KEYWORD EXTRACTION USING BERT AND GRAPH EMBEDDING ALGORITHMS

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY
IN
MATHEMATICS AND COMPUTING

Submitted by:
Ritik Mehta (2K18/MC/093)
Ramneek Kaur (2K18/MC/089)
Meenal Sarwaiya (2K18/MC/066)

Under the supervision of
DR. Goonjan Jain

DEPARTMENT OF APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering) Bawana Road, Delhi-110042

December 2021

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana road, Delhi-110042

CANDIDATE'S DECLARATION

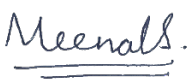
We, (Ritik Mehta 2K18/MC/093, Ramneek Kaur 2K18/MC/089, Meenal Sarwaiya 2K18/MC/066), students of B.Tech in Mathematics and Computing, hereby declare that the project Dissertation titled "**Keyword Extraction using BERT and Graph Embedding Algorithms**" which is submitted by us to the Department of Applied Mathematics, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship, or other similar title or recognition.

Place: Delhi

Ritik Mehta 2K18/MC/093

Date: 01/12/2021

Ramneek Kaur 2K18/MC/089



Meenal Sarwaiya 2K18/MC/066

DEPARTMENT OF APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Bawana road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled '**Keyword Extraction using BERT and Graph Embedding Algorithms**' which is submitted by Ritik Mehta (2K18/MC/093), Ramneek Kaur (2K18/MC/089) & Meenal Sarwaiya (2K18/MC/066) to Department of Applied Mathematics, Delhi Technological University in partial fulfillment of the requirements for the award of degree of Bachelor of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi

Date: 29/11/2021

Dr. Goonjan Jain

SUPERVISOR

Department of Applied Mathematics

Delhi Technological University

ABSTRACT

In recent times, there have been a lot of advancements in the field of Natural Language Processing. There are numerous applications of Natural Language Processing such as in Voice Assistants, Text Completion, Chatbots, Grammar Checkers, and so on. One such important application is Keyword Extraction. Keywords are the words that describe the most important and relevant information in a document. In today's day and age where we have millions of documents, it is not practically possible to manually extract keywords from each document. Hence, it becomes very essential to automate the process with Natural Language Processing techniques. The major problem that researchers are encountering is to develop a model that efficiently and correctly extract the core words from a document. Previous methods have used text and graph features to accomplish this task. However, there is a lack of model that can learn combine text and graph features in a best way.

In our project, we aim to develop a multi-modal approach to solve the problem of Keyword Extraction. We combined the BERT embeddings and the vectors obtained by Graph Embedding techniques in the first phase of our project. The next phase of our project would consist of training various Sequence Labelling models on the vectors obtained for every word, and rank the models based on a Manhattan Distance Based Approach. At the conclusion of the project, we would be using Gradient Boosting techniques on all the models obtained which would boost the accuracy by a significant margin.

We would be analyzing the performance of our models on three datasets including Inspec, SemEval 2010, and SemEval 2017.

ACKNOWLEDGEMENT

This project could not have been successful without the support and assistance of many individuals and organizations, and we are blessed to have their support all along the course of our project. We would like to extend our gratitude to every one of them.

We are also grateful to Delhi Technological University for constant guidance and supervision, as well as for providing us with a friendly environment and the required infrastructure and resources for the successful completion of the project.

We would like to express our gratitude to our project supervisor Prof. Gunjan Jain who took keen interest in our project and guided us throughout the project by providing all the necessary ideas, information and knowledge for the development of the project. We are thankful and fortunate enough to get constant support from our seniors and every teaching staff of the MCE department which helped us successfully in completion of our project. We would also like to remember all the authors who helped us with necessary information on the relevant topics for the completion of the project. Last but not the least, we would also like to express our gratitude to every one of our colleagues for their encouragement and support in developing the project.

Yours Sincerely,
Ritik Mehta 2K18/MC/093
Ramneek Kaur 2K18/MC/089
Meenal Sarwaiya 2K18/MC/066

CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

A keyword is a word that best describes the meaning and summarises the context of document. It best describes the essence of a document. ‘Key phrases’, ‘key segments’, ‘key terms’ and ‘keywords’ all are the terminology used for defining the more relevant information contained in a document.

Keyword Extraction also known as keyword analysis is a technique which help to automatically identify and extract words that best describe the subject of a document. It helps summarize the principal topics and summarize the content of a text.

If we consider a real-life scenario where we have a large pool of documents. It will be very time consuming and unfeasible to go through all the documents manually and find out the relevant documents of our areas of interest and the ones that meet our requirements. Keyword Extraction technique will help us sift through the whole set of data in minutes and obtain the words that best describe a particular subject [1]. Keywords and key phrases virtually give us the summary of the text and attempts are made to search and select for a document belonging to the relevant topic by finding out their keywords and key phrases. Document summarization is another use case where keyword extraction is of great help. It collects the relevant information from a data set in a condensed way and provides the user with most important information.

It is very strenuous to find the correct keywords and key phrases that best summarize the text manually, as a result a lot of algorithms are developed to automate the process of listing keywords in a document. Automated keywords can be used to index and categorize, group or classify documents into different domains, allowing easy search

and categorization of documents.

The automatic extraction of keywords and key phrases using machine learning has taken either supervised or unsupervised approaches. The difference between the two approaches lies in the availability of labelled training set in the learning process. The supervised method transforms the keyword extraction task to a regression or classification problem. The model is trained on a labelled training set and the trained model is then used to determine if the candidate word is a keyword or not. Supervised methods are superior to unsupervised methods but they require many labelled training sets, thus it require large amounts of manual labor. Unsupervised approach does not require large training data instead the keywords and key phrases are determined using various properties of text in the document. Unsupervised approaches can be divided into statistics-based method, graph-based method, topic-based method, language model-based method, and these methods can be classified into two schools: the linguistic school and the statistical school. The linguistic school analyses text using linguistic methods among which the most common is to analyse the topic distribution of articles, for example Key Cluster and Community Cluster. The statistical school on the other hand mainly analyses the article's probability features such as KP-Miner and YAKE based on TF-IDF, Text Rank or Single Rank [2].

In our project we aim to devise a supervised keyword extraction technique that efficiently help us extract the relevant keywords describing a text.

CHAPTER 2
LITERATURE REVIEW

Despite the utility of keywords for analysis, indexing and retrieval, most documents do not have assigned keywords. In a number of approaches keywords are assigned manually by professional curators who may use a fixed taxonomy or rely on authors judgement to provide representative list. Therefore, there is focus on methods to extract these keywords automatically from documents as an aid either to suggest keywords to professional indexer or to generate summary of documents.

In 2000, Turney(2000) and Peter D(200) approached the problem of keyword extraction as a supervised learning task. They were the first to propose keyword extraction as a supervised machine learning task. . The entire document was considered as a set of key phrases and then the algorithm was used to classify these key phrases into positive and negative. They used C4.5 decision tree induction algorithm and GenEx algorithm for this classification task.

Researchers earlier than this treated keyword extraction as classification on the basis of heuristics. Krulwich and Burkey (1996) [3] used heuristics to extract key phrases. The heuristics included syntactic clues such as use of italics, the presence of phrases in section headers, use of acronyms etc. Their approach uses heuristics to extract significant phrases from documents for learning rather than using standard mathematical techniques. This technique using heuristics used to generate a large number of key phrases with very low precision.

Salton (1989) [4] proposed a method based on word concurrence inside sentences. This method choses a main term (imposing a frequency threshold) and then associations are made between this term and other terms in the same sentence. In this method a lot of

weight is given to one selected term which is arbitrarily chosen produces a high percentage of meaningless associations. The results showed that there was a strong need to use relevance judgements to generate keywords. Munoz (1996) introduced compound keyword generation from document databases using an unsupervised learning technique which used a Hierarchical Clustering ART model. His method used frequency of words in a document for automatic keyword generation. He proposed a method which uses a hierarchical model made up of Fuzzy Adaptive Resonance Theory (ART) neural networks. The system uses Fuzzy ART modules to cluster isolated words into semantic classes. This knowledge is used with co-occurrence information to extract meaningful term associations. This method produced a large list of key phrases but with low precision. One other drawback of this method was that it could not produce one word or more than two-word key-phrases.

Jones (1972) [5] algo gave importance to frequency of words occurring in a document in his paper which explained statistical interpretation of term specificity and its application in retrieval. According to him, frequently occurring terms are required for good overall performance and the terms should be weighed according to collection frequency so that matches on more specific and less frequent words are of greater value than matches on frequent terms. A method for exporting phrases introduced by Steier (1993) and Belew (1993) produced similar results as Munoz (1996). It produced similar two-word key phrases with low precision. Their method took into account that some phrases have very different collocation patterns than their constituent words, we must look beyond occurrences of individual words to consider word phrases. Their method used mutual information statistic to measure the information content of phrases beyond that of their constituent words.

Andrade (1998) and Valencia (1998) [6] presented a system that automatically annotates proteins functions by extracting relevant keywords from scientific literature associated with a given protein. They base their approach on comparison of word frequency

distributions within a text against distributions from reference corpus

Turney (2000) proposed the automatic keyword extraction as a supervised machine learning task [7]. Hulth(2003) also performed automatic keyword extraction from abstracts using a supervised machine learning algorithm . The focussed on the point that by adding linguistic knowledge to the representation (such as syntactic features) , rather than relying only on statistics (such as term frequency and n-grams) , a better result is obtained as measured by keywords previously assigned by professional indexers. The method proposed that extracting NP-chunks gives better precision than n-grams and by adding POS tags assigned to the term as a feature a substantial improvement in result is obtained irrespective of the term selection approach. This method gives a small set of terms that describe a specific document independently of the domain it belongs to. One drawback of this approach was that there was no relation between the different tag feature values.

Zhang (2006) , Tang (2006) and Li (2006) [8] proposed a method for keyword extraction using Support Vector Machine. They proposed to use ‘local context information’ along with ‘global context information’ for extracting keywords from the document. The Support Vector Machines (SVM) model they used significantly outperformed the baseline methods for keyword extraction. The proposed method was also applied to document classification, a typical text mining process which led to significant increase in accuracy. Ercan (2007) and Cicekli (2007) [9] also gave a supervised learning method based on using lexical chains for keyword extraction. This method investigates the benefits of using lexical chain features in keyword extraction. The results obtained show that the lexical chain features improve the precision significantly in the keyword extraction process. The words in a lexical chain are syntactically related and may cover small or big portion of text. A keyword which represents the semantic content of a text should be selected from the lexical chains which represent the most semantic content of text. Keyword Extraction and Text Summarization are very related tasks. Barzilay (1997) and Elhadad (1997) have shown that features based on lexical features are good for text summarization. Silber (2000) and McCoy (2000) have described a efficient method for

creating lexical chains

In June 2021 , N. Nikzad–Khasmakhi poposed Phraseformer [10] which is a multimodal key-phrase extraction approach which uses transformer and graph embedding techniques. We aim to extend this model in our project.

CHAPTER 3
BERT EMBEDDINGS

Bidirectional Encoder Representations from Transformers (BERT) is a neural network-based technique for pre-training natural language processing. It can be used to assist Google better understand the context of words in search requests, in plain English.

The capacity of BERT to train language models based on the full set of words in a sentence or query (bidirectional training) rather than the usual method of training on the ordered sequence of words is its major advancement (left-to-right or combined left-to-right and right-to-left). BERT effectively allows the language model to understand word context from surrounding words rather than just the one that comes before or after it.

The word embeddings are low-dimensional dense vector representations of words. It is feasible to express the semantic significance of a word in a numeric form and so execute mathematical operations on it by converting it to an embedding. The BERT base model has 12 layers of transformer encoders, with each token's output serving as a word embedding. BERT is a type of general language modelling that aids in transfer learning and fine-tuning on specific tasks.

The special tokens that are used in BERT for fine-tuning and specific task training. These are the following:

[CLS]: Every sequence's first token. For classification tasks, a classification token is typically used in conjunction with a softmax layer. It can be safely ignored for anything else.

[SEP]: A token that served as a sequence delimiter during pre-training for sequence-pair tasks (i.e. Next sentence prediction). When a sequence pair task is required, it must be used. When using a single sequence, it is simply appended at the end.

[MASK]: A token for words that are masked. Only used as pre-training.

The input format that BERT expects is illustrated below:

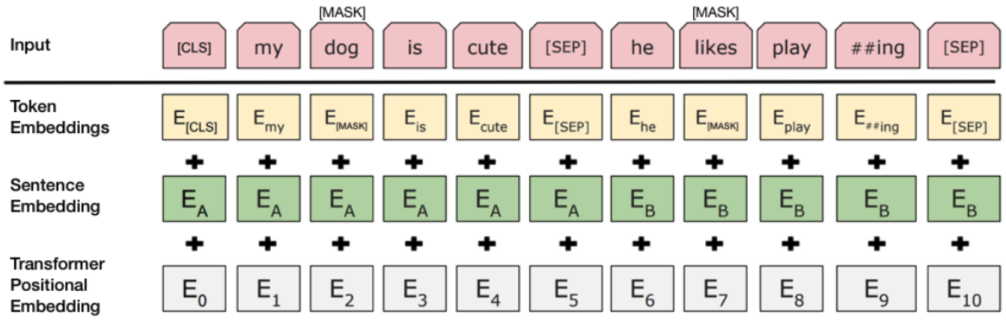


Fig. 3.1 BERT Input Representation

The input that to be used with BERT must be formatted to match the above.

The input layer consists of a vector that contains the sequence tokens as well as the special tokens. To explain the "##ing" token in the example above, BERT uses WordPiece for tokenization, which effectively separates tokens like "playing" into "play" and "##ing." This is primarily to encompass a broader range of words that are Out-Of-Vocabulary (OOV).

The vocabulary IDs for each of the tokens are called token embeddings.

Sentence Embeddings is simply a number class that differentiates between sentence A and sentence B.

Finally, Transformer positional embeddings show where each word in the sequence is located.

The BERT base model employs 12 layers of transformer encoders, with each token's output serving as a word embedding. Depending on the objective, summation of the last four layers was found to be one of the best performing options.

Sentence Transformers is a Python framework that allows you to create cutting-edge sentence, text, and image embeddings. The following is an example of a sentence-transformers model: It converts sentences and paragraphs into a dense vector space with 768 dimensions, which can be utilized for applications like clustering and semantic search. This framework can be used to generate sentence / text embeddings in over 100 languages. These embeddings can then be compared, for example, using cosine-similarity to locate sentences that have similar meanings. This is useful for things like semantic textual similarity, semantic search, and paraphrase mining.

The framework is built on PyTorch and Transformers, and it includes a huge number of pre-trained models that may be used for a variety of tasks. Furthermore, fine-tuning models is simple.

CHAPTER 4
GRAPH EMBEDDINGS

Graph embedding is a technique for transforming nodes, edges, and their attributes into vector space (a lower dimension) while maintaining as much graph structure and information as possible. Graphs are difficult to understand since they vary in scale, specificity, and subject.

The basic concept behind graph embeddings is graph-based approaches is to create a co-occurrence graph from documents. The co-occurrence network depicts how words in a corpus interact. Words are nodes in this graph, and there is an edge connecting two words if they co-occur inside a timeframe. To locate key nodes, certain centrality measures such as degree, proximity, betweenness, and eigenvector are applied to the co-occurrence graph once it has been constructed. The keywords are identified by the most central nodes in these approaches.

There are several approaches to graph embedding, each with a distinct amount of granularity. Embeddings can be done at the node level, at the subgraph level, or via graph walk methods. These are a few of the most widely used techniques.

4.1 DeepWalk

Deepwalk is among the first graph learning algorithms to be widely used as a benchmark in comparison to other methodologies. Deepwalk is a graph embedding approach that makes use of walks, which are a graph theory concept that allows you to traverse a graph by going from one node to another as long as they're connected by a common edge.

You can traverse a graph by representing each node with an arbitrary representation vector. By putting the node representation vectors adjacent to one other in a matrix, the

stages of that traversal might be aggregated. The graph's matrix might then be fed into a recurrent neural network. It can basically use the reduced stages of graph traversals as RNN input. This is similar to how word vectors in a phrase are assembled.

DeepWalk takes the strategy of completing a series of random walks using the equation:

$$\Pr(v_i \mid (\emptyset(v_1), \emptyset(v_2), \ldots\ldots \emptyset(v_{i-1})))$$

The purpose is to calculate the probability of seeing node v_i based on all of the prior nodes visited so far in the random walk, where $Pr()$ is probability and $\emptyset$ is a mapping function that represents each node v in the graph's latent representation.

The latent representations are what a neural network uses as input. Based on which nodes were encountered and how often they were encountered throughout the walk, the neural network can provide a prediction regarding a node attribute or categorization.

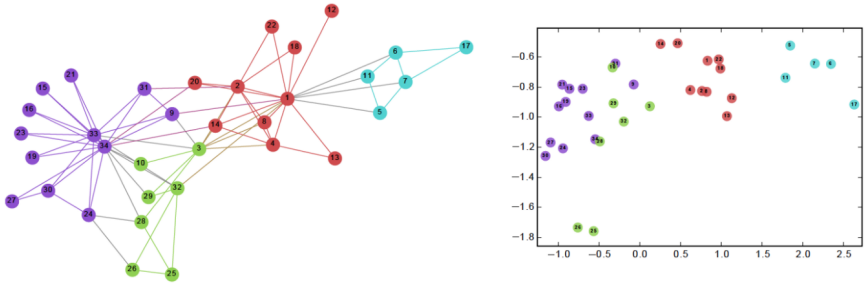


Fig. 4.1 Graph and its Embedding

The skip-gram approach is used to produce predictions, just like in the Word2vec architecture for text. DeepWalk learns an embedding by running along the graph rather than the text corpus. The model can forecast the "context" of a target node, which in the case of a graph means its connectivity, structural role, and node features.

Although DeepWalk is relatively efficient, with a score of O(|V|), it is transductive, which means the model must be retrained every time a new node is introduced in order to embed and learn from it.

4.2Node2Vec

Node2vec is one of the first Deep Learning attempts to learn from graph structured data, and it is one of the most prominent graph learning algorithms. The idea is similar to DeepWalk's: a neural network can learn representations for each node in a graph by turning each node into an embedding, similar to how words in a sentence are turned into embeddings.

There is a slight but substantial difference between Node2vec and DeepWalk. The walk bias variable p and q in Node2vec is parameterized by p and q. A breadth-first-search (BFS) procedure is prioritised by parameter p, whereas a depth-first-search (DFS) technique is prioritised by parameter q. Probabilities 1/p or 1/q therefore influence the decision of where to walk next.

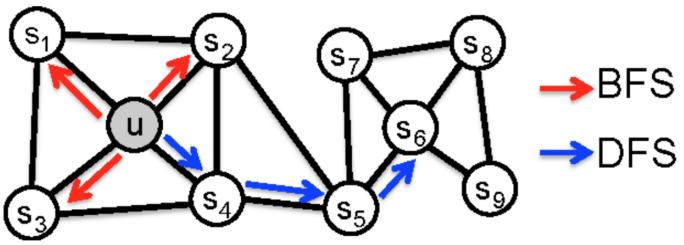


Fig. 4.2 BFS and DFS in Graph Theory

BFS is best for learning local neighbors, whereas DFS is better for learning global variables, as the diagram shows. Depending on the task, Node2vec can switch between the two priorities. This means that depending on the values of the parameters, Node2vec can produce different results for the same graph. Node2vec uses the latent embedding of

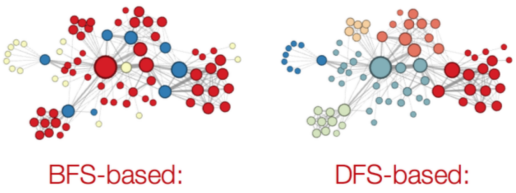


Fig 4.3 BFS vs DFS

the walks as input to a neural network to classify nodes, similar to DeepWalk.

Experiments show that BFS is better at categorizing based on structural roles (hubs, bridges, outliers, and so on), but DFS returns a more community-driven categorization scheme.

4.3 Graph2Vec

Graph2vec is a variation of node2vec that essentially learns to embed a graph's sub-graphs. An equation used in doc2vec, a similarly comparable variation, demonstrates this.

$$Pr(w_j|d) = \frac{exp(\underset{d}{\rightarrow} . \underset{w_j}{\rightarrow})}{\sum_{w \epsilon v} exp(\underset{d}{\rightarrow} . \underset{w}{\rightarrow})}$$

The probability of the word (wj) appearing in context given document (d) equals the exponential of the document embedding matrix (d) multiplied by the word embedding matrix (wj is sampled from the document), divided by the sum of all the exponentials of the document embedding matrix multiplied by the word embedding matrix for each word in the vocab list (V) across all documents

If a document is made up of sentences (which are then made up of words), then a graph is made up of sub-graphs, to use an analogy with word2vec (which is then made of nodes).

The user specifies the number of edges in these preset sub-graphs. The latent subgraph embeddings are sent into a neural network for classification once again.

4.4 Structural Deep Network embedding (SDNE)

SDNE does not use random walks, unlike earlier embedding approaches. Instead, it makes an attempt to learn from two different metrics:

First-order proximity: Two nodes are deemed comparable if they share an edge in the

first order (pairwise similarity)

Second-order proximity: two nodes are said to be comparable if they have a large number of neighboring/adjacent nodes in common.

The ultimate goal is to capture structures that are highly non-linear. This is accomplished by preserving the first order (supervised) and second order (unsupervised) network proximities with deep autoencoders (semi-supervised).

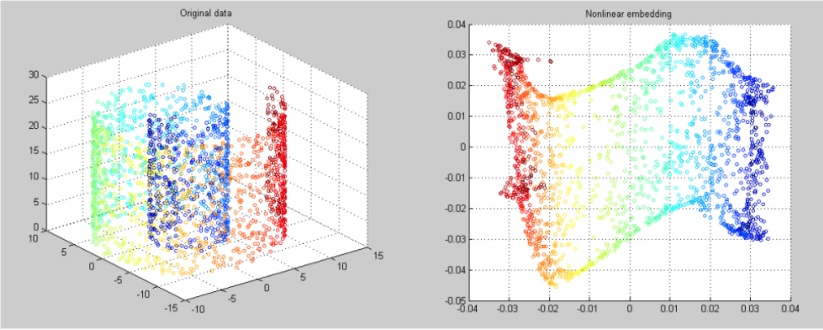


Fig. 4.4 Dimensionality Reduction

The model also uses a variant of Laplacian Eigenmaps, a graph embedding/dimensionality reduction technique, to preserve first order proximity. When comparable nodes are mapped far apart in the embedded space, the Laplacian Eigenmap embedding approach applies a penalty, allowing for optimization by decreasing the space between similar nodes.

The graph's adjacency matrix is sent through an unsupervised autoencoder with a built-in reconstruction loss function it must minimize to retain the second order proximity.

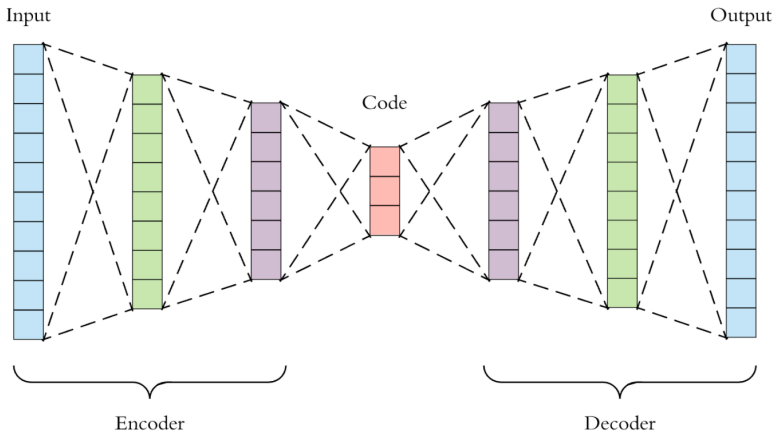


Fig. 4.5 An Auto-Encoder

To return a graph embedding, the first order proximity loss function and the second order reconstruction loss function are minimized together. A neural network then learns from the embedding.

4.5 Large-scale Information Network Embedding (LINE)

LINE specifies two functions explicitly, one for first order proximity and the other for second order proximity. Second order proximity performed much better than first order closeness in the studies conducted by the original research, and it was suggested that introducing higher levels may level off the advantages in accuracy.

LINE's purpose is to reduce the disparity between the input and embedding distributions as much as possible. KL divergence is used to do this.

For each pair of nodes, LINE creates two joint probability distributions and then minimizes the KL divergence between them. The adjacency matrix and the dot product of node embedding are the two distributions. In information theory and entropy, KL Divergence is an important similarity metric. The approach is utilized in probabilistic generative models such as Variational Autoencoders, which integrate autoencoder inputs into a latent space that generates the distribution.

If the application requires an understanding of node community structure, LINE does not perform well since the algorithm must define new functions for each rising order of proximity.

4.6 Hierarchical Representation Learning for Network (HARP)

HARP is a step forward from the embedding/walking-based models stated above. Because their objective functions are non-convex, previous models ran the risk of being stranded in local optima. This essentially indicates that the ball cannot roll all the way to the bottom of the hill.

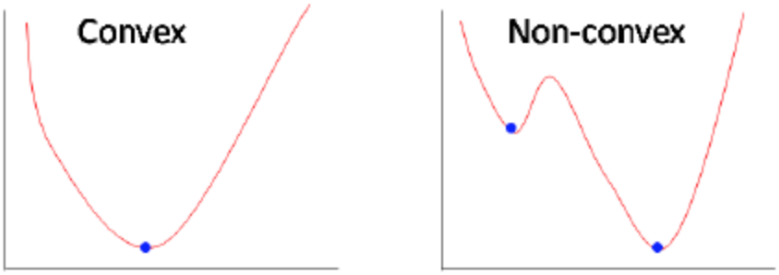


Fig. 4.6 Non-Ideal Gradient Descent

As a result, the desired outcome is to improve the solution and prevent local optima through better weight initialization as well as the suggested method: It was decided to use graph coarsening to group related nodes into "supernodes."

HARP is a graph preprocessing procedure that simplifies the graph to make training go faster.

It then constructs an embedding of the coarsest "supernode" after coarsening the graph, followed by an embedding of the entire graph (which itself is made of supernodes).

This method is used for each and every "supernode" in the graph.

Because HARP may be combined with other embedding methods such as LINE, Node2vec, and DeepWalk.

CHAPTER 5

FINAL WORD REPRESENTATIONS

The next stage is to merge the information from each word's vector representation of text information and co-occurrence network structure into a single representation. We believe that combining text-based and structure-based data can help us better understand the potential of words as keywords. As a result, we show each word with a single vector that combines the text and structural vectors. We show $V_{w_i} = T_{w_i} + N_{w_i}$ for word w_i, where T_{w_i} and N_{w_i} are the text and structure learning representations for word w_i, respectively.

CHAPTER 6
RESULTS

6.1 Word Cloud of the Dataset After Preprocessing

Fig. 6.1 Word Cloud of the Dataset

6.2 BERT Embeddings of Various Words

986	in	[-0.39861273765563965, -0.08793222904205322, 0...
987	line	[-0.32531723380088806, -0.3866935670375824, 0....
988	with	[-0.32526642084121704, 0.06888231635093689, 0....
989	similar	[-0.12189409881830215, 0.017909549176692963, 0...
990	problems	[-0.6587560772895813, -0.02208827994763851, 0....
991	in	[-0.39861273765563965, -0.08793222904205322, 0...
992	quantum	[-1.145789623260498, -0.5190446376800537, 0.29...

Fig. 6.2 BERT Embeddings

6.3 Co-occurrence matrix of words in the dataset

```
print(Xc.todense()) # print out matrix in dense format
```

```
[[0 0 0 ... 0 0 0]
 [0 0 4 ... 0 0 0]
 [0 4 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 0 0]]
```

+ Code + Markdown

Fig. 6.3 Co-occurrence matrix

6.4 Co-occurrence Network of Keywords in the dataset using NLOOP

Fig. 6.4 Co-occurrence Network of Keywords

6.5 Structural Vector of Words

Below is a structural vector of a word obtained using DeepWalk algorithm

```
embedder['rules']
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:1: DeprecationWarni
d will be removed in 4.0.0, use self.wv.__getitem__() instead).
  """Entry point for launching an IPython kernel.
array([ 4.4945125e-03,  1.5930567e-03, -3.2959159e-03, -4.7079829e-04,
        4.4389867e-04, -1.2436265e-03, -4.8047868e-03,  2.5819733e-03,
       -5.7200210e-05,  8.7829778e-04,  4.3049254e-03,  2.9845403e-03,
        3.4332103e-03,  5.9817522e-04,  1.3161903e-03, -8.8676787e-04,
        4.8860833e-03,  1.5178326e-03, -1.5985992e-04, -3.0056280e-03,
        1.3530839e-04,  1.2370693e-03,  4.6323761e-03, -9.5971179e-04,
        3.0666841e-03,  3.4303733e-03,  1.9075126e-04, -1.5128262e-03,
        1.4735778e-03, -3.9128684e-03, -3.3708457e-03,  2.0843293e-03,
        3.2222725e-03, -3.1112165e-03,  3.7655043e-03, -3.8775424e-03,
        1.9492760e-03, -3.4080050e-03,  5.6072732e-04,  2.5117244e-03,
        2.6082208e-03,  8.3765067e-04,  3.6582787e-04,  4.2678760e-03,
        2.1184401e-03, -8.7880943e-04,  3.8509169e-03,  4.6400065e-03,
       -2.3543526e-04, -2.9169531e-03, -2.8613161e-03,  3.1793453e-03,
        3.1446915e-03,  3.4396006e-03, -3.0795515e-03, -2.9761060e-03,
       -2.2148313e-03,  2.2076329e-03, -1.7944116e-03,  4.8192553e-03,
       -4.6103293e-04,  4.2750528e-03,  2.3331568e-05,  1.3910718e-03,
       -1.8496389e-03, -1.3622206e-03,  1.7156554e-03,  4.0130415e-03,
       -1.5254427e-03,  7.9786248e-04, -2.9873122e-03,  2.5911769e-03,
        1.5959020e-03,  1.8431140e-03, -3.5757564e-03,  8.5595349e-04,
       -2.9545464e-03, -1.2979975e-03, -4.3072742e-03, -3.2468848e-03,
        9.2306815e-04,  1.6582361e-03,  3.4944760e-03,  4.5917537e-03,
       -4.8012785e-03,  2.3411941e-03,  3.0506417e-04,  3.9770892e-03,
        1.4063698e-04, -1.6387179e-03,  6.7779532e-05, -2.4908417e-04,
        3.5820801e-03,  6.9161510e-04, -1.9912024e-03,  2.9275247e-03,
        3.8720781e-04,  3.9045049e-03, -4.9586892e-03, -4.3672929e-04],
      dtype=float32)
```

Fig. 6.5 Structural Vector of a Word

CHAPTER 7
FUTURE WORKS

7.1 Sequence Labelling and Classification

Sequence Labelling can be regarded as type on Pattern Recognition Task in the field of Natural Language Processing that involves the algorithmic assignment of a categorical label to every word in a given input sequence. There are broadly two forms of sequence labelling:

a) Token Labelling: Each token gets an individual Part of Speech (POS) label
b) Span Labeling: Labeling segments or groups of words that contain one tag (Named Entity Recognition, Syntactic Chunks).

There are various algorithms available for Sequence Labelling. Some of which we will account for would include Hidden Markov Model (HMM), Conditional Random Field (CRF), Average Perceptron (AP), Structured SVMs, Max Margin Markov Networks (M3N), and an integration of search and learning algorithm (SEARN).

In our project, we would be considering the problem of key-phase extraction from a document as a Sequence Labelling Task. We would consider Sequence Labelling in the form of a classification task using the BIO encoding scheme as output labels. Hence, our models will take $V_{w_1}, V_{w_2}, \dots\dots\dots, V_{w_n}$ as input, where V_{W_i} represents the vector representation of word w_i, and give each word a label $L_i \in \{B, I, O\}$ where B shows that w_i is the beginning of the key-phrase, I denotes that w_i is inside the key-phrase and O indication that w_i is outside the key-phrase.

7.2 Rankings based on Manhattan Distance Based Approach

Now, that we have trained various models in the previous step, the next step that would

come is ranking those models. The Manhattan Distance Based Approach (MDBA) [11] starts with specifying an optimal state of the overall objective, and consists the ideally good values of attributes involved in the process. The hypothetical model which represents the optimal state is referred to as OPTIMAL. The vector OP $(r_1, r_2, \ldots\ldots, r_n)$ is the collection of "optimum" simultaneous attribute values. It is referred to as the optimal point. The ideally good or optimal good value of an attribute is defined as the best value which exists within the range of values of attributes. In other words, the OPTIMAL is the model which has all the best values of all the attributes.

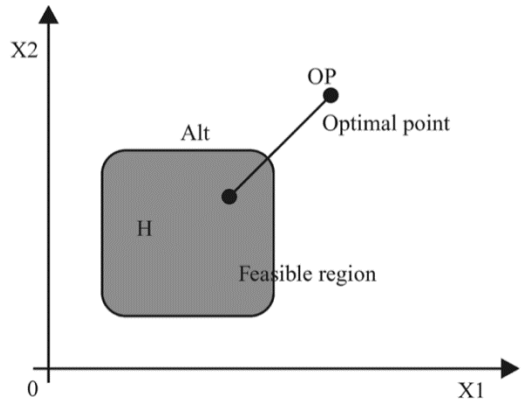


Fig. 7.1 Manhattan Distance Based Approach

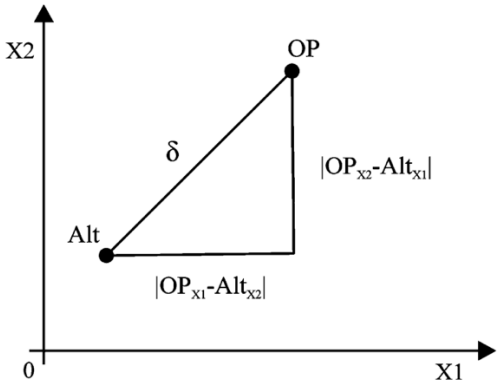


Fig. 7.2 Calculation of Manhattan Distance

It is very unrealistic that a particular model has all the best values of all the attributes.

Thus, the OPTIMAL is not considered as a feasible alternative and is only used as a measure to which the other models are compared. The capability of an alternative model to reach the optimal state is measured by the numerical distance between the OPTIMAL model and the alternative model. The lesser the difference, the better is the alternative model. Our objective can be summarized as to find a model that is closed to the OPTIMAL model. The objective function can be written as:

$$Minimize\ \delta(Alt(h), OPTIMAL)$$
$$Subject\ to\ h\ \in H$$

Where $Alt(h)$ and δ represents an alternative model in n-dimension, and distance from the optimal point respectively. Our objective function can be summarized in Fig. X where H is the feasible region and OP is the optimal point.

The MDBA will be used to determine which model is closed to the OPTIMAL model and is graphically shown in Fig. Y for a 2 – dimensional space. Therefore, in a 2 – dimensional space, the distance between the OPTIMAL model and the Alternative model can be formulated as:

$$\delta = |OP_{X1} - Alt_{X1}| + |OP_{X2} - Alt_{X2}|$$

In a n-dimensional system, the above metric will be used to rank the models. The lesser the value of δ, the better is the ranking of the alternative model.

7.3 Ensemble Models

Ensemble models is a machine learning approach that combine multiple models in the prediction process. The models that are combined are referred to as base eliminators. Ensemble models overcome the disadvantages of a building a single model which include:

a) High Variance: The model is very sensitive to the provided inputs to the learned features.
b) Low accuracy: One model or one algorithm to fit the entire training data might not be good enough to meet expectations.
c) Features noise and bias: The model relies heavily on one or a few features while making a prediction.

Now, since we have trained and ranked all our models, the last step would be to combine all those models to form an ensemble model as it would probably increase the efficiency by a significant margin.

7.3.1 Ensemble Algorithms

A single algorithm may not make the perfect prediction for a given dataset. Machine learning algorithms have their limitations and producing a model with high accuracy is challenging. If we build and combine multiple models, the overall accuracy could get boosted. The combination can be implemented by aggregating the output from each model with two objectives: reducing the model error and maintaining its generalization. The way to implement such aggregation can be achieved using some techniques.

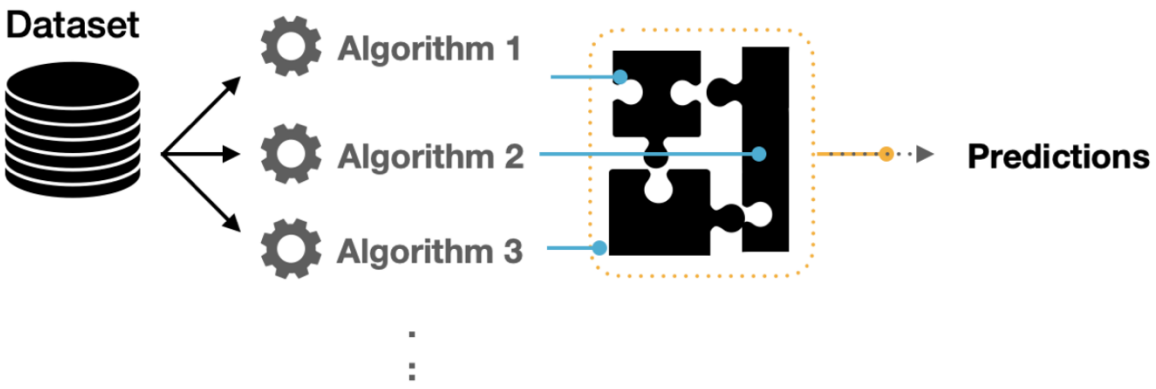


Fig. 7.3 Ensemble Algorithms

7.3.2 Ensemble Techniques

In our project, we will be using AdaBoost, CatBoost, and XGBoost for creating ensemble models:

a) AdaBoost - It is an ensemble of algorithms, where we build models on the top of several weak learners. Those learners are called weak because they are typically simple with limited prediction capabilities. The adaptation capability of AdaBoost made this technique one of the earliest successful binary classifiers. Sequential decision trees were the core of such adaptability where each tree is adjusting its weights based on prior knowledge of accuracies. Hence, we perform the training in such a technique in sequential rather than parallel process. In this technique, the process of training and measuring the error in estimates can be repeated for a given number of iteration or when the error rate is not changing significantly.

b) CatBoost – CatBoost is an open-source software library developed by Yandex. It provides a gradient boosting framework which attempts to solve for Categorical features using a permutation driven alternative compared to the classical algorithm.

XGBoost - XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting that solve many data science problems in a fast and accurate way.

CHAPTER 8
CONCLUSION

In our project we aim to propose a new model for keyword extraction by extending the Phaseformer Model proposed by N. Nikzad–Khasmakhi.We selected and preprocessed the SemEval 2010 , SemEval 2017 and Inspec datasets for our project.We performed text learning by first studying the structure and applications of BERT model in the field of Natural Language Processing (NLP) and using Sentence Transformer to find BERT embeddings for our dataset.We studied the various graph embedding techniques such as DeepWalk , Node2Vec , SDNE, ExEm etc and applied them for structural learning. We found the structural vector for every word based on these graph embedding techniques. Finally, word representations for our sequence labelling and classification task were found by combining the BERT embeddings and Graph embeddings.

REFERENCES

[1] "keyword-extraction," [Online]. Available: https://www.edia.nl/keyword-extraction.

[2] E. Gopan, S. Rajesh, G. Vishnu, A. Raj R. and M. Thushara, "Comparative Study on Different Approaches in Keyword Extraction," 2020, pp. 70-74.

[3] K. Bruce and B. Chad, "Learning user information interests through extraction of semantically significant phrases," 1996.

[4] 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.

[5] K. Sparck Jones, "A STATISTICAL INTERPRETATION OF TERM SPECIFICITY AND ITS APPLICATION IN RETRIEVAL," *Journal of Documentation,* vol. 28, pp. 11-21, 1972.

[6] M. A. Andrade-Navarro, "Automatic extraction of keywords from scientific text: application to the knowledge domain of protein families," *Bioinformatics,* vol. 147, pp. 600-7, 1998.

[7] P. Turney, "Learning Algorithms for Keyphrase Extraction," *Inf. Retr.,* vol. 2, pp. 303-336, 2000.

[8] K. Zhang, H. Xu, J. Tang and J.-Z. Li, "Keyword Extraction Using Support Vector Machine," 2006, pp. 85-96.

[9] G. Ercan, "Using lexical chains for keyword extraction," *Information Processing & Management,* vol. 43, pp. 1705-1714, 2007.

[10] N. Nikzad Khasmakhi, M. R. Feizi Derakhshi and M. Asgari-Chenaghlu, "Phraseformer: Multimodal Key-phrase Extraction using Transformer and Graph Embedding," 06 2021.

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