ISSN 2278-3091
Volume 10, No.5, September - October 2021
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse121052021.pdf
https://doi.org/10.30534/ijatcse/2021/121052021

A Comprehensive Analytical Study of Traditional and Recent Development in Natural Language Processing

Aditya Datta[1], Biswajit Jena[2*], Amiya Kumar Dash[3], Roshni Pradhan[4]
[1]International Institute of Information Technology, Bhubaneshwar, Odisha, India, Email:b518017@iiit-bh.ac.in.
[2]International Institute of Information Technology, Bhubaneshwar, Odisha, India, Email: c118002@iiit-bh.ac.in.
[3]KIIT University, Bhubaneshwar, Odisha, India, Email:amiya.dasfcs@kiit.ac.in.
[4]KIIT University, Bhubaneshwar, Odisha, India, Email: roshni.pradhanfcs@kiit.ac.in
*Corresponding Author

Received Date : August 04, 2021 Accepted Date : September 13, 2021 Published Date : October 06, 2021

ABSTRACT

This paper acts as a comprehensive analytical study of natural language processing (NLP) and provides a briefing of the most prominent astounding reforms of the field over a good chunk of time. It covers even the future research insights and most relevant features, which act as a result of the discussed concepts or research, until this paper's reading point. This paper starts with covering the most basic concepts of text cleaning, such as tokenization, the importance of stop words, etc., to concepts such as sequence modeling, speech recognition, the effect of quantum computing concepts in Natural Language Processing, and so on. The current development of deep neural networks, which is the current trend in artificial intelligence, always gives NLP a cutting-edge technology, also covered in this paper. This paper will also emphasize that it covers the broad area of explanations to the concepts to guide learners or researchers to have an excellent overall understanding of the field.

Key words: Natural Language Processing (NLP), Tokenization, Deep Learning, Recurrent Neural network (RNN), Long Short-Term Memory (LSTM), Bidirectional Recurrent Neural Network (BiRNN).

1. INTRODUCTION

Natural language processing [1] is an essential and profound concept in the field of artificial intelligence and mostly deals with areas related to human-computer interactions. This field has the most important applications, such as text processing, text summarization, document analysis, sentiment classification, etc., which will be further discussed in this paper, the applications section, and future research. Before discussing the most prominent features, this paper would like to state the name of the person who started the research in this field, who is none other than Alan Turing in his research article published in 1950 by the name “computing machinery and intelligence” [2].

Firstly, we would like to discuss concepts related to text processing and their importance in the future chain of topics going to be discussed. In recent times, many outstanding reforms have occurred in this field, no matter how it is analyzed from many points of view. For instance, right from traditional approaches like TF-IDF, parsing, regular expression matching, strings, etc., to word vectors, similarity scores with various parameters like Euclidean distance, cosine similarity, etc. This field has also witnessed ongoing breakthroughs like neural networks, machine translation, sentiment classification, emotional analysis of documents, and to enhance it, the percentage of importance of a particular statement in a specified context. On that note, the importance of word embedding in the applications mentioned above cannot be neglected, and what new application ideas and details it has given to the scientists to excel in the given area of proficiency should also be valued in its way. Its importance has grown so much that it would not be exaggerating to say that it currently holds a massive share in ongoing research in artificial intelligence. The world has also witnessed the wonders in this research and has accomplished successful tools like Alexa, Google Assistant, Siri of apple, etc., which gained profound importance and popularity in many households and the help it delivers in reforming technology bringing the future to the present. Speech-to-text converters helped in education, and machine translation helped the people connect globally even if they are of varied cultures, varied traditions, and varied languages. It has helped significantly in bringing the world near to us in its way. This breakthrough has also led to new and exciting research areas like the study of animal language, their speech, and the analysis of their emotions, which has helped many countries think about how technology can transform our understanding with respect to nature and its functionalities. It has also helped many animal researchers to show improvements in their research regarding animal behavior and functioning. This is just the beginning of the entire vast subject of NLP and what it can do in people’s lives and to get a clear picture of nature and its interactions with each other. It has removed the partitions among people with people, people, and wildlife and people with technologies as it has become a part of our lives, helping us develop day by day and to develop our understanding regarding it day by day. It is a special type of relationship between living beings, technology, and interdependencies if it is deeply understood. It can finally be said, and it is a beautiful enhancement done to the growing technology ever.

The remaining sections of the paper are organized as follows. Section 2 and 3 discuss the Text cleaning and Tokenization approaches of NLP. Section 4 is devoted to text preprocessing concepts. Postagging and noun chunks are covered in section 5. Word Embedding is covered in Section 6. Word vectors and similarity scores are discussed in Section 7. The main focus, which is the deep learning approach, is discussed in Section 8. Recurrent neural network (RNN), Long short-term memory (LSTM), and Bidirectional Recurrent neural networks (BRNN) are focused on sections 9, 10, and 11, respectively. Then Applications of NLP and more concepts associated with NLP are discussed in sections 12 and 13. Finally, the conclusion is given in section 14.

2. TEXT CLEANING

Text cleaning [3] is a crucial text preprocessing step on which many modern tools for NLP, such as machine translation, chatbots, speech recognition tools, etc., are highly dependent. In a word, we can say that it is the root of the efficiency of applications and research in the area. The most basic text cleaning steps require tokenization, lemmatization, stemming, removing stop words, etc. more profoundly, the topics or functions like lemmatization and stemming have a prominent impact on the normalization of text. Text cleaning results in almost what we can say as the feature matrix of machine learning or deep learning, and also it acts as the dominant player in producing accurate predictions or results. Now it is essential to explain the function of each of the above subfields of Text Cleaning.

3. TOKENIZATION

In simple words, tokenization [4] means to break the more complex sense into something more straightforward and more comfortable to work. In this process, practitioners sometimes remove mistakes or grammatically incorrect framing, which can harm the process of efficiency by forming a barrier for what we can say as an improper feature vector in machine learning terms. This process mainly contains two essential or widely used functions known as Word Tokenization and Sentence Tokenization.

3.1 Sentence Tokenization

This Tokenizer acts to divide or partition a chunk of text, paragraph, or document into each sentence. This partitioning is done based on a particular regular expression. Before this goes further, to give a simpler sense of regular expressions. Regular expressions can be understood as a pattern matcher in documents; what it does matches a specific pattern. For example, a sequence of four digits for which we design an expression matches any sequence of four-digit numbers in the entire document.

3.2 Word Tokenizer

This Tokenizer perhaps has got more significant importance in the step of text preprocessing as mostly in many documents, sentence tokenization is common, and in most of the cases, we get sentences as the input directly by the tool, so it becomes prominently important to divide them properly into word tokens. Here comes a tricky question or rather a basic fundamental question of why we are not considering a sentence as a whole, as a feature vector, and why we are fixated on words in determining tasks on various applications. The answer is rather simple: the same composition of words can give a different meaning depending on the phrasing.

Example:1) this dish is not bad but good. (Positive sense)
2) this dish is not good but bad. (Negative sense)
The above is related to the concept of sentiment classification, which this paper will discuss in a bit. The above example depicts the variance; the sentences are chosen as features rather than words. Now, to dive further into word tokenization, they are divided or partitioned according to individual needs, and many of the tools for this function follow different tokenization rules, for example, some partition according to the white spaces between the words, some with respect to commas, some give the user a choice of devising a regular expression according to its need, etc., as the applications of NLP are very much unique in each individual's approach and hence demand different variations of text processing which can be intuitively understood in the upcoming section of chunking of words, and further, in explanation of how the chunking can affect efficiency, strengthening the idea of the above-discussed scenario. Finally, words formed after tokenization can effectively take the role of an input feature vector to train data and form effective correlations in data to get valid output [5,43].

4. TEXT PROCESSING

It is the second major topic that controls the accuracy of the output. This is the step performed by an individual once he gets satisfied with the text Preprocessing steps. Text processing includes algorithms, tuning of hyperparameters, checking the algorithm's efficiency, and customizing the traditional algorithms to an individual's necessity. We would like to explain this section by giving frequent connections or relating to previous sections and how they are interconnected via various methods or applications [6].
Before we jump into discussing algorithms' applications, we need to understand the common words which are essential in many paths. There are previously standard machine learning approaches, but researchers have designed deep learning approaches for the same problems or applications, which will be explained briefly in the following content, and modern deep learning enhancements will be discussed. The common concept covered in both approaches is stop words.

4.1 Stop words

The stop words removal step is sometimes covered in the text preprocessing step but not always; sometimes, it may also be performed depending upon the efficiency of the algorithm without the removal of stop words. If necessary, only then the stop words are removed. In a simpler sense, stop words obstruct the correct formation of the Embedding vectors, or we can say the words on the removal of which the accuracy of the algorithm can increase are stop words. Generally, many standard machine learning libraries like NLTK and Spacy are a set of predefined words traditionally removed in applications such as text summarization, word tagging, etc. To intuitively understand their effect, we can say that in the routine sentences we speak, we often give a lot of junk information based on which the sentence meaning does not depend and on the removal of such words, the text processing as well as algorithms responsible for forming better correlations between words show improvement, as for many words tagging and word embedding algorithms may find best optimal correlations and might increase accuracy in a notable amount.Not only a predefined set, but there can also be different meanings of the word “stop words” depending upon the format of the document or the type of the document, for example, in the case of the following sentences.

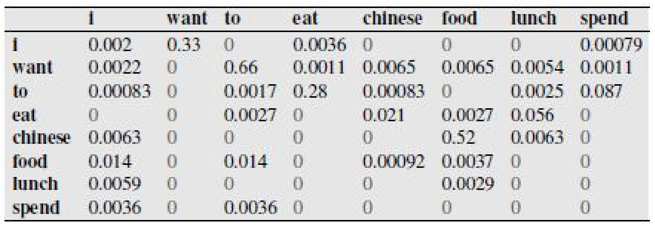
I am a senior analyst, and my email id is xyz@abcd.jkl.
In the above, while we use that sentence in a document where the sentence has no unique importance like a movie review by a customer who is the above person, in this case, we are only interested in rating but not email id. However, when we are trying to recommend a movie, the email id cannot be removed as it acts as an identity to the individual, in a similar sense for some documents; the stop words may not be the same as other documents and may even carry useful information. For further reading, there is a good understanding provided by this [1].

5. POSTAGGING AND NOUN CHUNK

Parts of speech tagging are a famous tool to assign each individual word with its part of speech. This tool has various versions of it provided by various organizations. Research is effectively going on to assign more precise parts of speech without any error, which is necessarily vital because parts of speech play an essential role in forming noun chunks which are useful if and only if we can effectively identify the nouns, adjectives, articles, etc. with Pos tagging we can also formulate our chunks of words with the help of regular expressions. It is instrumental in designing a regular expression based on the type of document for better results.
For example, the general chunking rule is:
{<DT>?<JJ>*<NN>+}
The above is a noun chunk expression that matches any chunk made of article+adjective+noun but, this is useful in the case of description documents or texts. In contrast, a regular expression of form {<NN>+} is much more helpful for a better correlation matrix in the case of scientific or experimental texts where there is no much weightage to the description. Hence, forming the right noun chunks is very much important in cases of text analysis, word embedding, named entity recognition, etc., further information on parsing and noun chunks is here [46], and information on how to extract noun chunks from large scale texts can be found here [45].

6. WORD EMBEDDING

To just brief, for now, what is the function of word embedding. In a more straightforward sense, word embedding does the same function as One Hot Encoding of sequences but not the same. However, during the initial times, one-hot encoding is used to denote sentences and sequences in vectors having value 1 wherever the word is present and 0 else. Later, due to an increase in the sparsity of feature matrix and increasing vocabulary usage and also the variation in the speech of languages, word embedding was designed based on specific features such as several adjectives, gender dictating words, phrasing, etc. which acts as different parameters for different tasks. This resulted in embedding a vector functioning as a feature vector in a finite number of dimensions for huge datasets, and one important thing is the parameters are chosen in such a way that there will be convergence while training the parameters in the least time. Now, **Figure 1** depicting the word embedding matrix with an example.



	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Figure 1: Image depicting an embedding matrix of a certain text.[7]

7. WORD VECTOR AND SIMILARITY SCORE

The idea of word vectors is well established in the paper by Mikolov et al. 2013 [5]. It is beautifully described how sentences containing words can be converted into word vectors by using two models.

7.1Continuous Bag of Words Model (CBOW)

In a sentence, if given a context of the sentence or given a certain number of neighborhood words (window size parameter), we need to predict the subject word of the context. A detailed representation is given in the form of a diagram below in **Figure 2.** As shown in the figure, an input vector (embedding vector) is given, which is the context, and there is the hidden projection matrix containing parameters, or in deep learning language, we call it as a hidden layer to learn the subject according to the context and finally we predict context and optimize it by using appropriate methods for training and optimization [23].

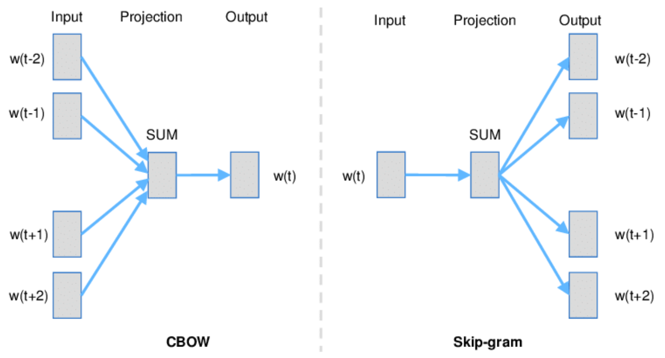


Figure 2: Depicting the CBOW and Skip Gram model [22].

7.2Skip Gram Model

Unlike the cbow model, given the word, we try to predict the context, and this model is very favorable when working on fewer amounts of data. When we have more data, we use the cbow model for appropriate reasons that when handling large data, we have enough information to train on as we can produce an efficient and optimized model. However, in some applications, though we have more data, we use the skip-gram model. This model has a clear edge for applications involving predictions of the context, word tagging task, and in some tasks of machine translation. Finally, we would like to conclude the word vectors by explaining similarity scores.

7.3 Similarity Scores

This is easiest to understand where we use the basic concept of Linear Algebra known as the dot product. To compute the similarity between two-word vectors, we take the dot product of vectors divided by the product of magnitudes of vectors given by the formula,

Similarity Score = (first word vector * second word vector)/(||first word vector|| * ||second word vector||) (1)

"*"- operation specifies the dot product.

"×"- operation denotes the normal multiplication operator.

8. DEEP LEARNING APPROACHES OF NLP

8.1Neural Networks

Neural networks are a revolution in the field of artificial intelligence, and many research papers like artificial neural networks by Geoffrey Hinton and many pioneers in the field like Andre Ng, Yann Le Cunn, etc., have produced unmatched outcomes from their efforts.

Feature Input: It is the input given to the neural network.
Hidden Layer: It may consist of many neurons identifying the underlying correlation between different dimensions of the features. This function is given in the basic form below.
Output Layer: The final prediction of the neural network.
Labels: The true classification from training data to compute loss and minimize error.
Consider a basic neural network as in **Figure 3**.

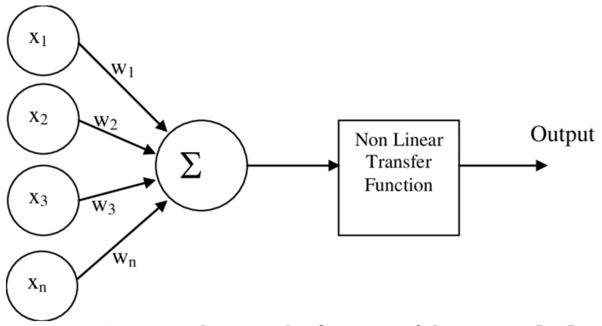


Figure 3: Showing the function of the neuron [21].

In **Figure 3**, $x_i^{(1)}$,$x_i^{(2)}$ and other input neurons together can be grouped as input vectors (feature vector) as $[x_i^{(1)}, x_i^{(2)}, x_i^{(3)},....,x_i^{(n)}]^T$ and this vector can be called x_i which is the ith training example and has 'n' dimensions or in this case of Figure 4 four features(n=4) to it and the circle in between input neuron layer and output layer is called the hidden unit, and every circular unit in the neural network is known as a neuron. For the hidden unit, the value is (W.T $*$ x^i), where the operator '*' used depicts the dot product between the two. After this, an activation function a_i^j is applied on the hidden unit value where "a_i^j" is the representation of the activation on the jth hidden unit in the ith layer of the network, here for this case the representation for the hidden unit activation is a_1^1. There are many types of activation functions such as tanh, relu, linear activation, sigmoid, etc., which are widely used.
Now, generalizing the above functions, we get to the following formulae of deep neural networks:
Forward Propagation:

X-feature vector of a particular training example
Z^L - the result of forwarding propagation of Z^{L-1} and W^L of the Lth layer.
A^L- activation layer L

$$Z^L = (W^L * X^L) \quad (2)$$
$$A^L = \text{activation} (Z^L) \quad (3)$$

MATRIX.T=>Transpose of the matrix in any of the given equations in this paper.
Above are the two essential equations used in feed-forward propagation. **Figure 4**, representing feed-forward neural networks. There is going to be a definition of the loss function and a discussion on backpropagation intuitively. X is the input feature matrix, and the deltas represent the gradients obtained through the backpropagation algorithm.

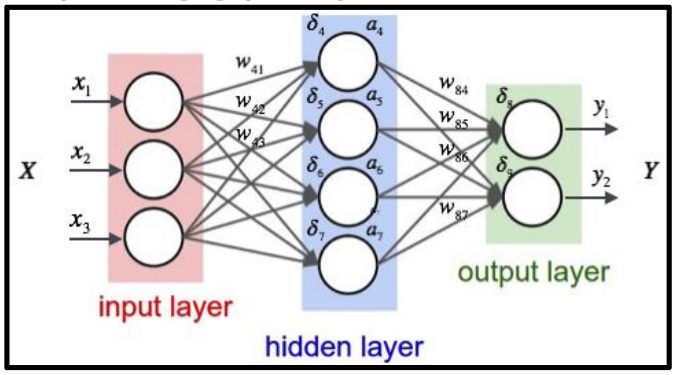


Figure 4: A deep neural network [47].

8.2 Back Propagation

Now, compute the gradients with respect to W's of every layer to perform optimization functions to reduce error. To understand this first, we will get to the equations part,

$\text{error}_{\text{output\_layer}} = \text{modulus(predictions-Y)}$ (4)
$\text{error}_{L} = (W^{L+1}.T * \text{error}_{L+1}) * g^{|}(Z^{L})$ (5)

Now, for the intuition part, we just need to think of it as a reverse calculation of forwarding propagation; hence, instead of propagating the input feature to the output layer, we make the error of the output layer and propagate the error according to the weights and taking derivative of activation function following the simple multivariate chain rule of calculus we considered the following cost function for above calculation of gradients,

$J = 0.5*((\text{predictions-Y})\text{^}2)$ (6)
Where, J - cost function, Y-Labels

The above cost function is known as the mean squared error function, and many other functions define cost accordingly. Further information on backpropagation or in-depth explanation can be understood from the paper [6]. There will not be any discussion on CNN in this paper. However, it is fascinating to note that many CNN algorithms have been applied to NLP, such as in speech recognition, machine translation, etc., to identify hidden patterns in data.

9. RECURRENT NEURAL NETWORK (RNN)

Recurrent neural networks [23] are designed to handle input at each neuron rather than input all of the feature vectors at once and entirely forward propagating and producing predictions. To understand this first, it would be better why simple neural networks fail in tasks, including sequences or word prediction or machine translation, etc. The primary disadvantage in simple neural networks is that we cannot handle input feature vectors of varying length and also, we cannot use partitioned input feature to predict the remaining partition of the feature, as in the tasks like word prediction; hence, it becomes very much necessary to design a neural network which can take input at each stage of forwarding propagation.
There are many types of RNN, such as, self RNN which takes only one single input and propagates it to generate a random sequence (at initial timesteps) but, certainly specific sequence (after multiple timesteps) whether it can be music, letters, or words, speech, etc. there is also an architecture by name LSTM which is very important in the field and also there will be information on deep Recurrent neural networks which is when we try to understand the activation functions in RNN.

9.1 Basic RNN

The concept of RNN is very much easily visualized as the given **Figure 5** below.

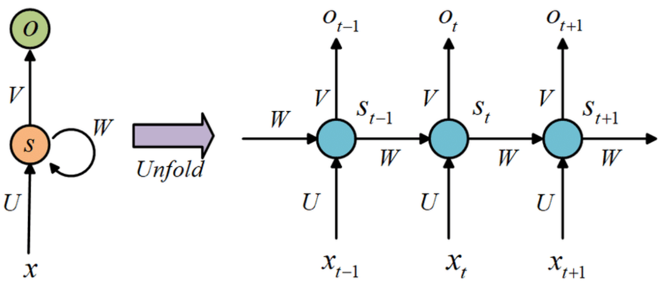


Figure 5: Basic Recurrent Neural Network [23].

X_t represents the tth time step of the feature vector, or it could be understood that if there is a sentence,
Input feature (x) = This is a flower.
then, x_1=" This" which is converted to word embedding during input
U denotes the input weight matrix.
W denotes the weight matrix.
S denotes the activation function of neurons.
V is the output weight matrix.

Hence, it can be seen that simple neural networks cannot handle the tasks as RNN, and it can be understood to have been designed to handle such specific tasks, and for much deeper insights in RNN, this research article [8] will be helpful. This paper will briefly discuss deep recurrent neural networks and further discuss LSTM with their equations for the next section.

9.2 Deep RNN

As said previously, this is a brief explanation of deep RNN and just is the discussion of one more type of RNN. It could be visualized as a stacking up of multiple layers of RNN on top of each other as shown in **Figure 6,** and to explain the things further, it is just that the output is now passed as input to the next layer with a certain activation function to just understand parameters quickly, there will be individual weight matrices for input feature x, for activation result obtained during the previous timestep, for each layer, and weight matrix for each vertical arrow (upward transition) between layers and output [18].
NOTE: The weight matrix for the entire horizontal layer is the same; similarly, for the same layer transition, the weight matrix is the same for that transition anywhere for that transition.

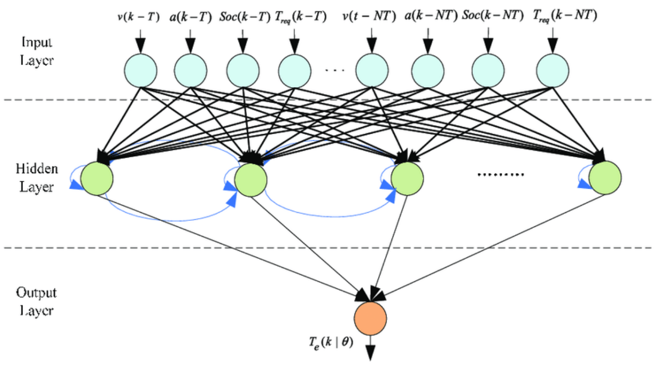


Figure 6: A deep recurrent neural network [18].

Now, as discussed above, every application has its own architecture for training the data. For example, Sentiment classification uses many to one architecture; machine translation uses many to many to architecture, which will be covered in the coming sections. However, the material covered here is only the basic version of the explanation, and in many cases, deep RNN is not used effectively, consisting of many layers unless the task is complex as the handling of the weight matrix (parameter matrix) would become tough and time-consuming.

10. LONGSHORT-TERM MEMORY (LSTM)

In handling sequences of greater length, the RNN may not effectively capture the impact created on preceding word on future estimate or prediction of the word as there is continuous updating of parameters at each and individual timestep, so it would be very much beneficial if the system could remember the past dependencies for a greater amount of timesteps to establish a well and good prediction of results. In this section, many equations from previous concepts are discussed, and firstly this section discusses the update equation of RNN.

$a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$ (7)

$y^{<t>} = (W_y * a^{<t>}) + b_y$ (8)

In every upcoming equation superscript on each symbol denotes the time step of the sequence. The subscript denotes to which value the parameter belongs to. But the above equation is not efficient in capturing dependencies over a longer period of time.

NOTE: 'g' denotes activation function and b_a denotes the bias of output, $x^{<t>}$ denotes input at timestep 't' and finally one more representation is,

$W_a[a^{<t-1>}, x^{<t>}] = W_{aa} * a^{<t-1>} + W_{ax} * x^{<t>}$(9)

W_{aa}=weight matrix belonging to output

W_{ax}=weight matrix corresponding to input feature vector.

* - denotes the dot product.

Now, to tackle the above issue, there is a need to construct an object for handling when to update the parameter for preserving the dependency and when to forget the previous dependency so as to delete the unwanted relational dependencies and also a final tweaking parameter for updating the output at each timestep so as to not copy the result directly for next time step.So, finally, the following equations act as an add-on to the above fundamental equation.

$S^{<t>} = \tanh(W_a[a^{<t-1>}, x^{<t>}] + b_s)$ (10)

$\text{gamma}_u = \text{sigmoid}(W_a[a^{<t-1>}, x^{<t>}] + b_u)$ (11)

$\text{gamma}_f = \text{sigmoid}(W_a[a^{<t-1>}, x^{<t>}] + b_f)$ (12)

$\text{gamma}_o = \text{sigmoid}(W_a[a^{<t-1>}, x^{<t>}] + b_o)$ (13)

$C^{<t>} = \text{gamma}_u * S^{<t>} + \text{gamma}_f * C^{<t-1>}$ (14)

$a^{<t>} = \text{gamma}_o * C^{<t>}$ (15)

In the above listed formulae,

Dot(.) - element wise matrix multiplication

Update gate-gamma(u) - responsible for when to update the current "C"

Forget gate-gamma(f) - responsible for when to forget the update for the current time step

Output gate-gamma(o) - amount of information needs to be passed on for next time step

To simply brief the above, it is seen that every gate is using the sigmoid function as the function becomes either 0 or 1 over a large range of values, and we need such a type of function to either update the previous or forget the previous to the current time step, as far as the output gate is concerned it decides how much of the current information needs to be followed on leaving room for the input feature for next time step to have a significant representation of the prediction. If clearly observed, it can be sensed that the forget gate and update gate has the opposite effect to each other, i.e., when for forget gate is nearly 1, we do not consider the current feature will have the least impact on the change of the value of activation which will be carried on to next LSTM and if an update is nearly 1 then the current value of "C" is updated. A simple depiction of LSTM cells is given below in **Figure 7**, explaining the features and mechanisms going inside it. This idea or concept can be further enhanced in the research article [9], which clearly explains the effects of different activation functions.

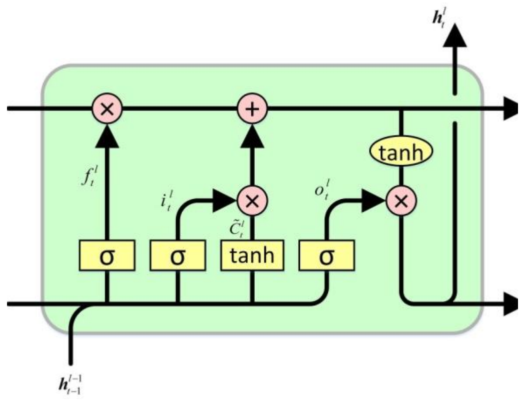


Figure 7: Block diagram of LSTM [48]

Gated recurrent unit cell is also depicted in **Figure 8** as it is also familiar with LSTM, and in some applications, gated recurrent units work better than LSTM, and if interested in-depth insights of GRU can be found here [9].

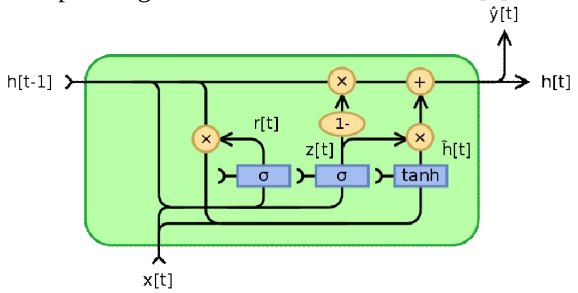


Figure 8: Block diagram of Gated Recurrent Unit (GRU) [49].

11. BIDIRECTIONAL RECURRENT NEURAL NETWORK (BRNN)

Bidirectional Recurrent neural networks are one of the important neural networks as the above architectures are unidirectional it becomes to predict output at a time step with respect to a future wording, and in such scenarios, the BRNN is helpful as it takes both forward propagation activation and forward propagation activation in the reverse direction in predicting output for each and individual output, it performs better for sequences of greater length and sequences of high interdependency [10]. The basic block diagram which shows the functionality of BRNN is depicted in **Figure 9**.

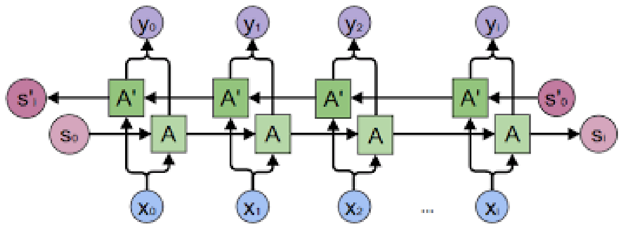


Figure 9: Showing a basic BRNN [50].

12. APPLICATIONS

The above-given theory and insights are very helpful in the following discussed applications, as they can be inferred as the transformation of the NLP field.

12.1Word Tagging

Suppose any time searched online about a question related to words, whether it might be technical or business-related or anything else, it will be suggested or shown as blocks or in any other format. The Word Tagging model thoroughly explains this type of correlation between the word searches and popping suggestions on related articles. This section comes from highlighting the previously discussed sections and their importance.

To start, this model first takes a raw document related to many subjects or topics, which is a common step in many Text processing applications, and then it performs basic and needed text cleaning and text processing operations, as discussed above in the previous sections. Then the processed text is converted into a feature matrix in the form of word embedding, and each and the individual sentence is given a <POS> and <EOS> tags determining the start and the end of the sentence. Now, the similarity scores between the sentences are found following any of the CBOW models or Skip Gram model by using a traditional machine learning library like WORD2VEC, which vectorizes the word embedding into vectors by capturing dependencies of all contexts of every word present in the "VOCABULARY" of the data matrix, OR the above can also be determined by using latest deep learning models like BERT which is used in performing the Named entity recognition task efficiently. The above methods have been used until now, but depending upon the scale of the task, the usage of the method is determined.

12.2Text Summarization

Text summarization is another important application of natural language processing that requires proper parameters applied so that there is an effective summary of the text and avoiding minimal errors as possible because sometimes unimportant things crop up due to less rigorous extraction of the text and can be corrected by following the opposite steps. Now, to discuss the most important and fundamental tool in the extraction of the text is the TF IDF model, which works on basic word frequency count across various documents and perform the similarity score between the sentences just like we do for words by word embedding vectorizer functions so as not only to vectorize words but also sentences as a whole. The important part here is to understand how the extraction is taking place and, more precisely, how the TF IDF algorithm is working; understanding this, we follow two steps that primarily calculate the word frequency individually and represent them as values, If more the frequency of words the more is the importance of the word in the document, but words such as "and", "the", "an" etc. are used extensively in any type of document and to reduce their impact there is a second step to penalize such scores by multiplying them with a factor which is calculating the ratio between the total number of documents to the number of documents in which the word appeared and finally apply logarithm function to base 2 to the result for controlled penalization. The same is given by the following equations,

Term frequency = (number of times the word appeared in the text)/(total number of words in the text) (16)

Inverse Document Frequency = log{(total number of the document)/(the number of documents the word appeared)}(17)

After obtaining the feature matrix, the individual conditional probabilities are considered following the Bayesian inference between sentences of the given document, and the sentence vectors are given as input for calculation of similarity scores, and further, the most relevant chunk of sentences with high relative similarity score is considered as an output. The further insights of weights for calculation of embedding can be found here [2] helpful in the summarization of text and also one interesting paper that covers much more depth here [4] and also another good reading here [3].

12.3Machine translation and Speech Recognition

Machine translation is a sequence-to-sequence model of RNN, and instead of outputting the sequence at each and individual time step, the architecture is designed to first memorize the entire input feature vector and output each output word vector after memorizing the input feature. This model works on maximizing the Bayesian probability of the output conditional on input, on covering a short formula used here, it is important to see the general formula here below,

If two events A and B are occurring with probabilities P(A) and P(B) respectively, then the probability of occurring of A given B is given by P(A|B), which is,

$$P(A|B) = (P(A)/P(B)) * P(B|A) \qquad (18)$$

The above extended to involve more variables (events) conditional on multiple events, which is the crux for the task,

$$P(A,B|C) = P(A|C) * P(B|A,C) \qquad (19)$$

Assuming B here to be output after the timestep of first timestep output A and C to be the input feature as a whole then it is clear that the above can be generalized and used for getting output at n^{th} time step as follows,

$$P(y^{<1>},y^{<2>},y^{<3>},y^{<4>},......,y^{<n-1>},y^{<n>}|x) = P(y^{<1>}|x)*P(y^{<2>}|x,y^{<1>})........*P(y^{<n>}|x,y^{<1>},y^{<2>},y^{<3>},......,y^{<n-1>}) \qquad (20)$$

Now, there is a formula to find the probability of getting n<th> output in terms of the conditional probability of previous individual timestep's output and the input feature vector 'x'. If the conditional probability at each step is maximized, we get the absolute translation of the input sentence; this approach is known as a greedy approach.

But the above approach may fail to produce the most sensible translation, for suppose let the correct translation for input be, "The plan is going to get executed tomorrow," and suppose another translation with the overall average probability being highest be, "The plan will be executed tomorrow."
Compared to the first sentence, where a greedy approach is followed at each individual time step, getting overall probability over multiple options is highest; for this, an algorithm known as Beam Search is used.

12.4Beam Search

Instead of always choosing the word with the highest probability, we choose a fixed number of different words depicting the top "n" number of conditional probabilities arranged in descending order of magnitudes of conditional probabilities [10]. For example,

For $y^{<1>}$ the top "n" number of outputs are chosen, which implies, there are "n" copies of the network made in which $y^{<1>}$ takes a different value, i.e., a different word is formed as output. The same can be viewed from **Figure 10**.
Now, for $y^{<2>}$ we again calculate,

Top "n" values of $y^{<2>}$ which maximizes the conditional probability $P(y^{<1>},y^{<2>}|x)$ for each and individual network of "n" networks present by which there are n square of probabilities present out of which , the top "n" conditional probabilities are selected and again "n" individual copies of architecture are made taking $y^{<1>}$ and $y^{<2>}$ as individual values of order respectively.
And the process is continued,
Here, "n" is known as beamwidth.
So, the final objective can be defined as,
$y = \text{argmax}_y \{\text{product}_{t=[1,Ly]}(P(y^{<t>}|x,y^{<1>},y^{<1>},\ldots,y^{<1>}))\}$ (21)

Rather than following the above objective for selecting the best y, It would be better to take the logarithm of the above objective to predict y as in both objectives maximizing y maximizes the function, and further, we can normalize the function by the length of output "L_y". Hence the function now becomes,
$y=\text{argmax}_y(1/L_y)*\{\text{summation}_{t=[1,Ly]}(P(y^{<t>}|x,y^{<1>},y^{<1>},\ldots,y^{<1>}))\}$ (22)

In an above-defined way, we can perform machine translation, and for any further reading on this concept, this paper [11] is beneficial and a summary here [12].

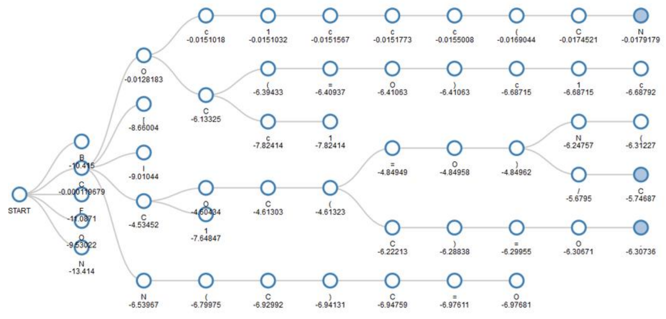


Figure 10: Depicting beam search with beam width = 5 [20].

12.5Attention Model

Now, it is important to mention a concept known as the attention model, which comes under machine translation, and also, this concept is used in speech recognition. This concept pitches in when there is a sequence of greater length then entirely memorizing the input during the first part of the architecture of LSTM (ENCODING), and after that completely outputting the sequence (DECODING) will become tough, and also the results would not be appropriate, to tackle this the model divides entire sequence into parts and performs beam search with respect to a new parameter known as 'context," and there is a small change in architecture as compared to the previous as shown in **Figure 11**.

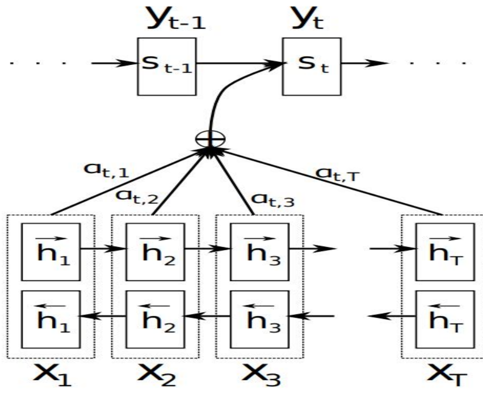

y_{t-1}
y_t
s_{t-1}
s_t
$a_{t,1}$
$a_{t,2}$
$a_{t,3}$
$a_{t,T}$
$\overrightarrow{h}_1$ $\overrightarrow{h}_2$ $\overrightarrow{h}_3$ $\overrightarrow{h}_T$
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ $\overleftarrow{h}_3$ $\overleftarrow{h}_T$
X_1 X_2 X_3 X_T

Figure 11: Depicting the attention mechanism [19].

The above is a BRNN (bi-directional recurrent neural network) creating a "context" which is a weighted sum of outputs from BRNN whose weights are known as "attention weights" and following equations are of help to understand above,
$\text{summation}_{t=[1,T]}$ (alpha<t,t'>) = 1 (23)
Now, to find out context "C", the following is carried out,

$C^{<t>} = \text{summation}_{t=[1,T]}$ (alpha<t,t'> * $O^{<t'>}$) (24)
Now, to calculate $y^{<t>}$, there can be a neural network used taking previous output $y^{<t-1>}$ and $C^{<t>}$as inputs and $y^{<t>}$ as output. This is suggested as it is exactly unknown the relation between input and output. So, it is easier for a neural network to identify the mathematical relation between them. There is a concept known as the Bleu score, which is explained in [13] and is considered for evaluating models.

Speech Recognition is just the application of the above concepts, but instead, we deal with audio data, and there is an algorithm known as Transformer and many further concepts. There is a good paper on this concept which gives good intuition here [14]. The attention model is also used to adapt to the context and to summarize a series, an excellent paper on video summarization using the model is here [42].

12.6Quantum encoding and decoding

The current research in artificial intelligence is becoming rigorous with respect to Quantum Computing; already there many applications of quantum computing, such as the QAOA algorithm [15], which is extensively used in the optimization of algorithms, and recently there is an article [17] depicting the quantum computing implementation for encoding and decoding part of neural network architecture and also there is increasing use of Grover's search algorithm [16] in this field. The further understanding of this very much requires a thorough understanding of Quantum Computing and Quantum Information Theory.

13. SOME MORE CONCEPTS

There are some of the important concepts covered in this paper. A few other mentions are similar to the above concepts or can be like good reading to better understand and instigate deep intuition both application and concept. They are the following,

Text clustering is an important algorithm that has meaning in its name itself. It clusters out the data in a general text when it is composed as a description of many types in common [31]. Text clustering knowledge can also be helpful in other subdivisions of the field, which are the dialogue systems [26, 30]. To completely understand the practical implementation tricks of Neural Machine Translation [24], especially; how it deals with embedding layers and how to handle large batches of input to the model. As there is a sequence-to-sequence model used in neural machine translation, but the sequence is getting handled at the level of words, and recently this sequence-to-sequence model is also being operated at the character level, which is beneficial in its own strong territory. This operation at the character level is well explained in the article [25].

Sentiment classification [37, 38] is another important application that is very helpful in analyzing the text's polarity. The machine learning implementation for sentiment classification [39] explains how gated recurrent units can be used for sentiment classification.Convolutional neural networks are one of the major influencers in coming up with different approaches for tasks such as clustering, speech recognition, etc. and there is a good number of papers, very much beneficial in grasping the concept of architecture for the various tasks in NLP, which are [27, 28, 29].

Information retrieval is an important major division of text processing as it gets out the important crux in the entire document, to address the previous, sometimes in a 100 pages document, there can be only 1 page of useful information which can be meaningful for the purpose, hence to solve this, there is a book [32] which can be treated as the complete introduction to information retrieval. A survey paper [33] describes the different methods for information retrieval and filtering methods. Understanding Neural network implementations in information retrieval can be found here [34]. Information retrieval done by the tokenization technique is explained here [44]. Self-organizing maps [36] are often considered as the clustering algorithm and perhaps more of identifying different chunks in the given data, which generally suits in case of text processing, describing this is an implementation of SOM in Information Retrieval in the paper here [35].

Optimization is a very important part of understanding neural networks and also many applications of it which decides the efficiency and accuracy of algorithms and parameters and thus is very much useful to learn about fundamental optimization algorithms like gradient descent, RMS prop, Adam, Adaboost, momentum, etc. [40,41]. Feature extraction is very important in machine learning and deep learning. The feature extraction for text categorization and getting a good understanding of text categorization are two useful aspects in that direction. BERT,which became successful in document classification, is the main pre-trained model used extensively in handling complex tasks, and hence relevant knowledge is necessary.

13. CONCLUSION

It can be said that Natural language processing is a massive field with a lot of cross-platform knowledge implementation, and the growing demand for the applications in current products makes it even more special. NLP is a complex concept with infinite dimensions, and one can always obtain more based on his efforts and creativity. Currently, the computationally efficient quantum computing ideas are being applied into various types of fields, and one can be sure that it is going to create a greater impact on all applications which are unable to perform well or are not being used because of being computationally expensive, but now with growing technology such as quantum optimization makes it possible for such applications to realize, and one major chunk of applications definitely belong to NLP. Thus, it is sure that in the coming day's text, speech, sequence, etc., based applications will come up with much more ease, and there will be an increase in the growth of practical applications of the theory proposed. Finally, Natural language processing depicts the human-computer relationship beautifully and certainly the positive growth of technology and human-machine interaction.

ACKNOWLEDGEMENT
Compliance with Ethical Standards:
Funding: This study was not funded by any person or organization.
Conflict of Interest: Authors declare that they have no conflict of interest.

REFERENCES

1.RasmitaRautray, Rakesh Chandra Balabantaray, Anisha Bhardwaj, Document summarization using sentence features, International Journal of Information Retrieval Research ,2015.

2. Rakesh Chandra Balabantaray, DK Sahoo, B Sahoo, M Swain,Text summarization using term weights, International Journal of Computer Applications,2012.

3.Weiguo Fan, Linda Wallace, Stephanie Rich, and Zhongju Zhang, "Tapping into the Power of Text Mining", Journal of ACM, Blacksburg, 2005.

4. Gupta, V., Lehal ,G. S.,A Survey of Text Summarization Extractive Techniques,JOURNAL OF EMERGING TECHNOLOGIES IN WEB INTELLIGENCE,Vol.2,No.3,2010.

5.Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. In Proceedings of Workshop at ICLR, 2013a.

6.Rumelhart, D. E., Hinton, G. E., and Williams, R. J.Learning representations by back-propagating errors.*Nature*, 323, 533--536.1986.

7.Jurafsky, Daniel & Martin, James. (2008). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.

8. Alex Graves. Generating sequences with recurrent neural networks. CoRR, abs/1308.0850, 2013.

9. Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term memory. Neural computation, 9(8):1735–1780, 1997.

10. A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM networks. In Proc. Int. Joint Conf. on Neural Networks IJCNN 2005, 2005.

11. Freitag, M., Al-Onaizan, Y.,Beam Search strategies for neural machine translation,arXiv preprint arXiv:1702.01806, 2017

12.Garg,A.,Agarwal,M.,Machine Translation : a literature review,arXiv preprint arXiv:1901.01122, 2018.

13.Papineni, Kishore &Roukos, Salim & Ward, Todd & Zhu, Wei Jing. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. 10.3115/1073083.1073135.

14.AwniHannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, Andrew Y Ng, Deep Speech : scaling up end-to-end speech recognition.

15. Farhi, E.,Goldstone, J.,Gutmann, S.,A Quantum Approximate Optimization Algorithm,arXiv preprint arXiv:1411.4028, 2014.

16. L. Grover. A fast quantum mechanical algorithm for database search. In Proc. 28th STOC, pages 212–219, Philadelphia, Pennsylvania, 1996. ACM Press.

17. Bausch, J.,Subramanian, S.,Piddock, S.,A quantum search decoder for natural language processing, arXiv preprint arXiv:1909.05023, 2019.

18. Kong, Huifang& Fang, Yao & Fan, Lei & Wang, Hai & Zhang, Xiaoxue& Hu, Jie. (2019). A novel torque distribution strategy based on deep recurrent neural network for parallel hybrid electric vehicle. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2917545.

19.Bahdanau, D., Cho, K. H., &Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. Paper presented at 3rd International Conference on Learning Representations, ICLR 2015, San Diego, United States.

20. Liu, Bowen &Ramsundar, Bharath &Kawthekar, Prasad & Shi, Jade & Gomes, Joseph & Nguyen, Quang & Ho, Stephen & Sloane, Jack &Wender, Paul & Pande, Vijay. (2017). Retrosynthetic Reaction Prediction Using Neural Sequence-to-Sequence Models. ACS Central Science. 3. 10.1021/acscentsci.7b00303.

21. Long, Dan &Wuest, S. & Williams, John &Rauwendaal, Randall & Bailey, M.. (2010). Contour Planting: A Strategy to Reduce Soil Erosion on Steep Slopes.

22.Landthaler, Joerg &Waltl, Bernhard &Huth, Dominik & Braun, Daniel &Matthes, Florian & Stocker, Christoph & Geiger, Thomas. (2017). Extending Thesauri Using Word Embeddings and the Intersection Method.

23. Zhu, Juncheng& Yang, Zhile & Mourshed, Monjur& Guo, Yuanjun& Zhou, Yimin& Chang, Yan & Wei, Yanjie& Feng, Shengzhong. (2019). Electric Vehicle Charging Load Forecasting: A Comparative Study of Deep Learning Approaches. Energies. 12. 2692. 10.3390/en12142692.

24.Neishi, M., Sakuma, J., Tohda, S., Ishiwatari, S., Yoshinaga, N., & Toyoda, M. (2017). A Bag of Useful Tricks for Practical Neural Machine Translation: Embedding Layer Initialization and Large Batch Size. *WAT@IJCNLP*.

25. Zhang, H., Li, J., Ji, Y., & Yue, H. (2016). A character-level sequence-to-sequence method for subtitle learning. 2016 IEEE 14th International Conference on Industrial Informatics (INDIN), 780-783.

26. Ren, D., Cai, Y., Chan, W.H., & Li, Z. (2018). A Clustering Based Adaptive Sequence-to-Sequence Model for Dialogue Systems. *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 775-781.

27.Allamanis, M., Peng, H., & Sutton, C.A. (2016). A Convolutional Attention Network for Extreme Summarization of Source Code. *ICML*.

28. Gehring, J., Auli, M., Grangier, D., & Dauphin, Y. (2017). A Convolutional Encoder Model for Neural Machine Translation. *ArXiv, abs/1611.02344*.

29. Xing, Y., Xiao, C., Wu, Y., & Ding, Z. (2018). A Convolutional Neural Network for Aspect Sentiment Classification. *IJPRAI, 33*, 1959046:1-1959046:13.

30. Aggarwal, C. C., &Zhai, C. (2012). A survey of text clustering algorithms. In *Mining text data* (pp. 77-128). Springer, Boston, MA.

31. Jing, L., Ng, M. K., & Huang, J. Z. (2010). Knowledge-based vector space model for text clustering. *Knowledge and information systems*, *25*(1), 35-55.

32. Manning, C. D., Raghavan, P., &Schütze, H. (2008). *Introduction to information retrieval*. Cambridge university press.

33.Faloutsos, C., &Oard, D. W. (1998). *A survey of information retrieval and filtering methods*.

34. Mitra, B., &Craswell, N. (2017). Neural models for information retrieval. *arXiv preprint arXiv:1705.01509*.

35. Lin, X., Soergel, D., &Marchionini, G. (1991, September). A self-organizing semantic map for information retrieval. In *Proceedings of the 14th annual international*

ACM SIGIR conference on research and development in information retrieval (pp. 262-269).
36.Kohonen, T. (1997, June). Exploration of very large databases by self-organizing maps. In *Proceedings of international conference on neural networks (icnn'97)* (Vol. 1, pp. PL1-PL6). IEEE.
37. Xia, R., Zong, C., & Li, S. (2011). Ensemble of feature sets and classification algorithms for sentiment classification. *Information sciences*, *181*(6), 1138-1152.
38. Pang, B., Lee, L., &Vaithyanathan, S. (2002, July). Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10* (pp. 79-86). Association for Computational Linguistics.
39. Tang, D., Qin, B., & Liu, T. (2015, September). Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 1422-1432).
40. Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.
41. Le, Q. V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., & Ng, A. Y. (2011). On optimization methods for deep learning.
42. Ma, Y. F., Lu, L., Zhang, H. J., & Li, M. (2002, December). A user attention model for video summarization. In *Proceedings of the tenth ACM international conference on Multimedia* (pp. 533-542).
43. Nayak, A. S., Kanive, A. P., Chandavekar, N., &Balasubramani, R. (2016). Survey on preprocessing techniques for text mining. *International Journal Of Engineering And Computer Science, ISSN*, 2319-7242.
44. Singh, V., & Saini, B. (2014). An Effective tokenization algorithm for information retrieval systems. *Department of Computer Engineering, National Institute of Technology Kurukshetra, Haryana, India*.
45. Chen, K. H., & Chen, H. H. (1994, June). Extracting noun phrases from large-scale texts: A hybrid approach and its automatic evaluation. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics* (pp. 234-241). Association for Computational Linguistics.
46. Abney, S. P. (1991). Parsing by chunks. In *Principle-based parsing* (pp. 257-278). Springer, Dordrecht.
47. Deep neural network tutorial [Online]. Avail: https://miro.medium.com/max/958/1*QVIyc5HnGDWTNX3m-nIm9w.png
48. Long Short-Term Memory tutorial [Online] Available: https://i.stack.imgur.com/RHNrZ.jpg
49. Gated Recurrent Unit tutorial [Online] Available: https://cdnimages1.medium.com/freeze/max/1000/1*OBCui-SbIRUtlBgWkgQUlw.png?q=2
50. Bidirectional Recurrent neural network tutorial [Online] Available:http://www.easy-tensorflow.com/tftutorials/recurrent-neural-networks/bidirectional-rnn-for-classification