ***Hadith*** dates back to the third or 8th century with the Khorasani scholar Abdullah Ibn Al-Mubarak. Hadith is the reports claiming to relate what Prophet Muhammad (S.A.W) said or did on various occasions. Sunnah or Sirah refers to the manner of Muhammad (S.A.W) in his life. The Arabic meaning of Hadith is "narration" or "report" and Sunnah is translated as "Traditions". Hadiths are supposed to demonstrate the Traditions of Muhammad (S.A.W). The Hadiths are the key to the understanding of the Quranic teachings and attaining details about the general instructions presented in the Quran ("Hadith and Sunnah", n.d.). Unlike the collection of Quran which was collected under the official directions of Islamic caliphate in the era of Hz Usman Ibn Affwan (r.a), collection of Hadiths started few decades later. It is believed that the Hadiths were not organized initially after the demise of Propher Muhammad (S.A.W) as the companions of Prophet (S.A.W) feared that people might would mix and confuse the Hadith with Quran. In the time of Caliphs, the collection of Hadiths were banned due to the reason it might lead to disagreements between the Muslims (Dhahabi, 1955) or it may also lead the Muslims to abandon and neglect the Quran. (Al-Nasa'i, 2001). However some documents were like letters to kings, instruction manuals for Muslims and about 65 other documents were recorded because of the insistence of Prophet(S.A.W) (Shaikh, 2006).

In order to understand religion, one has to understand and comprehend the religious text, be it the Quran, Hadith for Muslims or Bible for the Christians or the Vedas for the Hindus. If these religious texts are taken away from their respective religions, the latter will lose its meaning (Verma, 2017). The Islamic scholars use qualitative and critical methods for deep comprehension and understanding of the Islamic texts (Jurisprudence). These qualitative methods consists of going through tens of thousands of pages of plain text.Butt with aid in computing power and free availability of textual data in digitized form, the ability of human beings to store and accumulate information has evolved in a great extent.Since back in those days, the corpus was scant and it was not hard to categorize the data manually. Evaluating and assessing the digitized textual data and categorizing them into meaningful structures is a strenuous and very time-consuming process, and it has become next to impossible with the availability of enormous volumes of web-data. In particular an approach is required how we organize and categorize the Islamic data, in order to get the meaningful insights from the contextual similarity of data and match our ability to understand, comprehend and interpret the information with our enlarging ability to collect it. And to assist Islamic experts and jurists for the verdicts through finding the relevant text or having the contextually analogy to it.

1.3 Terminology

There are some terms used through out this thesis which might require exposition. As I detail them, I will also hone in on the topic we are concerned with. This is a thesis in computer science about the intelligent text clustering of Islamic corpus. We can also say that it is a thesis in the language technology regarding the deep unsupervised neural networks being applied to the texts to cluster and categorize them into similar contextually coherent clusters. Some might argue that language processing technology is a subset of computational linguistics, others that it should be called sub-field of computer science (in particular AI). IR is an approach which deals with how to find and extract information usually from a large corpus. Text clustering is a method of IR, but also a unsupervised deep neural network method which is used to explore and try to find out the new information by amalgamating the unstructured corpus . There is usually no single universal clustering technique which suits all the domains.

1.3.1 Clustering & Categorization

Automatic categorization means to let a model decide to which a domain of set of already established categories a document/text belongs to. In clustering, the model decides how a given text corpora should be partitioned. Categorization can be used when one intends to categorize the new corpus according to the known categorization while as clustering is suitable when one intends to discover and explore new clusters which are not previously known. These methods might give absorbing results on the unseen corpus set; categorization groups them according to well known predefined order and structure, clustering shows the pattern and structure of the particular text corpus set. This thesis mainly focuses on the clustering of textual data based on contextual similarity with the help of deep neural network models.

1.3.2 Cluster, Corpus, Word, Token

In this thesis work, the meta-data like titles, hyperlinks, document numbers etc is not used. The entities I cluster in this thesis are the series/sequence of words, depending on the context which I refer usually as corpus, text, documents etc. The word and token is used interchangeably, is the sequence of characters/letters in a corpus which are usually separated by blanks, questions marks, commas and other delimiters used in English language. For the english and other languages having same set of alphabets, this seems good explanation, but for Arabic, Turkish and other agglutinative languages, it might not be suitable and might be less appropriate to use with these languages. Usually the word comes in many different forms and its basic or lexical form is the lemma. The term words, sentences and documents (Hadiths) are used interchangeably in this thesis.

1.3.3 Clustering Techniques, Representation, Similarity and Word Embeddings

The main goal for clustering can be of various different types. For example, it could beneficial to have a set of corpus segregated into various different partitions of various different levels of understanding and the readability, so that Islamic jurists choose accordingly which text they consult, which may assist in while issuing the verdicts (*fatwa*) at some extremely important point of time. The corpus can be clustered into genres in various different ways. There are many clustering techniques. Most of them require to know the (dis)similarities between the entities (corpus). Some require representation for each word/document and a description of similarity which can be calculated when necessary. How the words/sentences/documents are represented and the definition of similarity vary between the applications. There are various methods to achieve this like, in deep learning domain, the words, sentences and documents are usually represented by word embeddings. The word embeddings is the dense vector representation of words and or documents that express and capture something about the meaning of words/documents. Word embeddings are generated by using a deep neural network to train on a set very large corpus of text. Each word/document is represented by a point in the high dimensional embedding space and these points are learned and shuffled around based on the context of words/documents it is surrounded by. The dissimilarity can be described as the distance between the vectors. In the deep learning models that will be discussed in following chapters, the words or documents as embedding vectors can be considered a set of features which are used to define the entities i.e the corpus.

The clustering technique delivers the partition or a group that tries to fulfill and satisfy some relevant criteria. It could be that the entities in each cluster should be as homogeneous and similar as possible. Howbeit, we should reason that the similarity description reflects the similarity among the actual entities. This work mainly focuses on content divisions i.e, clusters of texts that share similar content. So the question is how do we represent corpus and define the contextual similarity between them? In many deep learning application clusters are represented by words/documents that appear in them and similarity between the two documents / sentences (Hadiths) is defined by considering the vector dense representations of the Hadiths.

1.3.4 Research Goal

During my MS thesis work, I learnt about the word embeddings, clustering algorithms and understood the capacity of deep neural networks used as an observation tool for clustering. This is an application driven research work, but the results seems promising and interesting on their own.

1.4 Text Clustering

Clustering is one of the most important unsupervised technique that automatically analyzes the associations and relations among the textual corpus and categorizes and organizes them into coherent and compact structured clusters of texts based on their contextual similarities. It has a wide range of applications in the field of computer vision (Kanungo et al., 2002), bioinformatics (Zhang, Cheng, Guan, & Zhou, 2015), NLP (Xu et al., 2015).

In the past recent decades, numerous algorithms have been put forward to handle clustering problems (Dueck & Frey, 2007). Howbeit there is no single algorithm that fits all the problem domains. It depends on the specific domain task and the corpus to handle. Broadly there are two approaches of clustering techniques- similarity based and feature based clustering algorithms. Consider an example, suppose the Islamic jurist requires the background detailed knowledge for his verdict on the rulings, but it is not possible for him to look at the details of each Hadith. So what can we do instead is to cluster all the Haditths into say (50) many different clusters based on their contextual similarities. Usually most of the algorithms try to find the innate structures from the underlying original feature space.

1.4.1 Clustering Types

Clustering can broadly be be divided into 2 subgroups:

Hard Clustering: In this type of clustering, each data point (Hadith in our context) either belongs to cluster totally or not. Consider the case of above example, each Hadith is put into exactly one cluster.

Soft Clustering: In soft clustering, rather than putting each Hadith into a separate different cluster, a likelihood or probability of that Hadith in order to be in those clusters is assigned. For example, from the above example each Hadith is assigned a probability to be in either of th say 50 clusters of the text.

Literature Review

Chapter 2

Deep Learning

"Knowledge knocks at the door of action: it enters if the door is opened, but leaves if it does not receive a reply"

Imam Sufyan Al Thawri (RA)

In this chapter, a general outline of the most significant discoveries from the deep learning literature research is given. Most of the existing deep learning techniques in the literature research are detailed and compared. The expansion of deep learning utilization and employment raises an argument: Which technique should be deployed for a specific kind of application? By traversing through these deep learning algorithms, it can give the intuition and perception into the utilization for specific situation. This could give the best suitable method for a specific application domain.

2.1 Machine Learning Definition

Machine learning (ML) is a sub-field of AI in which the computers act in an intelligent way with out being explicitly programmed, relying on patterns and inference instead. It is the study of field that focuses on the creation of algorithms to make predictions based on the data available. ML targets to find and learn a function

$$f : X \rightarrow Y$$

that maps the input feature domain of data X onto the output possible prediction domain Y (Bekkerman, Bilenko, & Langford, 2011). It is creating a mathematical model based on the sample data, called "training data" in order to make predictions/decisions based on the function it learns. The training data consists of labeled (pair of input and desired output prediction) or unlabeled data. The function f is chosen in such a way that it fits the data more accurately, depending on the type of domain. (Mitchell, 1997) defines the ML as "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". E is the data-set that the ML algorithm undergoes. The measure P gives the idea about how good a certain ML algorithm is performing. In classification problems, usually the accuracy (proportion of corrected outputs) is chosen as a performance measure. The applications of ML are email filtering, image classification, text clustering, machine translation etc. ML is also referred to as predictive analysis. A clear distinction between ML and traditional programming can be seen in Fig 2.1

2.1.1 Classification of Machine Learning Algorithms

The ML can broadly be classified into 4 categories:

1. Supervised Learning
2. Unsupervised Learning
3. Semi-supervised Learning
4. Reinforcement Learning

2.1.1.1 Supervised Learning

Supervised learning is defined where we have input (X) and an output (Y) pair available during training and an algorithm is used to learn the mapping $Y = f(X)$.

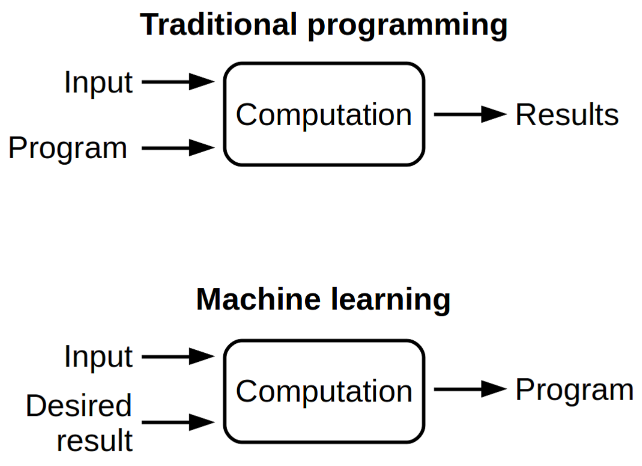

Figure 2.1: Difference between ML and Traditional Programming

The target of the learning is to approximate this function so well that we are able to predict the output on the new input sample not seen during the training phase. The supervised ML can be thought of as a tutor/supervisor supervising the training process. The algorithm continuously makes predictions on training data pair and is corrected by the teacher with the help of already provided corrected input-output pairs. Some common supervised algorithms are Nearest Neighbor, Naive Bayes, Decision Trees, Linear Regression, Support Vector Machines (SVM). The problems which can be solved are classification and regression tasks.

2.1.1.2 Unsupervised Learning

Unsupervised learning models are trained with unlabeled data. In unsupervised, there is only input data (X) available and no corresponding output variables. There is no supervisor at all in the training phase. The objective is to explore the interesting structures/patterns in the data. This family of ML algorithms are mainly used in descriptive modeling and pattern detection. These algorithms try to use methods on the input data to discover patterns, cluster the input data points, mine rules which help in extracting meaningful, accurate and deep insights. These algorithms are usually called descriptive models. Some main unsupervised algorithms include clustering, association rule learning algorithms.

2.1.1.3 Semi-supervised Learning

This category falls in between the supervised and unsupervised ones.It is a mixture of the previous two cases. Applications where only few of the input data is labeled (Y) in the huge quantity of input data (X) set are called semi-supervised learning problems. In most of the real life situations, it is very costly to label all the data as it requires adroit experts to perform labelling. So in this case semi-supervised algorithms are the best possible candidates for building the models. Consider an example of photo archive: the majority of the images are unlabeled and only few are labeled like dog, cat, person.

2.1.1.4 Reinforcement Learning

These techniques aim at utilizing the environment for agents to take actions in order to maximize the reward or minimize the risk in some specific condition. The agents incessantly learns from its nearby surroundings in a repetitive manner.

2.2 Artificial Neural Networks

Neural networks dates back to *The Organization of behavior* (Hebb, 2005), in which an idea of activation behavior of co-fired cells was laid out. This can be compared to how weights are updated in the neural networks:

$$\Delta w = \alpha x.y \tag{2.1}$$

Where w denotes the weights and α is the magnitude with which weights are updated (utilizing neural analogy). The motivation of Hebb was mainly developing mathematical models of biological systems.

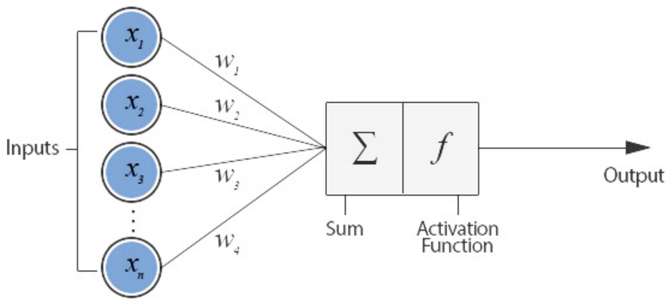

Figure 2.2: Perceptron: A single layered deep neural architecture

In 1958, Rosenblatt inspired by the ideas of Hebb, introduced the Perceptron see Fig. 2.2 which had a wide impact on the future of Deep Learning (DL) research. We can define a Perceptron as the dot product of its input vector x and its corresponding weight matrix w with some bias addition. This product is then being fed to the activation function σ.

$$z(x) = w.x + b \quad where \quad \sigma(z) = \begin{cases} 1 & \text{if } z(x) \geq 0 \\ 0 & otherwise \end{cases} \tag{2.2}$$

By using the terminology of neural networks, the Perceptron can be depicted as the one-layered feed forward neural network (FFNN) having only a single computation unit called neuron, using step-function as an activation function. (Minsky & Papert, 1969) discovered that the Perceptron can not represent some non linear logical operations like XOR which became a bottleneck to the DL research for many years.

Deep neural netwrok (DNN) architectures as shown in Fig. 2.3 gained global attention by overtopping many ML methods like SVM in various different applications such as pattern recognition and NLP. The DNN comprises of computational unites called neurons. Usually the DNNs contains three different layers: the input / 0^{th} layer which consist of input feature vector; the hidden layer containing neurons; and the output layer that consists of response of the neural network depending on its application. It is the hidden layer which maps the input layer with the output layer. These neural networks only allow signals/activations to pass through from input to output, hence the name Feed

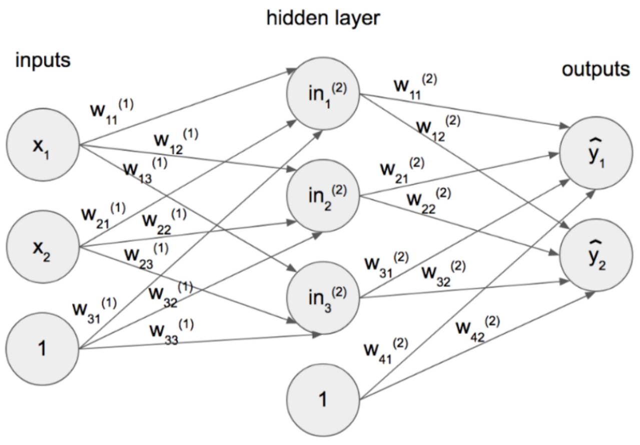

Figure 2.3: Feed Forward Neural Network with single hidden layer

Forward Neural Network (FFNN).

2.2.1 Feed Forward Deep Neural Network Architectures

The output of the one Perceptron can be fed as an input to the other Perceptron, resulting in a depth-wise stack of Perceptrons. (Cybenko, 1989) showed that it is possible to represent any function given the number of neurons at each layer by piling two or more Perceptrons with the sigmoid activation functions, thus resulting in Multi-Layered Perceptrons (MLP). This generalization of MLP is called as the Universal Approximation Property. It was later generalized to all activation functions by (Hornik, 1991). It was because of the use of non-linear, differentiable activation functions and the backpropagation algorithm by (Werbos, 1974) that makes it possible to stack various Perceptrons. Backpropgation is a method used for tuning weights back into the deep neural network with the help of gradient descent. The MLPs are commonly known as Feed Forward Neural Networks (FFNN) as hown in Fig. 2.3. Number of techniques were introduced in order to improve the robustness of the training of neural networks. Among them includes the AdaGrad method by (Duchi, Hazan, & Singer, 2011), an alternative method to gradient descent for tuning weights of the network. Dropout, another technique was introduced by (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) which has reduced the problem of overfitting, thus increases performance.

$$\hat{y_1} = w^2_{11} in^2_1 + w^2_{21} in^2_2 + w^2_{31} in^2_3 + w^2_{41} * 1 \tag{2.3}$$

$$\hat{y_1} = w^2_{12} in^2_1 + w^2_{22} in^2_2 + w^2_{32} in^2_3 + w^2_{42} * 1 \tag{2.4}$$

$$in^2_1 = w^1_{11} x_1 + w^1_{21} x_2 + w^1_{31} * 1 \tag{2.5}$$

$$in^2_2 = w^1_{12} x_1 + w^1_{22} x_2 + w^1_{31} * 1 \tag{2.6}$$

$$in^2_3 = w^1_{13} x_1 + w^1_{23} x_2 + w^1_{33} * 1 \tag{2.7}$$

Equations (2.3), (2.4), (2.5), (2.6), (2.7) translate the graphical representation of FFNN Fig. 2.3 to mathematical formula.

2.2.1.1 Convolutional Neural Networks

When we came a term across Convolutional Neural Networks (CNNs), we usually think of Computer Vision. CNNs were one of the major breakthroughs in image classification problems from self-driving cars to photo tagging. The best way to understand a convolution is to consider it as sliding window function applied to a matrix consisting of image pixels. The sliding window in the CNNs is known as filter, kernel or feature detector. The values of the original matrix are multiplied with the kernel and then summed up by sliding the kernel on each each element to get a full convolution. The CNNs derives its name from the convolution operation in mathematics and signal processing. CNNs use a mixture of various enumerated filters to inflate the data in assorted ways, thus allowing the synchronous examination of the various features in the data (LeCun, Bottou, Bengio, Haffner, et al., 1998). CNNs are actually just a couple of layers of convolutions with some non-linear activations functions like tanh, ReLU applied to the output results.

The application of CNNs to NLP is that instead of image pixels as an input to CNNs, we feed sentences or documents represented as matrix to the CNN network. Each corresponding row of the matrix represents a token (either a word or a character) in its vector form, which are called word embeddings. These embeddings are low-dimensional representations like WordVec (Mikolov, Chen, Corrado, & Dean, 2013) and GloVe (Pennington, Socher, & Manning, 2014) explained in chapter 3, but these representations could also be one-hot encoding of the words. Consider a case of 10 word sentence using a 100-Dimensional word representation, we get an input matrix of size $10 * 100$. The most best fit application for CNNs seems to be tasks like classification including spam detection, topic categorization, sentiment analysis etc. However (Xu et al., 2015) showed that CNNs can also be applied to short text clustering problems.

2.2.2 Recurrent Neural Networks

The another form of neural networks are the Recurrent Neural Networks (RNNs). (Jordan, 1997) came with the classification of NNs as recurrent if the NN graph has a directed cycle in it, opposite to the CNN which is acyclic. The Jordan Network is one of the first published RNN model. This was later formalized by (Elman, 1990) as Elman network, which made the model more flexible in terms of advancing the information through hidden layers. Due to the inclusion of cyclic graphs, it became necessary to discover the variation of backpropagation algorithm which results in the new version of algorithm for RNNs called backpropagation through time (BPTT).

RNNs has shown great promising results in many NLP tasks. RNNs has the ability to capture and process the inbuilt sequential units present in the language. The units could be sentences, words or characters. In RNNs, the computation function at the current input is conditioned on the previous computation. In RNNS, a recurrent unit is being fed by a sequentially fixed-size representation of the input text sequence as shown in Fig. 2.4.

The main advantage of an RNNs is the ability to remember the previous computation results and utilize those computations results in the ongoing computation. This memorization of RNNS makes it well suited for modeling context dependencies in the input sequence. The application of RNNs in NLP include machine translation, language modeling. When we compare RNN with

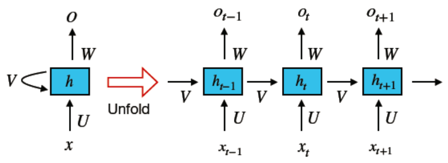

Figure 2.4: Simple Recurrent Neural Network

a CNN model, RNN is better at contextualized specific NLP tasks. The problem with simple RNNs are vanishing gradient problem which makes it very tough to learn and tune the parameters in the training. The other versions of RNNs are long short-term memory (LSTM), gated-recurrent networks (GRUs) and residual networks (ResNets).

2.3 Activation Functions

The activation functions are the primitive feature of ANNs, which decide whether a neuron (computing unit) should be activated or not. The activation functions decide that the information, computing unit is receiving should be ignored or not. Activation functions are the non-linear transformations of input signal, which are later fed to next layers.

$$Y = Activation(\sum(weights * inputs + bias)) \tag{2.8}$$

Without the weights and bias, the activation function simply do a linear transformation. The linear transformation or linear equation is very easy to solve but is limited in its ability to solve much higher complex problems. So, an ANN without an activation function is simply a linear regression model. It is because of these activation functions that makes ANNs capable of learning complex transformations. Some of the popular activation functions are:

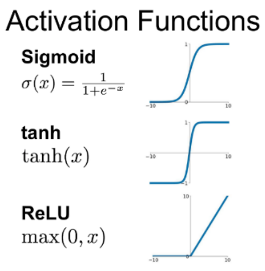

Figure 2.5: Activation Functions

2.3.1 Sigmoid or Logistic Function

It is a function of the form :

$$\sigma(z) = \frac{1}{1 + \exp^{-z}} \tag{2.9}$$

The sigmoid or logistic function is a smooth and differentiable function. The function has the range from 0-1 with an S shape as shown in Fig. 2.5.

2.3.2 Tanh Function

It is just a scaled version of sigmoid function as defined by:

$$\sigma(z) = \frac{\exp^{z} - \exp^{-z}}{\exp^{z} + \exp^{-z}} \quad = \quad 2\sigma(2z) - 1 \tag{2.10}$$

This activation function is symmetric over origin as similar to sigmoid as hown in Fig. 2.5, however it ranges from -1 to 1. Tanh has a steeper gradient than the sigmoid function. The choice of selection depends on the problem domain.

2.3.3 ReLU Function

This is the most widely used activation function . It is called Rectified Linear Unit as hown in Fig. 2.5. It is defined as:

$$\sigma(z) = max(0, z) \tag{2.11}$$

One of the biggest advantage of using ReLU is that it activates only if the input is positive, thus ignoring the negative values which makes the ANNs efficient and easy for computation

2.3.4 Softmax Function

It is a type of sigmoid function, used for multiple classification problem rather than 2. This function shrinks the outputs for each and every class in the range between 0 and 1 The softmax function would squeeze the outputs for each class between 0 and 1. This gives the probability distribution of the inputs being in a specific class. It is defined as

$$\sigma(z)_i = \frac{exp^{z_i}}{\sum_{k=1}^{K} \exp^{z_k}} \quad for \;\; i = 1, ..., K \tag{2.12}$$

Suppose we have the outputs: [1.2 , 0.9 , 0.75], After applying softmax, we would get [0.42 , 0.31, 0.27]. Now these probabilities can be used for the chances for each output class. Softmax is usually used in the final output layer of the NN in order to get the probabilities to define the class of each output.

There is no rule of thumb which activation functions are good or bad to use. Depending on the context of the problem, it is advisable to pick the easy one which is quicker in convergence. Sigmoid generally works good in classifiers. ReLU should be used only in hidden layers.

Chapter 3

Text Embeddings

"You shall know a word by the company it keeps"

Firth

In NLP, the input we feed to the DNNs usually consist of words, letters, sentences and or documents. But this notion or concept is not suitable for computers and thhus there is a requirement to map these text input to a processable numerical representations. The basic input component in this thesis is Hadith, so the current state-of-the-art word embeddings shall be explored and compared in this chapter.

The most basic representation of the input is *one hot encoding.* After preprocessing the corpus, all the basic units(words) are itemized in a vocabulary set and each word in this vocabulary set is assigned its index. These indices in the vocabulary set are random and thus doesn't capture the actual meaning or information.

Most advanced unsupervised learning algorithms point to capture the syntactic and the semantic relationships in the vector embeddings of the words. The similarity between the 2 vector embeddings of two respective words, sentences, documents (Hadiths) are computed in terms of the cosine similarity.

3.1 Word Representations

Understanding the deixis , semantics of a word, phrase and in addition to the extension of bigger units like sentences, paragraphs, documents (Hadiths in our case) is the fundamental pursuit of Natural Language Processing (NLP). In order to let the machines learn to obtain the profound understanding of words, sentences and or documents (Hadiths), we require to describe a characterization or representation of the words and documents which the machines can exercise or operate on. This characterization or representation in the deep learning terminology is called "*word representation or word embeddings*".

How to represent words or documents in a way so as to encode as much information as possible in the context is one of the fundamental challenges in the NLP community. (Bengio, Ducharme, Vincent, & Jauvin, 2003) is the first one who coined the term word embeddings, in the context of training a neural language model. (Collobert & Weston, 2008) exhibited that word embeddings /representations are useful for downstream tasks and are great candidates solutions for pre-training. Due to recent developments in the domain of Machine Learning (ML), more and more complex ML models are trained on substantial larger data-sets, as a result of that out-competing simpler models. The popularization of word embeddings can be ascribed to (Mikolov, Chen, et al., 2013) with the famous Word2Vec and their skip-gram algorithm. Kudos to the work of (Mikolov, Chen, et al., 2013) in which continuous bag-of-words model (CBOW), continuous skip-gram model (CSGM) and log-linear models were presented which leads to the universal recognition of word embeddings or neural embeddings. These models allow very effective estimation of continuous space word representations for words from very huge datasets. (approx. 1.6billion of words) (Mikolov, Chen, et al., 2013). The ongoing research into word representations has been over-topped by these new word embeddings. Their utmost acclaim and huge prosperity can also be ascribed to the distribution of Word2Vec package which comprises of pre-computed word embeddings on Google news data-set (Mikolov, Chen, et al., 2013), and amidst others being GloVe trained on Common Crawl dataset (Pennington et al., 2014), doc2vec (Mikolov, Chen, et al., 2013), fastText trained on Wikipedia data-set

(Bojanowski, Grave, Joulin, & Mikolov, 2017), ELMO (trained on 30 million sentences approximately) (Peters et al., 2018) and the latest one is BERT embeddings which was trained on the concatenation of BooksCorpus (800 million words) 2015) and English Wikipedia (2,500 million words) data-sets (Devlin, Chang, Lee, & Toutanova, 2018).

Neoteric contributions have emphasized the significance of these word embeddings for NLP tasks and their footprint in various experimental studies. Learning embeddings is either done by optimizing the prediction of the word from its context or the context from its word (Bojanowski et al., 2017; Mikolov, Sutskever, Chen, Corrado, & Dean, 2013; Pennington et al., 2014). This is based on the linguist Firth's approach: "You shall know a word by the company it keeps" (Firth, 1968).

It has become a necessity for deep neural network models to have large volumes of data available for training, all can be used as the amount of training data for word embeddings is colossal. Consider the case of GloVe (Pennington et al., 2014) embeddings model, are trained on the Common Crawl data-set which consists of approximately 200 terabytes of raw text which was crawled from all over the web. The training of the model based on the context evolves in the word representation which captures the semantics (meaning) as well as the syntax (already existing in the context).

The word embeddings like Word2Vec or GloVe can further be improved by training them to create the sentence embeddings (Wieting, Bansal, Gimpel, & Livescu, 2015). These embeddings for composition via averaging has remarkably shown significant improvements for semantic textual similarity tasks.

The embeddings of fastText (Bojanowski et al., 2017) is build on Word2Vec by learning the representations for each word and n-grams within each sentence. At each training step the values of the representations are averaged into one vector. This helps word embeedings to capture the sub-word information. It has been shown that the fastText vector representations are more accurate than the Word2Vec representations. ELMO model generates embeddings of a word based on its context in which it appears, thus generating different vector representations of the same word in different contexts (Peters et al., 2018). Recently researchers from Google AI language introduced a new embedding pre-trained model called BERT (Bidirectional Encoder Representations from Transformers) which caused excitement in the NLP community by showing state-of-the-art results in various NLP tasks like Natural Language Inference (NLI), question answering and others. The main key aspect in BERT's renovation is the application of bi-directional training of transformer, which is an attention model to language modelling. The BERT model looks at a corpus differently in contrast to the previous embedding model approaches, which looked at a corpus either from the left to right or combined right to left and left to right training. The BERT model reads the entire text sequence of words at once,thus making it bidirectional. The term bidirectional is usually confused here. It would be appropriate to call it non-directional. This particular characteristic allows the BERT model to learn the context of the given text word based on its nearby surrounding words.

3.1.1 Contextualism

The embeddings are usually learned by considering the adjacent words in the context. (Grice, Cole, Morgan, et al., 1975)'s theory is the benchmark work among the linguists. The significance of context in regards to semantics is mariginalized by Grice as he hypotheizes that the context is only significant for the semantics of indexical words like 'you', 'I' , 'here' etc. and of tenses which are context sensitive structures. However (Searle, 1980; Travis, 1977; Searle, 1985) contends that the semantic relevance of the context in the corpus is much more significant. If one takes an example: "I am at the door", it could be understood either as "Come on Hurry up!" if two or more persons are going for a meeting or or "I am already there", if it is intended for someone who has scheduled an appointment with him/her. The context can be described as the real-world context, the vicinity and with what purposes a word or a statement is made, or it can be described as the linguistic context in terms of nearby statements.

If the later description of the context is taken into meaning, and consider the development in NLP since the emergence of vector representations of words (learned through the context), this development can be used as arguments for the notions of (Travis, 1977; Searle, 1985).

3.1.2 Popular Word Embeddings

Directly after advancement in unsupervised DNNs, the word embeddings methods became popular. It is the mathematical space having one dimension per word from the vocabulary, to a real-valued vector space with lower dimensions $\mathbb{R}$.

The embeddings are indispensable part of the current DNNs for various NLP application domains which include machine translation (Sutskever, Vinyals, & Le, 2014), sentiment analaysis (Dos Santos & Gatti, 2014), document clustering, automatic text summarization (Yousefi-Azar & Hamey, 2017), question answering (Sukhbaatar, Weston, Fergus, et al., 2015), document retrieval (Ganguly, Roy, Mitra, & Jones, 2015) etc. Many DNNs of NLP have in fact successfully substituted conventional distributional features like LSA (Deerwester, 1988) , brown clusters (Brown, Desouza, Mercer, Pietra, & Lai, 1992) and many more.

3.1.2.1 Word2Vec

Contestably, the most favoured embedding representation model is the one which heralded in the popularity rise and resulted in several 100s of papers. The Word2Vec model was introduced in 2013 by (Mikolov, Chen, et al., 2013). This is a shallow deep learning model, consisting of only one hidden layer without non-linearities.

(Mikolov, Chen, et al., 2013) proposed two new unsupervised architectures for learning word representations from a huge text corpus. The first one among the two is called CBOW model as shown in Fig. 3.1. The CBOW model tries to predict the word **"ate"** from the summation of its neighboring context word vectors using a specific window. size.

The second one is called skip-gram model as shown in Fig. 3.2. This model is exactly opposite to the previous CBOW model, it tries to predict the context from the given word. The pre-trained Word2Vec embeddings are trained using the skip-gram architecture.

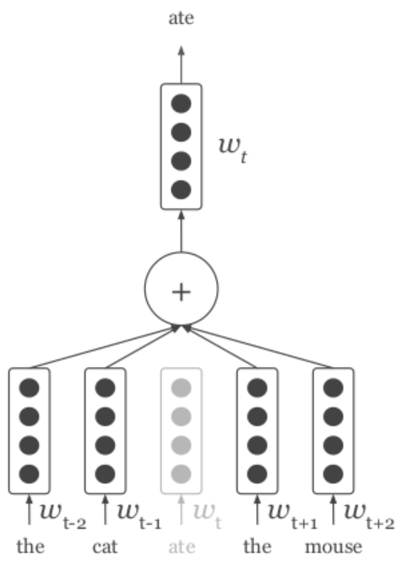

Figure 3.1: CBOW Model. The sentence is “The cat ate the mouse.” and the CBOW model tries to predict the word ‘ate’ from the left context ‘the cat’ and the right ‘the mouse

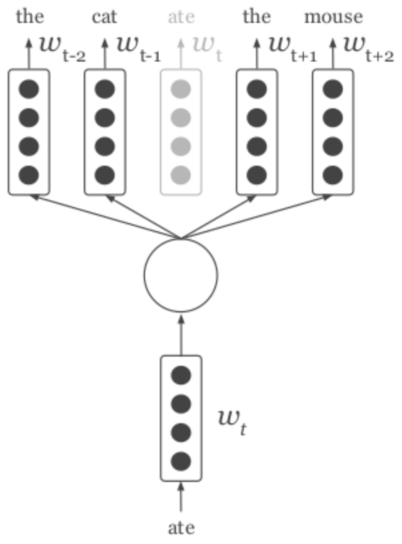

Figure 3.2: Skip-gram Model. Here the model tries to predict the left context ‘the cat’ and the right ‘the mouse’ from the center word ‘ate’

Each word in the skip-gram architecture is assigned both the target (v_w) and the context vectors (u_w), used to predict the context words (c) appearing in the neighborhood of word (w) that are within the window size of M tokens. The probability distribution is expressed in terms of activation called softmax function.

$$p(w/c) = \frac{exp^{u_c^T v_w}}{\sum_{i=1}^{n} \exp^{u_i^T v_w}} \tag{3.1}$$

In order to speed up the training, one could possibly use the methods like negative sampling or hierarchical softmax (Mikolov, Sutskever, et al., 2013).

The conditional probability of the context window factorizes as the product of the conditional probabilities of each word around the context.

$$p(w_{-M}, ...w_M/c) = \prod_{m=-M, m\neq 0}^{M} p(w_m/c) \tag{3.2}$$

In order to find the word embeddings, we maximize the log likelihood of the entire text corpus using the equations 3.1 and 3.3

$$\frac{1}{T}\sum_{i=1}^{T} \sum_{m=-M, m\neq 0}^{M} \log p(w_{i+1}/w_i) \tag{3.3}$$

The pre-trained word embeddings were published by (Mikolov, Chen, et al., 2013) with their work [1]. These embeddngs were trained on the large corpus of Google News of 100 billion tokens of English news articles with dimensionality of 300. In this thesis, we exploit the already publicly available pre-trained embeddings.

3.1.2.2 Doc2Vec

The main objective of the Doc2Vec model is to construct the numeric representations (regardless of length) of the documents. The idea the (Mikolov, Sutskever, et al., 2013) used a simple but a very clever approach: they simply

[1]The pre-trained Word2vec embeddings can be found at :https://code.google.com/archive/p/word2vec/

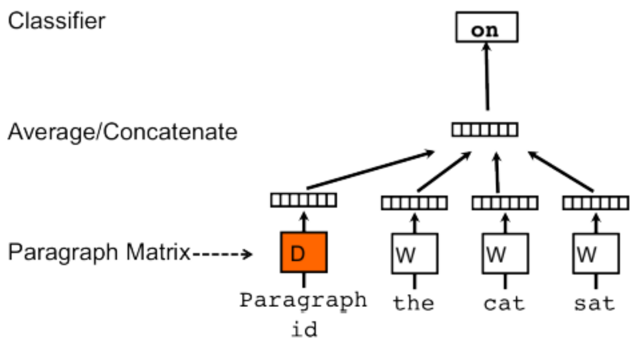

Figure 3.3: Distributed Memory version of Paragraph Vector

used the Word2Vec architecture with another additional vector for paragraph ids as shown in Fig. 3.3.

In Doc2Vec (also called Paragraph Vector- Distributed Memory PV-DM), each and every paragraph is mapped to its unique vector representation, which is represented by matrix (D) column and each word is also represented to its unique vector, represented by matrix column in W (Mikolov, Sutskever, et al., 2013). These vectors are then averaged in order to predict the next word based on the context. Consider paragraph token as an another word, which acts as memory element, remembers the missing information from the context. These context vectors are fixed-length and are shared across all contexts which are produced from the same paragraph but not across the paragraphs. However the word vector matrix W is hared across the paragraphs i. e., the vector for "prophet" is same for all the paragraphs. Stochastic gradient descent in combination with backpropagation was used to train these architectures.

3.1.2.3 FastText

This embedding model was introduced by (Bojanowski et al., 2017), which is the extension of the skipgram architecture. In this model, the idea inspired from (Schütze, 1993) who learned 4-gram character representations via singular value decomposition (SVD) , bags of character of n-grams were considered as atomic rather than the words. Consider an example for bags of character n-grams. The word "lions" with n = 3 will be represented by with character

n-grams as: $< li, lio, ion, ons, ns >$ and with additional sequence $< lions >$.

The biggest advantage of this method it became possible to transfer the meanings among the words, which results in embddings of of new words can now be estimated and extrapolated from the already learned embeddings of the n-grams. The application is obvious in the morphology. Consider that the word "lion" has been seen in the training data but not the word *"lionesque"*. By the help of bags of characters, we know that meaning of the suffix $< esque >$ and the root word $< lion >$, thus can extrapolate the meaning of the *"lionesque"*.

The training is carried out for skipgram with the *negativesampling* by minimizing the following:

$$\sum_{i=1}^{T}\Big(\sum_{m=-M,m\neq 0}^{M} l(s(w_i, w_m)) + \sum_{n\in N} l(-s(w_i, w_m))\Big) \tag{3.4}$$

where $l(x) = \log(1+e^{-x})$ and w_n corresponds the negative sample.The scoring function s is represented as the sum over the bag of character n-grams in the word:

$$s(w, c) = \sum_{g\in G_w} z_g^t v_c \tag{3.5}$$

where G_w represents the collection of all n-grams for the corresponding word w and z are the learned word embeddings. In order to get the embeddings of a specific word , we simply need to add the associated character n-grams.

The pre-trained vectors with dimensionality of 300 for 294 different languages [2] were published by the authors, trained on the Wikipedia dumps. Since in our thesis we are working on the dataset of the English version of ***Sahih Muslim & Sahih Bukhari***, so we use the English word pre-trained word embeddings.

[2]The pretrained fastText embeddings can be found at: https://github.com/facebookresearch/ fastText/blob/master/pretrained-vectors.md.

3.1.2.4 GloVe

This vector representation of words was inspired by the skipgram model (Pennington et al., 2014). The semantic information is contained in the ratio of co-occurrence probabilities of the two specific words as shown by (Pennington et al., 2014). This approach looks somewhat similar to the TF-IDF (Salton & McGill, 1986) approach, but giving the importance of a context word during the training. The co-occurrence statistics of the corpus were gathered in a huge sparse matrix X, in which each item represents the number of times the word i co-occurs with word j within a window size of M, similar to skipgram architecture. This co-occurrence matrix then defines the word embeddings as:

$$w_i^T w_j + b_j = \log(X_{ij}) \tag{3.6}$$

The following least square cost function should be minimized in order to get the optimal embeddings w_i and w_j.

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij})(w_i^T w_j + b_i + b_j - \log(X_{ij}))^2 \tag{3.7}$$

where f is a weighting function that helps to learn from all over the corpus rather than learning only from extremely common word pairs.

Various different representations have been published alongside their work [3] by the authors. The emneddings were trained on tweets, Wikipedia articles with varying dimensionalities. Word embeddings trained from the Common Crawl [4] dataset, which consists of more than 100 billion tokens. The embeddings that were trained on Common Crawl dataset consists of embeddings with dimensionality of 300 with vocabulary size of 2.2 million, which we utilize in this thesis for the clustering problem.

[3] The pretrained GloVe embeddings can be found at:https://nlp.stanford.edu/projects/glove

[4]Common Crawl dataset can be found at:http://commoncrawl.org

3.1.2.5 ELMO

Text embeddings, as shown in the previous sections are the state-of-the-art vector representations. However they are able to capture only the the nearby local context (semantic and syntactic of a given word) due to their shallow nature of network structures. The word can represent different meaning in different linguistic contexts (polysemy). In 2018, (Peters et al., 2018) gave a solution to this problem by developing a new deep contextual embeddings which they call it ELMO embeddings.

ELMO (Embeddings from Language Models) is a deep contextualized word representation, able to model polysemy of a word in different contexts and are also able to model the syntactic and semantic characteristics of a word (Peters et al., 2018). These word vectors are pre-trained representations trained on the humongous corpus. are the learned functions of the latent states of deep bidirectional language models (biLM). These ELMO embeddings have significantly improved the the state-of-the-art NLP challenges like sentiment analysis, textual entailment, question answering etc. In this thesis, we will explore and analyze the results of ELMO embeddings in clustering domain.

The idea (Peters et al., 2018) proposed is that the Language Models (LM) calculates the probability of the full sequence for a given set of N words $(w_1, ..., w_{i-1})$.

It is based on the conditional probability of the next word w_i give the previous words $(w_1, w_2, ..., w_{i-1})$:

$$p(w_1, w_2, ..., w_{i-1}) = \prod_{i=1}^{N} p(w_i / w_1, w_2, ..., w_{i-1}) \tag{3.8}$$

Initially the word representations of a word v_i is being fed to the LSTM with K layers, processing the input sequence in the forward direction. A context vector representation $\vec{h}_{i,k}$ is generated each layer $k = 1, 2, ...K$ for the word w_i and the final layer of the LSTM is trained to predict the next word w_{i+1}, and computing output with the FNN having softmax layer at the top. The other LSTM processes the sequence in the similar fashion but in reverse direction $i.e$, the word w_{i-1} is being predicted. The authors combine the two directions

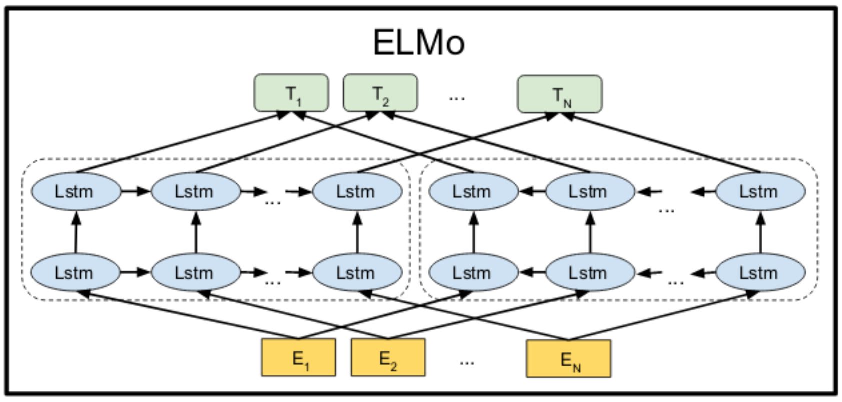

Figure 3.4: ELMO: Concatenation of left-to-right and right-to-left LSTM to the contextual embeddings

into biLM by maximizing their joint log-likelihood as optimization criterion:

$$\begin{aligned}&\sum_{i=1}^{N}(logp(w_i/w_1, w_2, ..., w_{i-1}\ \alpha_x, \overrightarrow{\alpha}_{LSTM}, \alpha_s)\\&+logp(w_i/w_{i+1}, w_{i+2}, ..., w_N\ \alpha_x, \overleftarrow{\alpha}_{LSTM}, \alpha_s))\end{aligned} \tag{3.9}$$

where α_x are the parameters for input word embeddings E and α_s represent the parameters for the top layer softmax and FNN.

ELMO combines the input embeddings v_i and hidden representations $\overleftarrow{h}_{i,k}, \overrightarrow{h}_{i,k}$ of biLSTM as shown in Fig. 3.4.

3.1.2.6 BERT

BERT which stands for Bidirectional Encoder Representations from Transformers is a recently developed language representation model which has been designed to pre-train the deep bidirectional representations in all layers by jointly conditioning on both the right and left context of the given word w_i (Devlin et al., 2018). Consequently, these BERT pre-trained representations can be fine tuned with an extra output layer to develop the futuristic language models like natural language understanding, language inference *etc.*

In its underlying architecture, BERT makes use of the transformer archi-

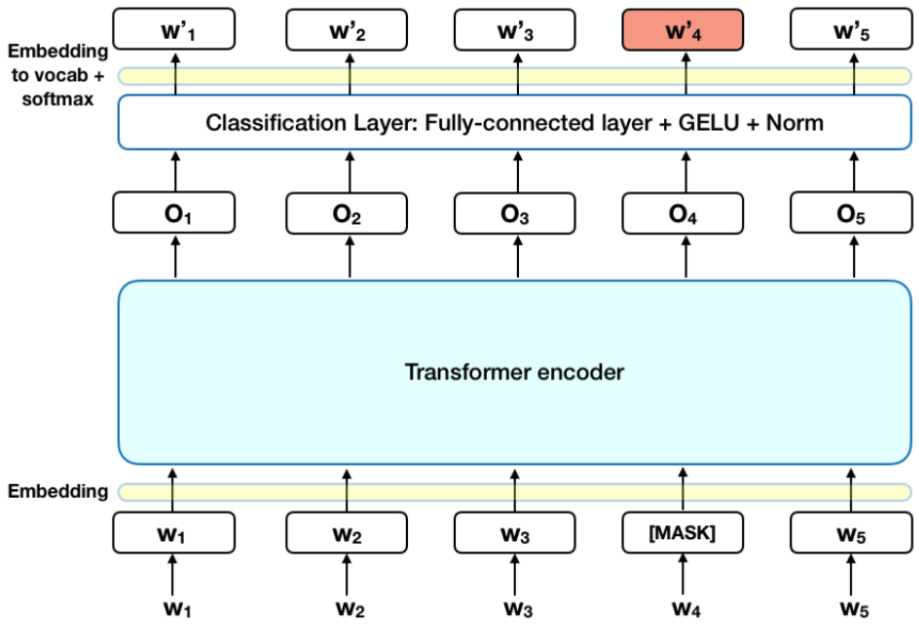
w'1
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

Figure 3.5: High Level description of BERT model

tecture, an attention mechanism able to learn the contextual representation of text. The transformer uses attentions over the sequence rather than the recurrent computation units. Transformer consist of 2 separate models-an encoder that reads the text input, creating the context vector and a decoder that produces a prediction for a give specific task by utilizing the already generated context vectors It is composed of various attention blocks. Each and every block transforms the input using linear and attention layers to the sequence. Since the objective of the BERT is to generate a language model, so only the encoder mechanism is important. In this thesis, we will use the embeddings generated from the BERT analyze its effect on clustering tasks.

The BERT model reads the entire input text sequence at once as opposite to the directional models as shown in Fig. 3.5 . The detailed workings of transformer and BERT can be found in the papers (Vaswani et al., 2017; Devlin et al., 2018).

Chapter 4

Text Clustering

In this chapter, we briefly describe few clustering techniques, among them we explored few in combination with the embeddings for this thesis work. Clustering is the grouping a set of entities in such a way that entities in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters) (Wikipedia contributors, 2019b). Clustering is a conventional methodology in multivariate data mining that is designed to find and explore the innate patterns in the given corpus, in which entities/documents in the same cluster share similar properties and are different from the entities in other clusters. Clustering techniques are applied in various domains like pattern recognition, medical research, economics *etc.* For tasks until three-dimensions, humans can perform clustering easily; for example when looking at a map which is a two-dimensional, one can automatically and easily recognizes various different regions as per to how nearby to each other the regions are situated. However, the intuitive assessments are hard to obtain if the interpretation of the entities reaches higher dimensions.

It is hard to define the notion "cluster" precisely, which is one of the main reasons why so many clustering algorithms exist. The definition of a cluster varies significantly in its characteristics as different researchers employ different cluster model. Having an intuition of these cluster models is an essential in understanding the differences between various algorithms. Some of the cluster models include:

1. Connectivity models: *e.g*, hierarchical clustering develops models based on distance connectivity.

2. Distribution models: clusters are modeled using statistical distributions, such as multivariate normal distributions used by the expectation maximization algorithm.

3. Density models: *e.g*, DBSCAN and OPTICS defines clusters as connected dense regions in the data space.

4. Neural models: the most well known unsupervised neural network is the self-organizing map and these models can usually be characterized as similar to one or more of the above models, and including subspace models when neural networks implement a form of Principal Component Analysis or Independent Component Analysis (Wikipedia contributors, 2019b).

4.1 Clustering Algorithms

Clustering involves the grouping of data points (Hadiths in our case). Given a set of Hadith documents, we can use clustering techniques to classify each document into a specific group. Clustering is a technique of unsupervised learning and is one of the most common methodology used in NLP. In the next section, we describe briefly some clustering algorithms used in this thesis work.

4.1.1 K-Means Clustering

The K-means clustering method is an unsupervised hard clustering method. that groups the N entities/documents (Hadiths) $d_1, d_2, ..., d_N$ into exactly K pre-determined clusters $C_1, C_2, ..., C_K$. The optimal number of clusters K is not known in advance, should be figured out from the corpus (Manning, Raghavan, & Schütze, 2010). The objective of the K-means clustering algorithm is to minimize the squared error shown in Eq. 4.1 or the total intra-cluster variance.

$$J = \sum_{i=1}^{K} \sum_{j=1}^{N} \left\| d_j^i - K_i \right\|^2 \tag{4.1}$$

Algorithm 1 K-Means Algorithm

```
1: Clusters the data points into pre-defined $K$ groups;
2: Choose $K$ random points as cluster centres;
3: Assign data points (Hadiths) to their closest cluster according to the distance function used;
4: Compute the new centroids/means of all data points (Hadiths) in each and every cluster;
5: Repeat steps 2,3,4 until convergence
```

K-Means algorithm is relatively an effective technique. However, it needs to specify the number of clusters, in advance and the final clusters are very sensitive to random initialization and it doesn't guarantee convergence and frequently gets stuck into local optimum. The convergence is exponential in time. The K-means clustering takes only cluster centres into consideration but not the shape. This is equivalent to assuming that clusters are having spherically symmetric shapes and are equal sizes in spheres, meaning that all the dimensions are weighted equally. Unluckily there isn't any way to find the optimal number of clusters. A practical approximation approach is to compare the outcomes of multiple runs with different K and select the best one based on a predefined criterion. In general, if K is large it probably decreases the error but increases the risk of over-fitting. The K-means fails if the clusters have disparate sizes

4.2 Hierarchical Clustering

This type of algorithms creates a hierarchy of clusters. On one extreme, the clustering structure consists of all the data points while on the other en, there are number of clusters equal to the data points. In this clustering technique, it is not possible to re-organize the previously established clusters which makes it a rigid procedure. Originally, hierarchical clustering was meant only for hard clusters, but it was later shown that it can be used for soft clustering as well. The hierarchical clustering can be agglomerative or divisive clustering. The divisive clustering technique is top-down approach it starts from a single cluster to the maximum number of clusters. The agglomerative is a bottom-up approach and opposite to the previous one: each and every observation starts in its own cluster and the groups of clusters are combined into a single cluster while moving up to the hierarchy. The clusters with closest inter-cluster similarity measure are merged continuously until the final one cluster or the pre-defined number of clusters remain. The simple agglomerative clustering algorithm is as follows:

Algorithm 2 Agglomerative Clustering Algorithm

```
1: Calculate the similarity between all pairs of clusters i.e. calculate a similarity matrix whose $ij^{th}$ entry gives the similarity between the $i^{th}$ and $j^{th}$ clusters;
2: Merge and combine the most similar (closest) two clusters;
3: Update the similarity matrix to reflect the pairwise similarity between the new cluster and the original clusters;
4: Repeat steps 2 and 3 until only a final single cluster or pre-defined clusters remains.
```

The agglomerative algorithms are classified based on the inter-cluster similarity measures they use. Among them are single-link, group average, and complete-link. In the single-link, minimum distance between any pair of data points (drawn from clusters) is the distance between the clusters. In the average link, it is the average distance and in the complete-link, it is the maximum distance.

4.2.1 BIRCH Algorithm

BIRCH which means Balanced Iterative Reducing and Clustering using Hierarchies, an unsupervised machine learning algorithm that is used to carry out hierarchical clustering on very large datasets or for streaming data due to its ability to discover very good clustering solutions with only a single scan of data. The technique can improve the clustering quality by making further scans through out the data. The high efficiency achieved by the BIRCH clustering algorithm is by the smart use of smaller set of summary statistics representing large set of data points. These summary statistics are called CFs for clustering purposes which represents ample replacement of the actual data set. The input to the BIRCH algorithm is a set of N data points, represented as real-valued vectors, and a desired number of clusters K (Wikipedia contributors, 2019a). The BIRCH clustering algorithm operates in 4 phases:

The first phase of the BIRCH clustering algorithm builds a height-balanced CF tree, defined as follows:

1. The CF for a given set of N-dimensional data points is defined as the triplet $CF = (N, LS, SS)$, where LS and SS are defined as follows:

    (a) **Linear Sum: LS** Sum of individual coordinates, which is the measure of the coordinates (locations) of the cluster.

    $$\overrightarrow{LS} = \sum_{i=1}^{N} \overrightarrow{X_i} \tag{4.2}$$

    (b) **Squared Sum: SS** This is defined as the measure of the spread of the cluster.

    $$\overrightarrow{SS} = \sum_{i=1}^{N} (\overrightarrow{X_i})^2 \tag{4.3}$$

2. The CFs are arranged in a height-balanced tree with three parameters: branching factor B (which determines the maximum number of children allowed for a non-leaf node), threshold T (is the upper bound to the radius in a leaf node of a cluster) and L (number of entries in a leaf node). The CF entry equals the sum of the CF entries in the child nodes of that entry for non-leaf nodes and for a leaf node, it is simply its value. The creation of CF tree, carried out by the sequential clustering approach, where one data point at a time is scanned at a time, and decides about the assignment of a given data point to be included into existing cluster or the new one. The general CF tree is shown in Fig.

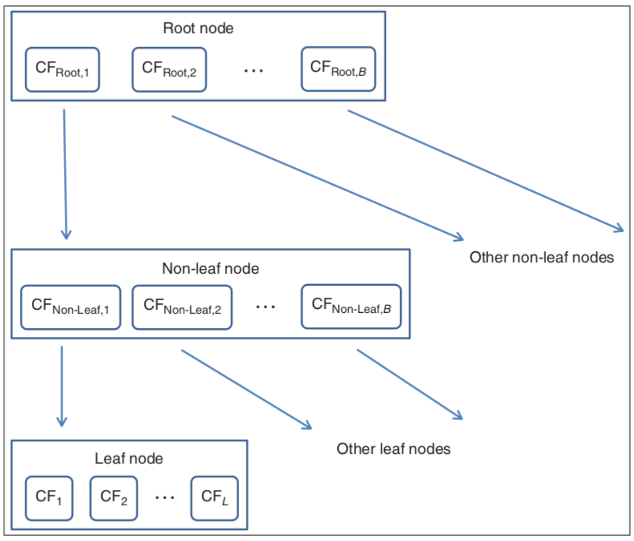

Figure 4.1: CFT General Structure

4.1.

In the original presentation of BIRCH method, this step is marked as optional.In this 2nd phase, the BIRCH clustering algorithm reconstructs a smaller CF tree by scanning all the entries of leaf nodes in the initial CF tree, grouping the crowded subclusters into bigger ones and removing the outliers.

In the 3rd phase of BIRCH method, all leaf entries are clustered by an existing agglomerative hierarchical clustering algorithm (applied directly on subclusters represented by their CF vectors). This provides user , a flexibility to specify either the yearning diameter threshold for clusters or the number of clusters. The set of clusters obtained after this phase captures the major distribution of the data points.

Some localized and minor inexactness might remain after the end of phase 3, which then can be sorted out in the last and final phase of the BIRCH method. In this 4th phase, the centroids generated from the phase 3 are used as seeds and redistributing the data points to its closest seeds to generate new set of clusters. Outliers can also be dicarded in this phase.

Consider an example for CF calculations in the Fig. 4.2, the data points which are in the cluster 1 and 2 respectively are $(1,1)$,$(2,1)$,$(1,2)$ & $(3,2)$,

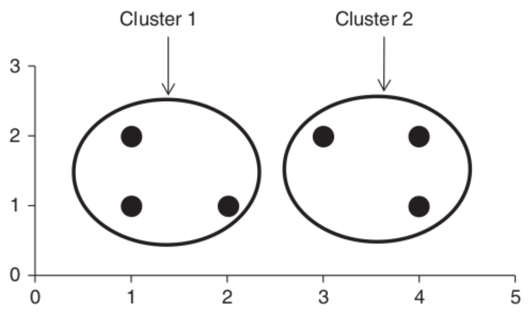

Figure 4.2: Clusters

$(4,1)$, $(4,2)$. The cluster feature CF_1 for cluster 1 contains:

$$\begin{aligned} CF_1 = \{3, (1+2+1, 1+1+2), (12+22+12, 12+12+22)\} \\ = \{3, (4,4), (6,6)\} \end{aligned} \tag{4.4}$$

And CF_2 for cluster 2 is

$$\begin{aligned} CF_2 = \{3, (3+4+4, 2+1+2), (32+42+42, 22+12+22)\} \\ = \{3, (11,5), (41,9)\} \end{aligned} \tag{4.5}$$

The merging of CFs of 2 clusters can simply be added according to the $AdditivityTheorem$. Thus for the clusters in the Fig. 4.2m the resulting CF would look like:

$$CF_12 = \{3+3, (4+11, 4+5), (6+41, 6+9)\} = \{6, (15,9), (47,15)\} \tag{4.6}$$

4.2.1.1 Calculations with Cluster Features

Without the actual knowledge of underlying values given only CFs $= (N, \overrightarrow{LS}, \overrightarrow{SS})$, the above measures can be calculated as follows:

$$Centroid : \overrightarrow{C} = \frac{\sum_{i=1}^{N} \overrightarrow{X_i}}{N} = \frac{\overrightarrow{LS}}{N} \tag{4.7}$$

$$Radius: \sqrt{\frac{\sum_{i=1}^{N}(\overrightarrow{X_i} - \overrightarrow{C})^2}{N}} = \sqrt{\frac{N.\overrightarrow{C}^2 + \overrightarrow{SS} - 2.\overrightarrow{C}.\overrightarrow{LS}}{N}} \tag{4.8}$$

and the average linkage distance between the clusters $CF_1 = (N_1, \overrightarrow{LS_1}, \overrightarrow{SS_1})$ and $CF_2 = (N_2, \overrightarrow{LS_2}, \overrightarrow{SS_2})$, is calculated as:

$$D = \sqrt{\frac{\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}(\overrightarrow{X_i} - \overrightarrow{Y_j})^2}{N_1}.N_2} = \sqrt{\frac{N_1.\overrightarrow{SS_2} + N_2.\overrightarrow{SS_1} - 2.\overrightarrow{LS_1}.\overrightarrow{LS_2}}{N_1.N_2}} \tag{4.9}$$

4.3 Spectral Clustering

Most of the clustering algorithms like K-means makes an assumption on the shape of the data (radial basis, spiral *etc.*) and also requires multiple restarts at initialization to find the local minimas. Spectral clustering helps to solve these problems by neither making the assumption on the shapes of the clusters nor is sensitive to the initialization. This technique is based on the power of graph theory and makes use of the eigenvalues of the similarity matrix for dimensionality reduction before performing clustering (Wikipedia contributors, 2019f). The algorithm works in 3 stages as:

Algorithm 3 Spectral Clustering

```
1: Construct a similarity graph (e.g., KNN graph) for all the data points;
2: Create a low-dimensional spectral embeddings with the use eigenvectors
   of the graph Laplacian of the data points, where the clusters are more
   obvious to categorize;
3: Apply any classical clustering algorithm like $K$-means to partition the
   embedding space i.e, use a lowest eigen value to choose the eigenvector for
   the cluster.
```

4.4 Self Organizing Map (Kohonen Maps)

Self-organizing map (SOM) or self-organizing map feature map (SOFM) invented by T.Kohonen, is a class of ANNs, trained unsupervisedly to produce low-dimensional (typically 2D), discretized representation of the input space

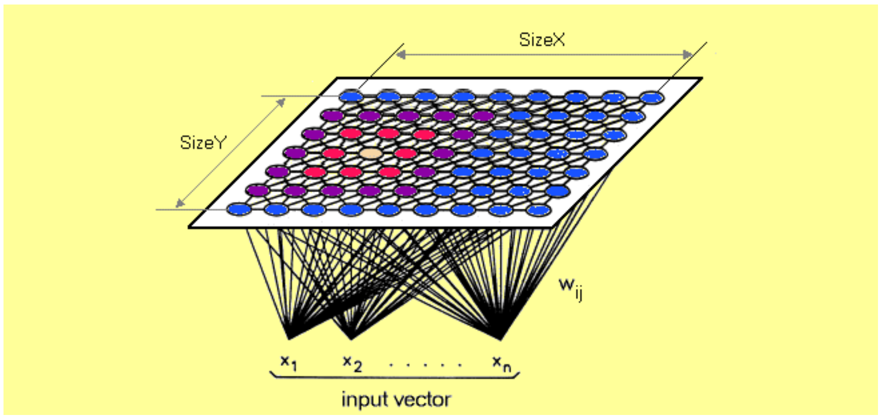

Figure 4.3: Self-Organizing Map scheme

training samples, known as a map, thus is a method of dimensionality reduction. SOMs applies competitive learning as opposite to other ANNs which applies error-correction learning (like propagation with gradient descent). A neighborhood function is used in SOMs to preserve the topological properties of the input space (Wikipedia contributors, 2019d).

SOM also provides a data visualization technique, helps to understand the high dimensional data by reducing the dimensions to a map as shown in Fig. 4.3. A clustering concept can also be represented by SOM , by grouping the similar data objects together, thus SOM captures clusters as well as reduces the dimensions. In SOM, the computaional units (neurons) are ordered in two layers: the input and the competition layer. The input layer consists of N neurons (for each input variable) and the competition layer consists low-dimensional grid of neurons. Clustering in SOM is performed by having multiple several units vie for the current data object. After the data enters into the system, the ANN is trained based on the inputs. Units having the weight vector closest to the current object becomes the active or winning unit. The neighborhood relationships existing within the input data are preserved by gradually adjusting the values of input variables during the training phase. Each data point recognizes themselves by competing for representations. The SOM begins by initializing the weight vectors. After initializing the weight vectors, a vector is sampled randomly and is searched in the map of weight vectors to find the best match that represents the corresponding sample. Each weight vector consists of nearby neighboring weights. The chosen weight and its neighbors are rewarded to become more likely to select the sample sample vector randomly allowing the map to grow and form different shapes, generally, square/rectangular/hexagonal/L shapes in 2D feature space. In short, learning occurs in several various steps and over many iterations:

Algorithm 4 SOM Clustering Algorithm

```
1: Each node's weights are initialized;
2: A vector is chosen at random from the set of training data;
3: Every node is examined to calculate which one's weights are most like the
   input vector. The winning node is commonly known as the Best Matching
   Unit (BMU);
4: Then the neighbourhood of the BMU is calculated. The amount of neigh-
   bors decreases over time;
5: The winning weight is rewarded with becoming more like the sample vector.
   The nighbors also become more like the sample vector. The closer a node
   is to the BMU, the more its weights get altered and the farther away the
   neighbor is from the BMU, the less it learns;
6: Repeat step 2 for N iterations [1].
```

4.5 Deep Autoencoders with K-means

Autoencoders belong to a family of FFNNs where the out is same as the input. The input data is compressed into a lower-dimensional latent-space representation and then re-formulate the output from this latent-space representation. The autoenocder is composed of 3 components: encoder, latent-space representation and decoder. The encoder compresses the input data and creates the latent-space representation, which is then used by the decoder to reproduce the input as shown in Fig. 4.4. These methods are usually compression or dimensionality reduction algorithms. After reducing the high input dimensionality using autoencoders, we feed it to the K-means algorithm to explore and analyze the clusters being formed.

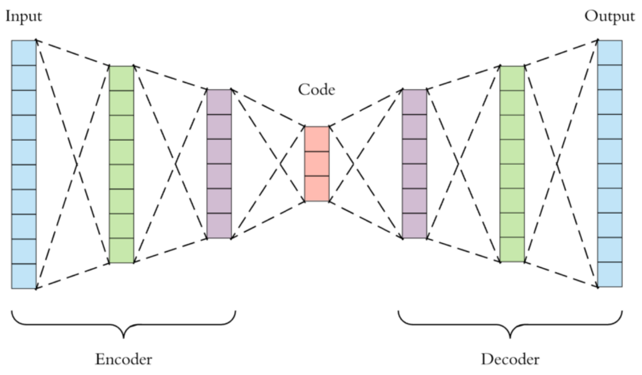

Figure 4.4: Autoencoder Architecture

4.6 Deep Embedded Clustering

Deep embedded clustering (DEC) is a neural network based method proposed by (Xie, Girshick, & Farhadi, 2016) which learns feature representations and clustering assignments concurrently. A mapping is learned from the input data space to a lower-dimensional feature space where the clustering objective is optimized iteratively.

Instead of performing clustering data points n $\{x_i \in X\}_{i=1}^{n}$ directly into k clusters with centroids $\mu_j, j = 1, 2, ...k$, the DEC method first performs a non-linear mapping of the input data space X into a different lower-dimensional latent-space Z

$$f_\theta \colon X \to Z$$

where θ are learnable parameter and Z is the latent feature space.

Clustering on data points is performed by concurrently learning a set of k cluster centres $\{\mu_j \in Z\}_{j=1}^{k}$ in the feature space Z and the parameters θ of DNN that maps input data space into latent feature space Z. The DEC consists of 2 phases:

1. Initialization phase with deep autoencoders.
2. Parameter optimization (clustering) iterartively between computing an auxiliary target distribution and minimizing the Kullback-Leibler (KL) divergence to it.

4.7 Metrics

The most primitive consideration is assessing the quality of DL models while deploying. The assessment is easier for supervised class of problems, as the labels are there for each and every example. However there is no such richness in the unsupervised contexts, the notion if testing in this domain is a flawed premise. But that is not a lost cause for assessing the unsupervised models. Myriad metrics study the quality of unsupervised clustering results. These metrics can be insightful into the clusters might vary depending on the choice of algorithm selected and the natural propensity of data to categorize together. Some of the metrics include:

4.7.1 Psueod-F Statistic

Consider, we have K clusters, each having n_i data points each respectively, so that $\sum n_i = N$ (total size of the sample). Let x_{ij} point to the j^{th} data value in the cluster i, let m_i be the cluster centroid of the i^{th} cluster, and M be the overall mean of all the data points. The (SSB) *sum of squares between* the clusters and (SSE) *sum of squares* within the clusters are defined as:

$$SSB = \sum_{i=1}^{K} n_i.Distance^2(m_i, M) \tag{4.10}$$

$$SSE = \sum_{i=1}^{K} \sum_{j=1}^{n_j} Distance^2(x_{ij}, m_i) \tag{4.11}$$

where

$$Distance(a, b) = \sqrt{\sum (a_i - b_i)^2} \tag{4.12}$$

Then, the $pseudo - F$ statistic is defined as:

$$F = \frac{\frac{SSB}{K-1}}{\frac{SSE}{N-K}} = \frac{MSB}{MSE} \tag{4.13}$$

The $pseudo - F$ statistic measures the ratio of (i) MSB mean square between the clusters which computes the separation between the clusters to (ii) the

MSE mean squared error which measures the spread of the data within the clusters.

4.7.2 Davies-Bouldin (DB) Index

The formula for calculating DB index is:

$$DBI = \frac{1}{n}\sum_{i=1}^{n} max_{j\neq i}(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}) \tag{4.14}$$

where n is the number of clusters and σ_i is the average distance of all data points in cluster i from the cluster centroid c_i.

The small index corresponds to good clusters implying that the clusters are compact and centres are way away from each other. It is therefore, the cluster layout that minimizes DB index is taken as the optimum number of clusters (Wikipedia contributors, 2018).

4.7.3 Silhouette Coefficient

This technique provides a compact and succinct geometrical representation of how good each data point has been clustered. It is a method of validation and exposition of consistency within the clusters. This value measures how similar a data point is to its own cluster (cohesion) in compassion to other clusters (separation) (Wikipedia contributors, 2019e). The value ranges between { -1,1 }. The high value means that the data point is very well matched to its own cluster and poorly matched to neighboring clusters. If the majority of data points have high silhouette value, then we can say that the clustering structure is the appropriate one. If the majority contains low or negative values, then the clustering structure contains either very few clusters or too many clusters. The silhouette metric can be calculated with any distance metric like Manhattan or Euclidean distance. The silhouette value is defined as:

$$S(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{4.15}$$

where $a(i)$ is defined as the average distance of point i from all other points in its own cluster and $b(i)$ is the smallest average distance of i to all other points in any other cluster.

4.7.4 Dunn Index

It is defined as follows:

$$D = \frac{min_{1 \leq i < j \leq n} d(i,j)}{max_{1 \leq k < n} d'(k)} \tag{4.16}$$

where $i, j, \& k$ are the cluster indices, d is the inter-cluster distance and $d^{'}$ is the intra-cluster distance. The higher the value of Dunn index, the appropriate the clustering is (Wikipedia contributors, 2019c).

Chapter 5

Methodology, Tools and Experimental Setup

In this chapter, we describe the dataset, the design approach, and the software tools that we used to implement and conduct our experiments.

5.1 Dataset

The experiments were conducted on the English version of the Islamic texts: ***Sahih Muslim & Sahih Bukhari*** [1]. The *Sahih Bukhari* is composed of 93 different chapters with a total of 7275 hadith documents and the *Sahih Muslim* is composed of 43 different chapters with a total of 9200 different narrations. Before feeding the data to the model, some basic preprocessing were carried out on the dataset. The frequent words (stop words) were removed.

5.2 Research and Design Strategy

The preprocessing and clustering techniques planned to use during this thesis are the existing ones. However, we propose a word embedding based clustering like Word2Vec, Doc2Vec, GloVe and fastText and their analysis while used with the clustering algorithms.

[1] https://www.quranexplorer.com/hadithebook/english/index.html

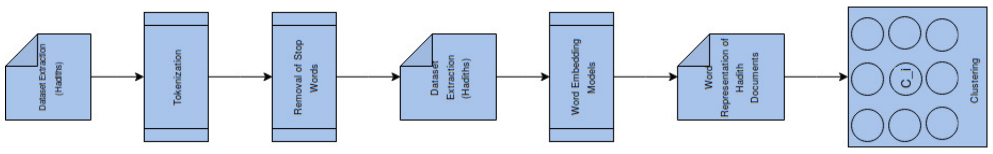
Dataset Extraction (Hadiths)
Tokenization
Removal of Stop Words
Dataset Extraction (Hadiths)
Word Embedding Models
Word Representation of Hadith Documents
C_i
Clustering

Figure 5.1: Overall research design

In this thesis, several clustering algorithms like K-means, Birch, DEC, and Spectral clustering have been explored and analysed in combination with the various deep representation of the documents. The representations we use in this case study are Word2Vec, Doc2Vec, GloVe and fastText. The overall research methodology of this case study is illustrated in the Fig. 5.1:

5.2.1 Dataset Preprocessing

Before feeding the Hadith documents directly to the unsupervised clustering algorithms, we first require to pre-process the data. The preprocessing includes:

5.2.1.1 Tokenization

Tokenization is the process of splitting up a long sequence of strings into its constituent pieces like words, phrases called tokens. Consider an input sequence *In the name of Allah, the most beneficent and most merciful* and its tokens will be {'In', 'the', 'name', 'of', 'Allah', 'the', 'most', 'beneficent', 'and', 'most','merciful' }. Since we are using the English version of the corpus, so. we used $RegexpTokenizer$ [2] to carry out the tokenization process on our corpus.

5.2.1.2 Removal of Stop Words

Stop words are the useless words such as *the*, of, *an*, *etc.* which occurs very frequently in the language. After the removal of stop words, the input sequence in the previous section will look like {'name', 'Allah', 'most', 'beneficent', 'most', 'merciful' }. Removing stop words is one of main steps to deal with the various NLP text problems by preparing the dataset better before feeding

[2]https://www.nltk.org/_modules/nltk/tokenize/regexp.html

it to the deep neural network models. For the removal of stop words, we use the library called stopwords from the NLTK[3] toolkit.

5.2.2 Representation of Hadith Documents

Word embedding models are very efficient methods to represent words, sentences or documents as vectors in high dimensional space. In this thesis, we use various embedding models like Word2Vec, Doc2Vec, GloVe, and fastText to generate the embeddings from the whole preprocessed corpus and analyzed their results when used in combination with various clustering techniques. In our case study, rather than using the pre-trained embeddings, we retrain the embedding models to generate new embeddings on our corpus with an embedding dimension size of 300. The embedding models are explained in detail in chapter 3.

5.2.3 Clustering Algorithms

After generating the embeddings from the preprocessed corpus, we feed these unlabeled embeddings to the various unsupervised clustering algorithms. In this thesis, we analyze and compare the effect of word embeddings on K-means, BIRCH, DEC, Spectral and SOM unsupervised clustering techniques which are detailed in chapter 4.

5.2.4 Software Hardware Tools

Due to the wide use of Python programming language in computing research and in the deep learning vicinity, we have chosen Python (version 3.7.2) for this thesis implementation with the libraries like Numpy(Oliphant, 2006), Scikit-learn(Pedregosa et al., 2011), and the Keras(Chollet et al., 2015) deep learning library of Tensorflow.The experiments were conducted on an 2,2 Ghz4-core Intel Core i7 processor.

[3]https://www.nltk.org/book/ch02.html

Chapter 6

Results

Chapter 7

Discussions and Conclusion

Appendix A

Appendix A

Bibliography

Bekkerman, R., Bilenko, M., & Langford, J. (2011). *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, *3*(Feb), 1137–1155.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146.

Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., & Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational linguistics*, *18*(4), 467–479.

Chollet, F. et al. (2015). Keras. https://keras.io.

Collobert, R. & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on machine learning* (pp. 160–167). ACM.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, *2*(4), 303–314.

Deerwester, S. (1988). Improving information retrieval with latent semantic indexing.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:18 Attention is All You Need 10.04805*.

Dhahabi, A. A. A. S. a. (1955). Tadhkirat al-huffaz. Beirut: Dar Ihya# Turath al-Arabi.

Dos Santos, C. & Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of coling 2014, the 25th international conference on computational linguistics: Technical papers* (pp. 69–78).

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*(Jul), 2121–2159.

Dueck, D. & Frey, B. J. (2007). Non-metric affinity propagation for unsupervised image categorization. In *2007 ieee 11th international conference on computer vision* (pp. 1–8). IEEE.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179–211.

Firth, J. R. (1968). *Selected papers of jr firth, 1952-59*. Indiana University Press.

Ganguly, D., Roy, D., Mitra, M., & Jones, G. J. (2015). Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th international acm sigir conference on research and development in information retrieval* (pp. 795–798). ACM.

Gantz, J. & Reinsel, D. (2011). Extracting value from chaos. *IDC iview*, *1142*(2011), 1–12.

Grice, H. P., Cole, P., Morgan, J. L., et al. (1975). Logic and conversation. *1975*, 41–58.

Hadith and Sunnah. (n.d.). Retrieved May 15, 2019, from https://onthesigns.com/2019/04/26/hadith-and-sunnah-reports-and-traditions-of-the-prophet/

Hebb, D. O. (2005). *The organization of behavior: A neuropsychological theory*. Psychology Press.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, *4*(2), 251–257.

Jordan, M. I. (1997). Serial order: A parallel distributed processing approach. In *Advances in psychology* (Vol. 121, pp. 471–495). Elsevier.

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., & Wu, A. Y. (2002). An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7), 881–892.

Ku, Y., Chiu, C., Zhang, Y., Chen, H., & Su, H. (2014). Text mining self-disclosing health information for public health service. *Journal of the Association for Information Science and Technology*, *65*(5), 928–947.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lee, H.-L. (2010). Organizing knowledge the chinese way. In *Proceedings of the 73rd asis&t annual meeting on navigating streams in an information ecosystem-volume 47* (p. 18). American Society for Information Science.

Manning, C., Raghavan, P., & Schütze, H. (2010). Introduction to information retrieval. *Natural Language Engineering*, *16*(1), 100–103.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111–3119).

Minsky, M. & Papert, S. (1969). Perceptrons cambridge. *MA: MIT Press. zbMATH.*

Mitchell, T. M. (1997). *Machine learning* (1st ed.). New York, NY, USA: McGraw-Hill, Inc.

Al-Nasa'i, A. (2001). Al-sunan al-kubra. *Hasan 'Abd al-Man 'am Shilbi*, 1–10.

Oliphant, T. E. (2006). *A guide to numpy.* Trelgol Publishing USA.

Özyirmidokuz, E. K. (2014). Mining unstructured turkish economy news articles. *Procedia Economics and Finance*, *16*, 320–328.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)* (pp. 1532–1543).

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Salton, G. & McGill, M. J. (1986). Introduction to modern information retrieval.

Schütze, H. (1993). Word space. In *Advances in neural information processing systems* (pp. 895–902).

Searle, J. R. (1980). The background of meaning. In *Speech act theory and pragmatics* (pp. 221–232). Springer.

Searle, J. R. (1985). *Expression and meaning: Studies in the theory of speech acts*. Cambridge University Press.

Shaikh, K. M. (2006). *Hadith & hadith sciences*. Adam Publishers.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929–1958.

Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in neural information processing systems* (pp. 2440–2448).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112).

Travis, C. (1977). Saying and understanding: A generative theory of illocutions.

Turing, A. M. (1950). I.—Computing Machinery And Intelligence. *Mind*, *59*(236), 433–460. doi:10.1093/mind/LIX.236.433. eprint: http://oup.prod.sis.lan/mind/article-pdf/LIX/236/433/9866119/433.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention is all you need. Retrieved from https://arxiv.org/pdf/1706.03762.pdf

Verma, M. (2017). Lexical analysis of religious texts using text mining and machine learning tools. *International Journal of Computer Applications*, *168*(8), 39–45.

Werbos, P. (1974). Beyond regression:" new tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University.*

Wieting, J., Bansal, M., Gimpel, K., & Livescu, K. (2015). Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198.*

Wikipedia contributors. (2018). Davies–bouldin index — Wikipedia, the free encyclopedia. [Online; accessed 1-June-2019]. Retrieved from https://en.wikipedia.org/w/index.php?title=Davies%E2%80%93Bouldin_index&oldid=836730962

Wikipedia contributors. (2019a). Birch — Wikipedia, the free encyclopedia. [Online; accessed 8-June-2019]. Retrieved from https://en.wikipedia.org/w/index.php?title=BIRCH&oldid=894397173

Wikipedia contributors. (2019b). Cluster analysis — Wikipedia, the free encyclopedia. [Online; accessed 28-May-2019]. Retrieved from https://en.wikipedia.org/w/index.php?title=Cluster_analysis&oldid=893363946

Wikipedia contributors. (2019c). Dunn index — Wikipedia, the free encyclopedia. [Online; accessed 1-June-2019]. Retrieved from https://en.wikipedia.org/w/index.php?title=Dunn_index&oldid=883919411

Wikipedia contributors. (2019d). Self-organizing map — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Self-organizing_map&oldid=892805093. [Online; accessed 1-June-2019].

Wikipedia contributors. (2019e). Silhouette (clustering) — Wikipedia, the free encyclopedia. [Online; accessed 1-June-2019]. Retrieved from https://en.wikipedia.org/w/index.php?title=Silhouette_(clustering)&oldid=888649513

Wikipedia contributors. (2019f). Spectral clustering — Wikipedia, the free encyclopedia. [Online; accessed 10-June-2019]. Retrieved from https://en.wikipedia.org/w/index.php?title=Spectral_clustering&oldid=891681676

Xie, J., Girshick, R., & Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In *International conference on machine learning* (pp. 478–487).

Xu, J., Peng, W., Guanhua, T., Bo, X., Jun, Z., Fangyuan, W., Hongwei, H., et al. (2015). Short text clustering via convolutional neural networks.

Yousefi-Azar, M. & Hamey, L. (2017). Text summarization using unsupervised deep learning. *Expert Systems with Applications*, *68*, 93–105.

Zhang, R., Cheng, Z., Guan, J., & Zhou, S. (2015). Exploiting topic modeling to boost metagenomic reads binning. In *Bmc bioinformatics* (Vol. 16, *5*, S2). BioMed Central.