REVIEW OF VARIOUS KEYWORD EXTRACTION TECHNIQUES AS A SEQUENCE LABELLING PROBLEM

[1]Ritik Mehta, [2]Ramneek Kaur, [3]Meenal Sarwaiya, [4]Goonjan Jain

[1]Final Year, Bachelor of Technology, Delhi Technological University
[2] Final Year, Bachelor of Technology, Delhi Technological University
[3] Final Year, Bachelor of Technology, Delhi Technological University
[4]Assistant Professor, Delhi Technological University

Email: [1]mehtaritik11@gmail.com, [2]ramneek983@gmail.com, [3]meenalsarwaiya008@gmail.com, [4]goonjanjain@dtu.ac.in
Contact: [1](+91)8527315621., [2](+91) 9911094949, [3](+91) 9319756736, [4](+91)9354461533

Abstract:. Keywords and keyphrases provide the true essence and highly summative information about a text which helps us in understanding and organizing the content. With the substantial growth in the data on a daily basis, it is important to organize and analyze the data. Keywords and keyphrases provide us with a brief and to-the-point summary of the documents, which gives the high-level specification of the document. Extracting the keywords and keyphrases manually is a tedious task because of which automatic keyword extraction has received considerable attention in recent decades because it provides crucial information of the text without manual labor. There has been a considerable amount of research in the areas of keyword extraction in the past. This paper is a study of various unsupervised and supervised techniques put forward by various researchers for keyword extraction and compares these techniques on the basis of precision, recall, and F1 score, calculated using BIO Tagging as a sequence labeling task on SamEval 2017 dataset which contains data belonging to various scientific publications.

Index terms: *BIO Tagging, Keyword Extraction, Manhattan Distance Based Algorithm, Transformer*

I. INTRODUCTION

In today's world where there is a continuous development of the information age, the content in the world is rapidly increasing making it a hard task to manage the growing large-scale information. It has become impossible to process this extensive data manually due to the continuously increasing data and the inability to process this data manually. This creates the need for automatic keyword and keyphrase extraction techniques which can replace the manual labor to summarize the data and convey the context without losing the originality and giving the summary of the text.

Keyword extraction is considered one of the pivotal tasks in data analysis. A key phrase denotes a multi-word lexeme and a keyword denotes a single word that best highlights the main context of the document and provides us with the main features, concept, theme, etc. of the document. Apposite keywords provide a concise summary of the document and help us in easily organizing the document and retrieving them based on context. Automatic summarization, filtering, indexing, information visualization, and topic detection, all are tasks that depend on keyword extraction. Finding the significant and suitable keywords from the large document which best convey the meaning of the text is a challenging task. The automatic keyword extraction techniques help us save time and effort.

Machine Learning approaches for automatic keyword extraction can be divided into supervised and unsupervised methods. Supervised techniques are trained on data to identify if a phrase is a key phrase or not. These techniques require a lot of labeled training data to give appropriate results which is a challenging task. Due to the large amount of data generated every day it becomes difficult to create labeled datasets as it would require a large amount of manual labor. This makes unsupervised methods without human intervention really essential. Unsupervised techniques do not require training data. It determines keywords and keyphrases using the properties of text in the document. Supervised learning techniques give better results than unsupervised techniques but the collection of the labeled corpus is a difficult task.

This paper is a comparative study of various supervised and unsupervised techniques of keyword extraction which efficiently help in extracting the keywords. In the sections below we have discussed the various techniques put forward by various researchers in the past for automatic keyword extraction, for instance, unsupervised learning methods like Text Rank [1] and YAKE [2] and supervised learning methods like Bidirectional LSTM-CRF and Bidirectional LSTM-CNN's-CRF and performed a comparative study of these algorithms as discussed in the Experiment and Results section below.

II. LITERATURE REVIEW

Over the past few years, Natural Language Processing has found many useful applications in solving real-world problems. Some of the common areas of its application are voice-assistants, text autocorrect and autocomplete, chatbots, and so on. One more such area is keyword extraction. There is a significant growth in the number of text documents available to users on various internet platforms. Hence, it is very essential to devise methods that can be used to summarize these documents efficiently and effectively. Keyword extraction can be defined as the automatic identification of terms that best illustrate the subject of the document. Many models and techniques have been proposed for keyword extraction. But each of these models has been trained and tested on different datasets. Some are trained on very large datasets while others on relatively smaller datasets. So, it is impossible to predict which would be the correct model to be taken as a base model to achieve success. Ideally, we would like to select, before starting, which model we should use.

In 2020, Ricardo Campos proposed a method called YAKE [2] which is a lightweight unsupervised keyword extraction method. YAKE uses statistical text features extracted from a document to select the most relevant keywords. Ricardo was significantly able to outperform the other unsupervised methods available at the time.

Furthermore, in September 2020, Mingxi Zhang, Xuemin Li, Shuibo Yue, and Liuqian Yang conducted an empirical study on TextRank for finding optimal parameter settings for keyword extraction [3]. They came to the conclusion that TextRank produces the best results when setting co-occurrence window size w=3, iteration number t=20, decay factor=0.9, and rank k=10 respectively. They also showed that the results are independent of text length.

Also, In January 2019, Boshko Koloski, Senja Pollak, Blaz Skrlj, and Matej Martinc extended neural keyword extraction with TFIDF tagset matching [4]. They took the keywords returned by a neural network and, subsequently, complemented the returned keyword list by adding the missing keywords to accomplish the set goal of n keywords. The added keywords were selected by taking the top-ranked candidates from the TFIDF tagset matching.

Ankit Aich, Amit Dutta and Aruna Chakraborty, in March 2018, proposed a scaled conjugate gradient backpropagation algorithm for keyword extraction [5]. In a backpropagation algorithm, weights are usually adjusted in the steepest descent direction. The problem with this approach is that although the performance function decreases fastest along with the negative of the gradient, this does not necessarily produce the fastest convergence. The conjugate gradient algorithm proposed by Ankit Aich is more optimized as a search is proposed along the conjugate direction, which usually provides the user with faster convergence than the steepest descent directions.

In May 2018, Saroj Kr. Biswas in the paper "A graph based keyword extraction model using collective node weight" [6] proposed a model that considers frequency, centrality, position and strength of neighbors of a node to calculate the importance of the node. Here, the node represents a token. The keywords were extracted using NE Rank Centrality with degree as a tie-breaker. Saroj also compared the model with two graph based keyword extraction models - TKG and KEGBA, and a non-graph based keyword extraction technique - TF-IDF, and concluded that her proposed model outperforms the other three models on the datasets considered.

Furthermore, in June 2021, N. Nikzad–Khasmakhi and many other researchers worked on Phaseformer [7] which is a multimodal key-phrase extraction using Transformer and Graph Embedding. They started by finding the textual representation of all the words in the dataset using the BERT transformer which helped in providing a better understanding to evaluate the semantic similarity between words. They then calculated the structure vector of every word based on three graph embedding algorithms, and subsequently, concatenated the text information and structure information and create a single representation of each word. At last, they considered keyphrase extraction as a sequence labeling task and label each word based on the BIO tagging scheme through a fully connected layer to classify the word. On the three datasets they considered, Phaseformer significantly outperformed single-modality methods.

It is fair to say that a wide range of keyword extraction techniques are now available. Our objective is to compare these techniques using a Manhattan Distance-Based Approach [8] which is a modified version of the traditional distance-based approach [9], and rank the models on the basis of their ability to extract the keywords from a text.

III. DATASET AND PREPROCESSING

The dataset used in this paper is SemEval 2017 Task 10 [10]. The dataset is publicly available on GitHub. The dataset consists of abstracts of 493 research papers available in different text files. A set of keywords were given corresponding to each text file. On the dataset, the following preprocessing techniques were used:

a) The first step consisted of preparing a comma-separated values (CSV) file from the dataset. The file is comprised of two columns - one containing the abstract itself and the other containing keyword corresponding to that abstract.
b) In the second step, we removed the punctation marks present in the abstract. This step increases the efficiency of our algorithms by a significant margin.
c) Since the length of each abstract is different, the next step comprises of finding the maximum length of the abstract that is present in the dataset and padding all the other text values so that the length of each abstract is the same.

d) The final step consists of BIO tagging the text. Hence, each word in a text is classified as B, I, or O where B shows that the word is the beginning of the keyphrase, I denotes that word is inside a keyphrase, and finally O illustrates that word is outside of a keyphrase. The BIO tagging is done according to the keywords present in the dataset.

IV. TECHNIQUES IN KEYWORD EXTRACTION

A. Text Rank

Mihalcea and Tarau in 2004 introduced a graph-based approach for keyword extraction in their paper TextRank: Bringing Order into Texts [1], where words in the documents are represented as nodes in the graph and edges between the nodes are decided using a co-occurrence sliding window that traverses the entire document. Edges are added between nodes/words belonging to a particular sliding window. The algorithm works well because it ensures that it does not rely on the local context of a text unit (vertex) but takes into account information recursively drawn from the entire document (graph). The first step in the algorithm involves cleaning off the nonprintable characters, lemmatization, and removing the stop words. Syntactic filers are applied to extract nouns and adjectives, which are considered potential candidates for forming the graph. There is an edge/connection between the words if the words co-occur within a window of a specified size in the processed text. The scores of all vertices are initialized as 1. After the graph creation step, the Page Rank algorithm is run on the graph for several iterations until it converges (usually for 20-30 iterations, until it converges).

$$S(V_i) = (1-d) + d * \sum_{j=\ln(V_i)} \frac{1}{|Out(V_j)|} * S(V_j) \qquad (1)$$

Here d is a damping factor which is added to ensure that the Page Rank Algorithm doesn't get stuck into dead cycles in recursive computations due to sinking pages or newly added pages. S(V) refers to the PR score of vertex, In(V) refers to the in-degree of node and Out(V) refers to the out-degree of the node. The score is iteratively updated until convergence. Post convergence each vertex/word has a unique score and the words which originally occur as a neighbor in the actual text are merged to form a single keyphrase, where the score is the sum of the scores of each candidate word. The top-ranked words are then taken as keyphrases of the document.

We have used sequence labeling in the form of a classification task using the BIO encoding scheme where each output keyword is assigned a label $L_i \epsilon \{B, I, O\}$. The BIO tags then obtained are used to calculate the precision, recall, and F1 score of the algorithm.

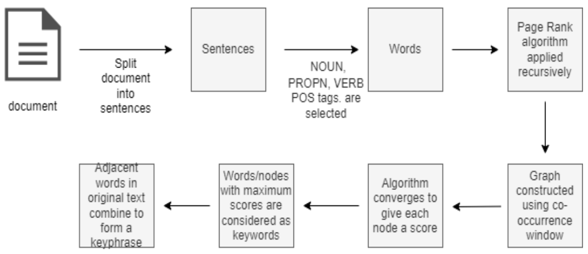
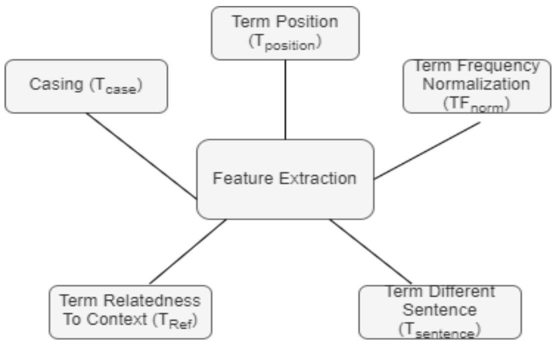

document
Split document into sentences
Sentences
NOUN, PROPN, VERB POS tags. are selected
Words
Page Rank algorithm applied recursively
Graph constructed using co-occurrence window
Algorithm converges to give each node a score
Words/nodes with maximum scores are considered as keywords
Adjacent words in original text combine to form a keyphrase

Figure 1: Text Rank Algorithm Flow

B. Yake

In 2018, Ricardo Campos introduced the YAKE algorithm [2] for keyword extraction in his paper which is a lightweight unsupervised keyword extraction method that rests on statistical text features obtained from single text documents to select not relevant keywords describing the text. This is an unsupervised approach that does not require any annotated text corpora or training data. Relevant keywords are extracted based on statistics extracted from the document and it requires only a stopwords list to be language agnostic. The algorithm defines a set of five features that capture the keyword characteristics which are then heuristically combined to assign a single score to every keyword. The lower the score, the more significant the keyword. The first step involves splitting the sentences using a rule-based sentence segmenter, segtok, and based on white spaces and special characters. The feature set of the algorithm consists of five features, namely Casing (T_{case}: uppercase terms and acronyms are considered more relevant than others), Tern Position ($T_{position}$: terms that occur in the beginning of the text and the terms that occur in the early sentences of the text have a higher value than others), Term Frequency Normalization (TF_{Norm}: terms with higher frequency have greater importance), Term relatedness to context (T_{rel}: higher the diversity of context in which the word occurs, higher the chances of it being a common word) and Term different sentence ($T_{sentence}$: candidates appearing in many different sentences have a higher probability of being important). All the features are combined together to give a single term weight which gives the candidate's importance, given by the following formula.

$$S(t) = \frac{T_{Rel} * T_{Position}}{T_{Case} + \frac{TF_{Norm}}{T_{Rel}} + \frac{T_{Sentence}}{T_{Rel}}} \qquad (2)$$



Term Position ($T_{position}$)
Casing (T_{case})
Term Frequency Normalization (TF_{norm})
Feature Extraction
Term Relatedness To Context (T_{Ref})
Term Different Sentence ($T_{sentence}$)

Figure 2: YAKE Feature Extraction

There are high chances of getting similar morphology words while using the above scheme therefore Levenshtein distance-based deduplication scheme is used wherein a word is not chosen if it has a small Levenshtein distance with the already selected word.

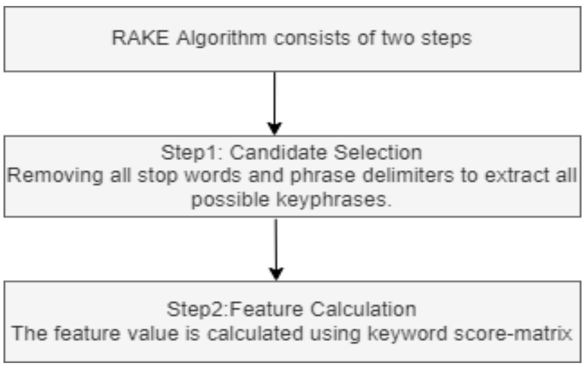
Finally, we perform BIO Tagging of the keywords obtained from the YAKE algorithm, to calculate the precision, recall, and F1 Score of the algorithm.

C. RAKE

Rapid Automatic Keyword Extraction [11] is an algorithm which gives extremely efficient results and operates on individual documents to enable dynamic collection. The RAKE algorithm is based on the observation that keywords rarely obtain stopwords and punctuation words. These words carry minimum lexical meaning. On the other hand, words that explain the context of the document are described as content bearing/content words. The algorithm input consists of a list of stop words also a set of phrase delimiters and word delimiters. All the documents are partitioned using stop words and phrase delimiters. A graph of word co-occurrence is created which calculates the score of each candidate, i.e., ember word score. All the stop words and phrase delimiters are removed to extract all possible keywords. RAKE performs feature calculation using score matrix rather than TF-IDF scores that are commonly used. The estimates used for calculating the score-weight matrix are as follows:

a) Word Frequency: It gives the count of occurrences of word in a document.
b) Word Degree: It represents how often a word co-occurs with different words in candidate keywords.
c) Ratio of degree to frequency: The matrix created is consequently utilized for keyword selection in the document.

The best ranked words are considered as keywords among all the extracted keyphrases. Thus RAKE provides us an individual document-oriented dynamic information retrieval method which is domain independent.



RAKE Algorithm consists of two steps
Step1: Candidate Selection
Removing all stop words and phrase delimiters to extract all possible keyphrases.
Step2:Feature Calculation
The feature value is calculated using keyword score-matrix

Figure 3: RAKE Algorithm

D. Bidirectional LSTM-CRF

Long Short-Term Memory Networks (LSTM) [12] are similar as Recurrent Neural Networks (RNN), except that the hidden layer updates are replaced by purpose-built memory cells. Hence, they are general better at finding and exploiting long range dependencies in the data.

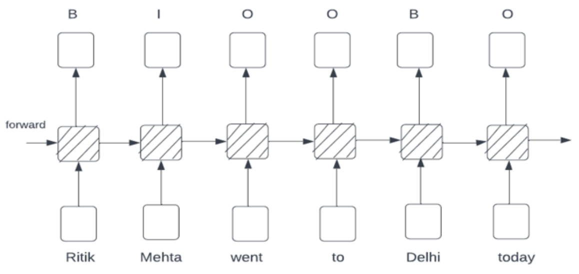


Figure 4: A LSTM Network

Since we have to access both past and future input features at a given time in sequence tagging task, we can therefore use a bidirectional LSTM network. The bidirectional LSTM networks are trained using back propagation through time (BPTT).

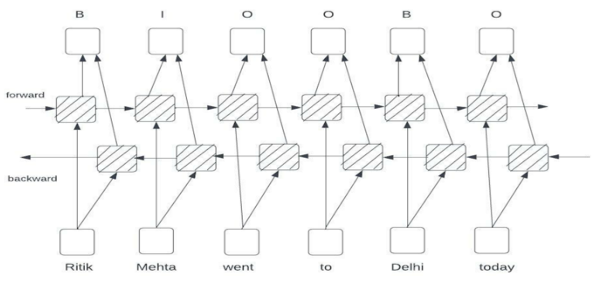


Figure 5: A LSTM Network

Now, the neighbor tag information can be used in two ways to predict current tags. The first comprises of predicting a distribution of tags for each time step and then using beam-like decoding to find optimal tag sequences. The second method consists of focusing on sentence level instead of individual positions, thus producing Conditional Random Fields (CRF) [13] models. In general, CRFs produce higher tagging accuracy.

For our research, we combine a bidirectional LSTM network and a CRF network to form a BI-LSTM-CRF model.

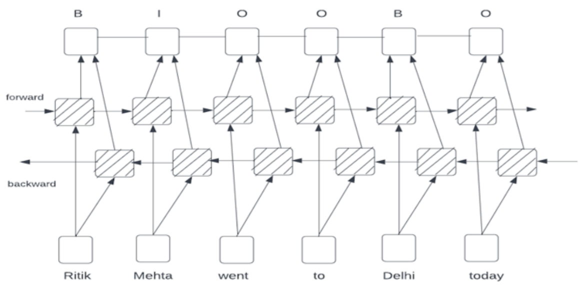


Figure 6: A BI-LSTM-CRF Model

E. Bidirectional LSTM-CNNs-CRF

This architecture was proposed by Xuezhe Ma and Eduard Hovy in May 2016 [14]. They evaluated the character-level representation of each word using the Convolutional Neural Network (CNN). Subsequently, they concatenated the character level representation with word embedding vectors that are then fed into BI-LSTM. At last, the output vectors of BI-LSTM are fed to the CRF layer to jointly decode the best label sequence. The performance of their model significantly increased by applying the dropout layer to both the input and output vectors of BI-LSTM.

F. BERT

Researchers at Google AI Language, Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova introduced BERT (Bidirectional Encoder Representations from Transformers) [15]. It is a state-of-the-art model for NLP providing commendable results for a variety of NLP tasks including Question Answering, Natural Language Interface, Keyword Extraction and many more. BERT applies bidirectional training of Transformer to language modelling. It uses Masked Language Model (MLM) which allows bidirectional training of models, hence getting a deeper sense of language context and flow than the single direction language models.

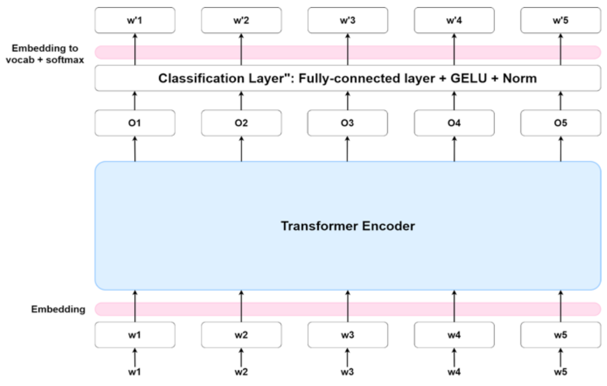
BERT uses Transformer to understand the contextual relation between words. Before the introduction of BERT, models were trained one directionally. They looked at text one directionally either from left-to-right or from right-to-left but BERT is bidirectionally trained which provides it a deeper sense of language and flow compared to single-direction models. BERT is based on Transformer model architecture instead of LSTMs. Transformers perform small steps wherein they try to apply attention model to understand the relationship between the words in the sentence, since it is a language representational model, it only needs the encoder part. The input for BERT encoder is a sequence of tokens, first converted to vectors and then processed in a neural network. Some extra metadata is added to the input before processing:

a) Token Embeddings: [CLS] token is added at the beginning and [SEP] token is added at end of each sentence.
b) Segment Embedding is added to distinguish between sentences.
c) Positional Embedding is added to each token to indicate position in a sentence.

The Figure 7 gives a high-level idea of Transformer encoder. A sequence of tokens is taken as input which are embedded into vectors and then processed in a neural network. The output is the sequence of vectors corresponding to the input tokens.

BERT was pre-trained using unlabeled, plain text corpus, namely the English Wikipedia and the Brown Corpus. It continues to learn unsupervised and improve as being used in practical applications like Google Search. BERT is trained using two strategies:

a) Masked Language Model (MLM): 15% of the words in the input text are replaced with a [MASK] token. The BERT model then tries to predict the masked words using the context of non-masked words in the sequence.
b) Next Sentence Prediction (NSP): In the training process, pairs of sentences are given to the model as input and it learns to predict if the second sentence is the subsequence sentence in the document. In the input dataset, 50% of the inputs are in a pair in which the second sentence is the subsequence sentence in the document, while other 50% are a random sentence in which is disconnected from the first sentence.


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
w4
w5
Figure 7: BERT Transformer Architecture

The MLM and NSP Models are trained together to minimize the combined loss of the two strategies. Transformers use attention mechanism to observe relationships between words.

BERT can be called an Encoder stack of transformer architecture. It is an encoder-decoder network that uses self-attention on the encoder side and attention on the decoder side. $BERT_{BASE}$ has 12 layers in the Encoder stack while $BERT_{LARGE}$ has 24 layers in the Encoder stack. BERT architecture has larger feedforward-networks and more attention heads than the transformer architecture.

BERT is released in two sizes $BERT_{BASE}$ and $BERT_{LARGE}$.

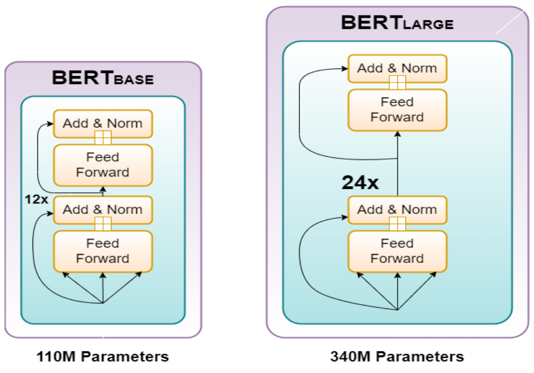


Figure 8: BERT Sizes

V. MANHATTAN DISTANCE BASED ALGORITHM

The development of the Manhattan Distance-Based Approach (MDBA) begins with defining the optimal state of the overall objective, and specifying the ideally good values of metrics involved in the process. The optimal state of the objective can be represented by the optimum model, the OPTIMAL. The vector $OP(r_1, r_2, \ldots\ldots, r_n)$ is the set of "optimum" simultaneous attribute values. The vector OP is called the optimal point in an n-dimensional vector space. For practical purposes, the optimal good value for attributes is defined as the best values which exist within the range of values of attributes. Hence, the OPTIMAL can simply be considered as a keyword extraction model that has all the best values of attributes.

It is very unlikely that a particular keyword extraction algorithm has the best value of all attributes. Instead, a variety of alternatives may be used to simulate the optimal state. Hence, the OPTIMAL model is not to be considered a feasible alternative and is only used as a reference to which other alternatives are quantitatively compared. The quantitative difference resulting from the comparison represents the effectiveness of alternatives to reach the optimal state of the objective function. Therefore, the decision problem is to find a feasible solution which is as equivalent as possible to the optimal point. The objective function can be represented as:

$$Minimize\ \delta\ \{Alt(x), OPTIMAL\} \qquad (3)$$

$$Subjected\ to\ x \in X$$

Here, $Alt(x)$ and δ represent an alternative keyword extraction model in the n-dimensional space, and the distance from the optimal point, respectively.

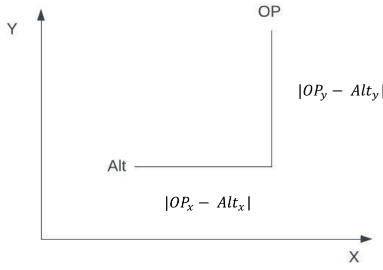


Figure 9: Manhattan Distance Based Approach

The MDBA approach is 2-dimensions is graphically illustrated in Figure 9. The value of δ can be generally formulated as:

$$\delta = \sum |OP_{ij} - Alt_{ij}| \qquad (4)$$

Where i = 1, 2, 3, …, n = alternative keyword extraction models and j = 1, 2, 3, …., m = selection attributes.

To implement the discussed approach, let us assume that we have a set of n keyword extraction models consisting of m selection attributes. Let each alternative be represented by $\mathrm{Alt_1}(\mathrm{r_{11}}, \mathrm{r_{12}}, \mathrm{r_{13}}, \ldots\ldots, \mathrm{r_{1m}})$, $\mathrm{Alt_2}(\mathrm{r_{21}}, \mathrm{r_{22}}, \ldots\ldots., \mathrm{r_{2m}})$, ……… , $\mathrm{Alt_n}(\mathrm{r_{n1}}, \mathrm{r_{n2}}, \ldots\ldots, \mathrm{r_{nm}})$, and the $\mathrm{OPTIMAL}(\mathrm{r_{b1}}, \mathrm{r_{b2}}, \ldots\ldots\ldots, \mathrm{r_{bm}})$, where $\mathrm{r_{bi}}$ = the best value of attribute 'i'. The whole set of alternatives can be represented by the matrix:

$$[r] = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ r_{21} & r_{22} & \cdots & r_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nm} \\ r_{b1} & r_{b2} & \cdots & r_{bm} \end{bmatrix} \qquad (5)$$

To ease the process and standardize the matrix, the following operations are performed:

$$Z_{ij} = \frac{r_{ij} - \bar{r}_j}{S_j} \qquad (6)$$

$$Here, \quad \bar{r}_j = \frac{1}{n}\sum_{i=1}^{n} r_{ij} \qquad (7)$$

$$S_j = \left[\frac{1}{n}\sum_{i=1}^{n} (r_{ij} - \bar{r}_j)^2\right]^{1/2} \qquad (8)$$

Where i = 1, 2, …, n and j = 1, 2, ….., m.

Therefore, the standardized matrix can be represented as:

$$[r_{std}] = \begin{bmatrix} Z_{11} & Z_{12} & \cdots & Z_{1m} \\ Z_{21} & Z_{22} & \cdots & Z_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ Z_{n1} & Z_{n2} & \cdots & Z_{nm} \\ Z_{b1} & Z_{b2} & \cdots & Z_{bm} \end{bmatrix} \quad (9)$$

Now, we have to obtain another matrix, by subtracting each element of the alternative set from the corresponding element of the optimal set, and then taking modulus of the result. The final matrix that comes can be represented as following:

$$[r_{dis}] = \begin{bmatrix} |Z_{b1} - Z_{11}| & |Z_{b2} - Z_{12}| & \cdots & |Z_{bm} - Z_{1m}| \\ |Z_{b1} - Z_{21}| & |Z_{b2} - Z_{22}| & \cdots & |Z_{bm} - Z_{2m}| \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ |Z_{b1} - Z_{n1}| & |Z_{b2} - Z_{n2}| & \cdots & |Z_{bm} - Z_{nm}| \end{bmatrix} \quad (10)$$

Hence, the Manhattan Distance between each alternative keyword extraction technique and the optimal state is given by the following equation:

$$MD_{OPTIMAL-Alt} = \left[\sum_{j=1}^{m} |Z_{bj} - Z_{ij}|\right] \quad (11)$$

Where MD stands for Manhattan Distance. Now, this Manhattan Distance can be used as a quantitative measure to rank the keyword extraction models.

In other words, the Manhattan Distance can be referred as a mathematical expression of several distances on each attribute in which keyword extraction models can be compared.

Table I
Evaluated Value of Metrics for various keyword extraction techniques

MODELS	ACCURACY	PRECISION	RECALL	F1 SCORE
TextRank	74.11	0.507	0.510	0.506
YAKE	76.16	0.502	0.511	0.506
RAKE	65.50	0.492	0.492	0.504
BI-LSTM-CRF	89.72	0.35	0.40	0.37
BI-LSTM-CNNs-CRF	81.11	0.33	0.17	0.22
BERT	81.78	0.64	0.64	0.64
Optimal Value	89.72	0.64	0.64	0.64

VI. EXPERIMENT AND RESULTS

The value of accuracy, precision, recall, and F1 score for the six keyword extraction techniques have been evaluated on the SemEval 2017 dataset. The evaluated value of these parameters has been provided in Table I.

From the comparison of rankings of the six keyword extraction techniques based on the values of these four criteria as given in Table I, it is observed that the ranking of these keyword extraction models varies with respect to the criteria of selection. To avoid this problem, we apply Manhattan Distance Based Algorithm (MDBA) to rank these keyword extraction models. The demonstration is to test the applicability of the method and to develop a procedure for an effective application for the considered six keyword extraction techniques based on their attribute values.

The matrix $[r_a]$ can represent the adjusted matrix of the process. Note that the best numerical value of some criteria is smaller than that of the worst level. To avoid confusion and difficulties in performing the analysis, those values have been adjusted using the following two cases:

Case-I: When a smaller value of the attribute represents fitting well to the actual data, then

Attribute Adjusted Value = Attribute Maximum Value in the database - Attribute Value

Case-II: When a bigger value of the attribute represents fitting well to the actual data, then

Attribute Adjusted Value = Attribute Value - Attribute Minimum Value in the database

Finally, the Manhattan Distance, MD, between each alternative keyword extraction model and the OPTIMAL model is derived from the equation (11). Table II shows the Manhattan Distance and the ranking of alternate keyword extraction models that are determined considering all four metrics using MDBA. The alternate keyword extraction model with the lowest Manhattan Distance value is given rank 1, that with the second-lowest Manhattan Distance is given rank 2 and so on

Table II
Ranking of various Keyword Extraction Techniques

MODELS	MANHATTAN DISTANCE	RANK
TextRank	4.59	3
YAKE	4.39	2
RAKE	5.83	5
BI-LSTM-CRF	5.67	4
BI-LSTM-CNNs-CRF	9.26	6
BERT	0.92	1

VII. CONCLUSION

In this paper we summarized and discussed the various approaches and methods put forward by various researchers in recent years for keyword and keyphrase extraction. TextRank, YAKE, RAKE, BI-LSTM-CRF, BI-LSTM-CNN's-CRF, and BERT models were implemented on the SemEval 2017 Task 10 dataset which consisted of abstracts of 493 research papers. The results of these approaches were analyzed and compared by using accuracy, precision, recall, and F1 score obtained. After using Manhattan Distance Based Approach on the result obtained, we concluded that BERT is the best model with rank 1, and BI-LSTM-CNNs-CRF is the worst model with rank 6.

REFERENCES

[1] R. Mihalcea, Rada, P. Tarau and Paul, "TextRank: Bringing Order into Texts," 07 2004.

[2] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes and A. Jatowt, "YAKE! Keyword Extraction from Single Documents using Multiple Local Features," *Information Sciences,* vol. 509, pp. 257-289, 2020.

[3] M. Zhang, X. Li, S. Yue and L. Yang, "An Empirical Study of TextRank for Keyword Extraction," *IEEE Access,* vol. 8, pp. 178849-178858, 2020.

[4] B. Koloski, S. Pollak, B. Škrlj and M. Martinc, "Extending Neural Keyword Extraction with TF-IDF tagset matching," 2021.

[5] A. Aich, A. Dutta and A. Chakraborty, "A Scaled Conjugate Gradient Backpropagation Algorithm for Keyword Extraction," 2018, pp. 674-684.

[6] S. Biswas, M. Bordoloi and J. Shreya, "A Graph Based Keyword Extraction Model using Collective Node Weight," *Expert Systems with Applications,* vol. 97, 2017.

[7] N. Nikzad Khasmakhi, F. Derakhsh, M. Reza, M. Asgari-Chenaghlu, M. Balafar, A. R. Feizi Derakhshi, T. Akan--R.Farshi, M. Ramezani, Z. Jahanbakhsh, E. Zafarani-Moattar and Khadivi, "Phraseformer: Multimodal Key-phrase Extraction using Transformer and Graph Embedding," 06 2021.

[8] S. Craw, "Manhattan Distance," pp. 1-1, 01 2016.

[9] K. Sharma, C. Nagpal and R. Garg, "Selection of Optimal Software Reliability Growth Models Using a Distance Based Approach," *IEEE Transactions on Reliability,* vol. 59, pp. 266 - 276, 07 2010.

[10] I. Augenstein, M. Das, S. Riedel, L. Vikraman and A. McCallum, "SemEval 2017 Task 10: ScienceIE - Extracting Keyphrases and Relations from Scientific Publications," pp. 546-555, 01 2017.

[11] S. Rose, D. Engel, N. Cramer and W. Cowley, "Automatic Keyword Extraction from Individual Documents," in *Text Mining: Applications and Theory*, 2010, pp. 1 - 20.

[12] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural computation,* vol. 9, pp. 1735-80, 1997.

[13] H. Wallach, "Conditional Random Fields: An Introduction," *Technical Reports (CIS),* 02 2004.

[14] X. Ma and E. Hovy, "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF," pp. 1064-1074, 03 2016.

[15] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 10 2018.

★★★