Screening resumes for jobs using nlp techniques

Nishit Reddy Lingala AshokManavarthi Mounika kandula

***Abstract*— When someone posts a job opening on the internet, they can expect to receive a large number of applications in a relatively short period of time. Because it takes a lot of time and incurs significant expenditures that hiring organizations cannot afford to endure, manually filtering out the resumes is not practically practicable. This is due to the fact that it takes a lot of time. In addition, the process of reviewing resumes is not fair because a large number of candidates who are a good fit do not receive the adequate consideration that they merit. Because of this, it is possible that the job will be filled by applicants who are not a good fit for the position, or that the ideal individuals would be overlooked. In this work, we describe a system that seeks to tackle these challenges by automatically suggesting the individuals that are the most appropriate according to the job description that is provided. Our method employs natural language processing to extract relevant information from unstructured resumes, such as skills, education, experience, and so on and then uses that information to generate a summarized form of each application. Recruiters are now able to perform a more thorough analysis of each resume in significantly less time as a result of the elimination of all superfluous material throughout the screening process. After this text mining process is finished, the suggested method applies a BERT vectorization model and makes use of cosine similarity to match each resume with the job description. After the ranking scores have been calculated, they can be applied to identify which candidates are the most qualified to fill a certain job opportunity.**

Keywords: NLP, BERTmodel, Cosine similarity, lemmatization, Term Frequency - Inverse Document Frequency

I. INTRODUCTION

The recruitment practices of all major corporations have evolved in response to the exponential growth of internet access. Recruiters can reach a large pool of potential candidates by posting jobs on a variety of online job boards and websites. While electronic recruitment has saved time and money for both employers and job seekers, it has also presented some novel difficulties. Numerous resumes are received daily by large corporations and employment agencies. The greater mobility of workers and the greater competition for jobs in times of economic turmoil only make matters worse. Considering that only a small fraction of applicants will be chosen, it would be impossible for recruiters to read over every single résumé individually. It's also difficult for employers to compare applicants because resumes are presented in a variety of formats. Various people have different educational and occupational backgrounds. Each applicant has a varied way of presenting his or her credentials in the CV due to differences in education, experience, and focus. Resumes are not standardized documents; they are available in a wide variety of file types (.pdf,.doc,.docx,.jpg,.txt, etc.) and do not adhere to any particular template or structure.

Because of this, it takes some time for recruiters to sift through applications and choose the best applicants. As a solution to the problem of managing a wide variety of resume formats, many job boards, and other websites have emerged. Candidates must manually enter all of their résumé information into a standardized web form, yielding candidate metadata.[1] The drawback of this method is that it forces candidates to put in more work, and they frequently forget to fill in necessary details while using these forms. The websites in question all adopt a one-size-fits-all design that isn't great for any one industry or role.

Employers can then utilize these samples to implement a candidate shortlisting process based on keywords. This keyword search feature is inadequate for finding people who are a good fit for the position. This is due to the fact that it is overly dependent on the presence of a small set of necessary keywords and suffers from a number of extraction limitations, such as failing to take into account the natural language semantics of the content in the resume, which includes things like synonyms, word combinations, and contextual meaning. As a result, these Boolean search techniques frequently provide useless results, preventing qualified applicants from being considered. Better outcomes for resume shortlisting can be attained by exploring alternative methods of matching candidates with open positions.

Through an analysis of how closely the primary characteristics of each applicant's profile match the requirements specified in the job description, our proposed method would select the most qualified individuals for any given opening. You can think of the system's operation as having two distinct phases. The first step is to go through the resumes for any nuggets of structured data that might be useful in making a hiring decision. In order to build a streamlined version of each resume, the system employs Natural Language Processing to extract the most pertinent qualification details and arranges them in a consistent fashion across all files. The assessor can swiftly examine the condensed form and evaluate

the candidates' credentials without having to go through unnecessary information. Secondly, our system ranks resumes depending on how well they match the job description. [2] Once the documents have been transformed into vectors, we can use similarity measures like cosine similarity to determine which group of applicants is the most qualified. A final outcome is a list of applicants in order of rank.

The following chapters make up this paper: The previous research in this area is outlined in Section 2. In Section 4, we introduce the specific approach we took and the theoretical ideas that underpinned our final product. In Section 5, we break down the framework of the system we've created. Experiments conducted with our system are shown in Section 6, our work is summarized and its future applications are discussed in Section 7.

II. LITERATURE SURVEY

The Natural Language Processing (NLP) Based Extraction of Relevant Resumes Using Machine Learning Technique paper described the parsing of resumes using the fewest possible rules to teach the call and address. The CV parser system is used by scout packs to identify resumes. Since resumes can be found in a wide variety of formats and contain a wide variety of genuine elements (e.g., structured and unstructured evaluations, meta experiences, etc.). The method of extracting components from relocated CVs is provided by the proposed CV parser approach.

Online Applicant Tracking Program The first step in the four-stage process of analyzing a candidate's resume, psychometric test results, and social media profiles involved acquiring the necessary data (resumes) and converting them into a structured format suitable for subsequent analysis with deep learning methods. In the second stage, candidates take a psychometric test in which their answers to open-ended questions are mined for information. Third, using the information gleaned from web scraping, they determine which social media sites are most relevant to each candidate and then make job recommendations to them.[3] As a final step, the system will advise students on how to fill any gaps in their education and help them land jobs with their dream companies.

A Successful Resume Using Neural Networks and Conditional Random Fields Neural networks and CRF fall under the parsing category and can be used to identify relevant resume sections and extract relevant data. It compares the performance of a CNN model for segmentation to that of a Bi LSTM model. In order to extract information, a CRF-based model is selected and compared to a Bi-LSTM-CNN model. By breaking down the categories of "person," "education," and "work," they were able to extract and organize a wealth of useful data. The outcomes are encouraging, with 23 data fields in the final JSON file.

Applying the Entity Extraction Process and Large Data Tools to Model a Curriculum Vitae Parser The problem was defined in terms of developing an automated system for parsing resumes, one that could categorize applicants' applications in accordance with specific job descriptions. And it'll make resumes that are currently in an unorganized format into a neat and tidy one. It will also keep track of how applicants are ranked. The extracted data will be used as the basis for the rankings, which could be based on technical expertise, academic achievement, or other factors. The Curriculum Vitae parser is put to use here. One such method is CV parsing, which is used to gather curriculum vitae. The CV parser is multilingual, has a semantic mapping for skills, works with job boards and recruiters, and can be easily modified.[4] Using hire ability for parsing allows us to obtain reliable results. With its help, users can easily become API keys for integration projects. In order to determine a person's name and address, the parser follows a set of rules. The majority of staffing agencies now use a CV parser to sort through applications. Given that resumes can be found in a wide variety of file formats and contain a wide variety of data, including structured and unstructured data, metadata, and more. The entity extraction method from the submitted CVs is provided by the proposed CV parser method.

An easy text analytic-based method for evaluating a group of resumes was described in the Unstructured Text Analytics Approach for Qualitative Evaluation of Resumes. This method involved evaluating resumes qualitatively based on a number of different quality parameters. [5]We modified the extracted ratings into a more all-encompassing quality rating after processing the resume pool for two qualitative coverage, comprehensibility, and the aspects. For the purpose of associating a quality metric for resumes, all the parameters were uninformed into a combined 1 to 5 rating scale. The algorithmic approach yielded qualitative evaluation results that were consistent with, and thus validated by, the wisdom of the crowds.

III. EXECUTION PLAN

- Dataset Collection
- Dataset
- Preprocessing
- Tokensing
- Vectorsing
- Modeling
- Performance Analysis
- Documentation.

IV. METHODOLOGY

Finding the resume of the ideal candidate from among all of the available resumes is the objective of this activity. We have come up with a solution that is based on NLP in order to

accomplish this goal. The whole structure of the model being suggested can be seen in the following image.

A. Data Preprocessing

1. Cleaning and Removing Special characters:

During this step, the attached resume (CV) that is being used as input will be cleaned by removing any special characters or trash characters that may be included in the CVs. During the cleaning process, all special characters, numerals, and words consisting of a single letter are deleted. After completing all of these processes, we were able to obtain a clean dataset that had no single-letter words, numerals, or special characters. Using the NLTK tokenizes [12], the dataset is broken up into its constituent tokens. In addition, the preprocessing procedures are applied to the dataset that has been tokenized. These steps include the removal of stop words, stemming, and lemmatization. After importing the raw CV file, the data in the resume field was purified by getting rid of the numbers and the excessive spaces in the date.

2. Stop words removal:

Since stop words like and, the, and was, as well as others like them, exist quite frequently in the text yet contribute nothing to the prediction process, they are eliminated. Following are the steps to filter out the Stop Words: The grammatical rules of the English language allow for a single word to take on multiple forms when used in different phrases. Examples of this include the verbs implement, implemented, and implementation. Because of this, it is necessary to strip down all the numerous forms of a word to its root so that terms that share a common derivational etymology are not arbitrarily distinguished from one another.

Though they both aim to do the same thing, Stemming and lemmatization go about it in different ways. The process of "stemming" returns a word to its original form after applying various inflectional or derivational rules. Chopping off the ends of words and sometimes even whole affixes in order to reach this goal is a simple heuristic operation (Jivani, A.G., 2011). These are rule-based algorithms that examine a word under various [12] circumstances and use a database of prefixes and suffixes to determine the best way to shorten it. It's important to note that the steaming process may produce a root that has nothing to do with the word's morphological root. Under- and over-stemming are problems that arise from stemming's heuristic foundation. Porter-Stemmer, Snowball, and Lancaster stemmers are three popular options. Lemmatization, on the other hand, is when a dictionary is used to accurately reduce root words. Lemmatization is a more careful strategy that employs language vocabulary and morphological analysis of words to provide linguistically proper lemmas, as opposed to Stemming, which merely cuts off tokens by basic pattern matching. Because of this, lemmatization can tell the difference between words with distinct meanings according to their parts of speech, just as it can tell the difference between singular and plural forms of the same word. Our software employs the NLTK python package's WordNet Lemmatizer (which is based on WordNew Database) to process English text.

3. Tokenization:

After converting the various resume formats into text (.docx,.pdf,.jpg,.rtf, etc.), we start the tokenization process to find phrases or words that constitute a character sequence. This is done by looking for character sequences that contain several terms or words. This is significant because, with the help of these words[13], we will be able to extract meaning from the sequence of the original text. The process of tokenization involves chopping up large sections of text into more manageable pieces known as tokens. This is accomplished by omitting or separating letters like as whitespace and punctuation symbols.

When paragraphs are tokenized, the sentences that result constitute the first stage of the tokenization process. These sentences are then further broken down into individual words. Through the process of tokenization, we are able to derive a variety of information, like the number of words included inside a text, the number of times a specific term appears within the text, and much more. The tokenization can be done in a variety of different methods, such as by utilizing the Natural Language Toolkit (NLTK), the spaCy library, or any of the other available options. Tokenization is an essential step that must come before continuing with text processing in any form, including the elimination of stop words, stemming, or lemmatization.[14]

B. TF-IDF

"Term Frequency - Inverse Document Frequency" is the abbreviation for this analysis method (Stecanella, 2020). Text mining methods frequently employ the TF-IDF weight. The original intent of the term frequency-inverse document frequency (TF-IDF) algorithm was for use in these two tasks. The significance of a given phrase in relation to a given document within a collection or corpus is quantified by the value of its weight.[9] If a word appears more often in the document, it becomes more significant. However, this effect is mitigated if the word appears in a larger number of papers. Common phrases that appear in numerous documents but don't add much meaning to any one of them, such as "this," "and," "what," "whom," "is," "the," "if," and so on, are given a low weight in the ranking system (Stecanella, 2020). Multiplying two metrics yields the TF-IDF value for a term in a document, as indicated below equation (Stecanella, 2020).

$$TF\text{-}IDF\ (t,d) = TF\ (t,d) * IDF\ (t,d)$$

Term Frequency: Term Frequency is a metric that determines how often a particular word appears in each of the documents that make up the corpus. Because a word could appear a greater number of times in longer papers than in shorter ones, you need to modify or normalize the frequency of this term's appearances. To arrive at a normalized term frequency, simply divide the number of times a term appears in a document by the total number of terms found in that document. This will yield the normalized term frequency. In terms of mathematics, we may express it as (Jabri, Siham, et al., 2018), which is displayed down below in the form of a below equation

$$TF\ (t,d) = \frac{freq\ (t,d)}{\sum_i^n freq\ (t_i,d)}$$

Here, n is the total number of unique words in d, freq (t, d) is the number of times t appears in d, and TF (t, d) is the fraction of its occurrences in d.

Inverse Document Frequency: As the name suggests, Inverse Document Frequency determines a word's significance relative to the total number of documents in the corpus. In other words, this metric helps to know how rare or common a term is across in the corpus. It weighs down the terms that appear more often while scaling up the rare terms. The IDF is close to 0 for terms that exist frequently in the set of documents, while it is high for terms that appear infrequently. It is calculated by dividing the total number of documents by the number of documents that contain a phrase and then computing the logarithm (Stecanella, 2020). (Stecanella, 2020). It can be written as the equation below, which is a mathematical representation of the solution.

$$IDF(t) = log\left(\frac{N}{count(t)}\right)$$

In this case, N represents the total number of unique documents in the database, and count(t) represents the total number of documents that include the term t.

The TF-IDF score of a word in a document is calculated by taking the product of these two metrics, which are represented by equations (2) and (3). A higher TF-IDF score for a word indicates that the word has greater significance in the context of the document. In our approach, we modeled both the job descriptions and the resumes into a vector space. This allowed us to compare and contrast the two. Creating a dictionary of terms that are found in the documents and translating each phrase to a dimension in the vector space are the steps that need to be taken in order to accomplish this goal. After that, we used the CountVectorizer and the TfidfTransformer python modules in order to compute the TF-IDF matrix for the CVs as well as the job query. The following thing that needs to be done is a calculation to determine the degree of similarity that exists between the job description and the resumes.

*C. **BERT Model***

Bidirectional Encoder Representations from Transformers, or BERT for short, is a strategy for pre-training natural language processing (NLP) systems that were developed by Google.[6] BERT is a transformer-based machine learning method. Jacob Devlin and his fellow Google employees are responsible for the creation and publication of BERT in 2018. The foundation of BERT is a transformer language model that can accommodate a varying number of encoder layers and self-attention heads. The architecture is "nearly identical" to the implementation of the original transformer that was published in Vaswani et al (2017). Language modeling (15 percent of tokens were hidden, and BERT was trained to deduce their meaning from context) and next sentence prediction were the two tasks that BERT was pre-trained on (BERT was trained to predict if a chosen next sentence was probable or not given the first sentence). [7]

The contextual embeddings of words are something that BERT picks up as a result of the training process. After pretraining, which requires a significant amount of computational power, BERT can be fine-tuned using fewer resources and focusing on datasets that are smaller in order to optimize its performance on particular tasks.

BERT is essentially a transformer architecture Encoder stack. An encoder-decoder network with self-attention on the encoder side and attention on the decoder side is referred to as a transformer architecture. The Encoder stack of BERTBASE has 12 layers, whereas the Encoder stack of BERTLARGE has 24 layers. This is in addition to the Transformer architecture described in the original paper (6 encoder layers). BERT architectures (BASE and LARGE) have larger feedforward networks (768 and 1024 hidden units, respectively), as well as more attention heads (12 and 16, respectively) than the Transformer architecture proposed in the original paper. It has 512 hidden units as well as 8 attention heads. BERTBASE has 110M parameters, whereas BERTLARGE has 340M.

This model takes a CLS token as input first, then a sequence of words as input. CLS is a classification token in this case. It then sends the input to the layers above. Each layer performs self-attention, then passes the result through a feedforward network before handing off to the next encoder. The model generates a hidden-size vector (768 for BERT BASE). We can use the output corresponding to the CLS token to generate a classifier from this model.

The goal of the masked language model is to predict, from the context alone, the original vocabulary id of a masked word that was removed at random from the input. In contrast to pre-training a left-to-right language model, we are able to train a deep bidirectional Transformer by having the representation fuse the left and right context. BERT employs a masked language model and a next sentence prediction challenge to jointly pre-train text-pair representations.

Pre-training and tuning are the two stages of BERT. The model is initially trained using unlabeled data across a variety of pre-training activities.[8] All of the BERT model's parameters are fine-tuned using labeled data from the downstream tasks after being started with the pre-trained ones. Models for each downstream job are tweaked independently, although sharing the same pre-trained parameters.

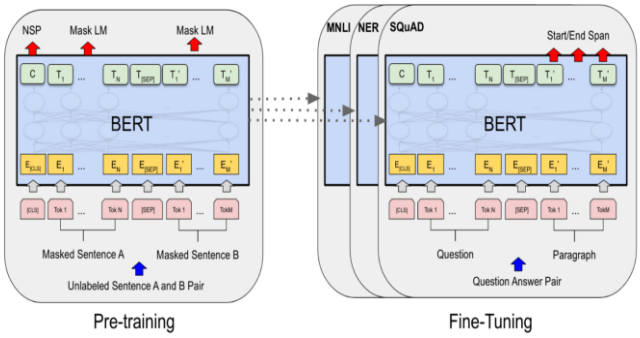

Fig 1. BERT pre-training and fine-tuning

D. *Cosine similarity*

Similarity measures are quantitative indicators of the degree to which two entities are equivalent. When comparing two texts of varying lengths, cosine similarity (Sidorov, Grigori, et al., 2014) is a useful metric. When documents are plotted in N-dimensions, where each dimension represents a different aspect of the object, the resulting diagram illustrates the documents' preferred direction. [10] Due to its symmetric nature, the results of computing the similarity of item X to item Y are identical to those of computing the similarity of item Y to item X. We can express this in mathematical terms using the equation below

$$cos(\theta) = \frac{\vec{a}.\vec{b}}{\left|\left|\vec{a}\right|\right|\left|\left|\vec{b}\right|\right|} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

The cosine similarity between any two elements can be found using this formula. After that, you may use it to sort resume papers according to a keyword query vector. Cosine similarity, on the other hand, narrows in on variables directly related to the text's words and so produces less reliable conclusions. [11] Incorporating semantic information into similarity measures can boost their effectiveness. What we've just described is the potential application of our automated resume screening technology.

V. SYSTEM ARCHITECTURE

Building a content-based job recommendation system that uses the BERT Model to calculate the similarity between the content of candidate resumes and the job requirements in order to recommend the best-fitting candidates to the employer is the fundamental solution to the problem.

This is the solution that should be implemented first and foremost in order to solve the issue. The following outlines, on a more general scale, the fundamental process that underlies this system.

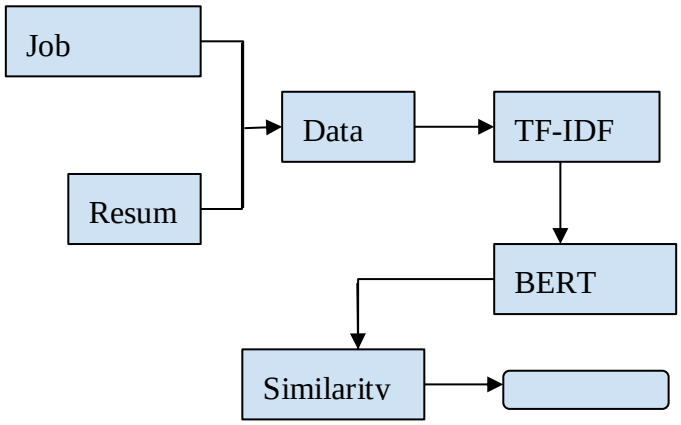

Fig 2. Resume analysis process

- The first thing to do is load up the resumes.
- Step 2 involves loading the job description, followed by Step 3, which involves cleaning up both the resumes and the job description.
- Step 4 involves utilising the TF-IDF technique to isolate the most significant terms.
- The fifth step involves applying the Bert model to transform the extracted words into vectors.
- The score will be determined based on the comparison of the vectors in step 6.

VI. RESULT

In order to build a ranked list of candidates for this job, we will pre-process, extract, summarise, and compute cosine similarity on some relevant resume examples gathered from the Internet. There are 13 Job Descriptions available. From that, one should be selected

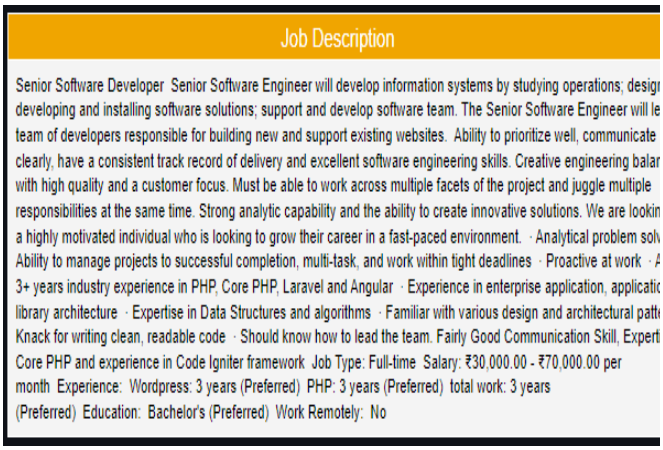

Fig 3. Job description

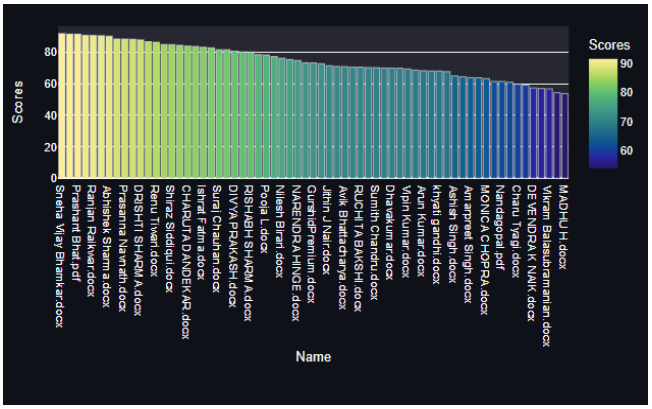

Fig 4. Score and Rank distribution

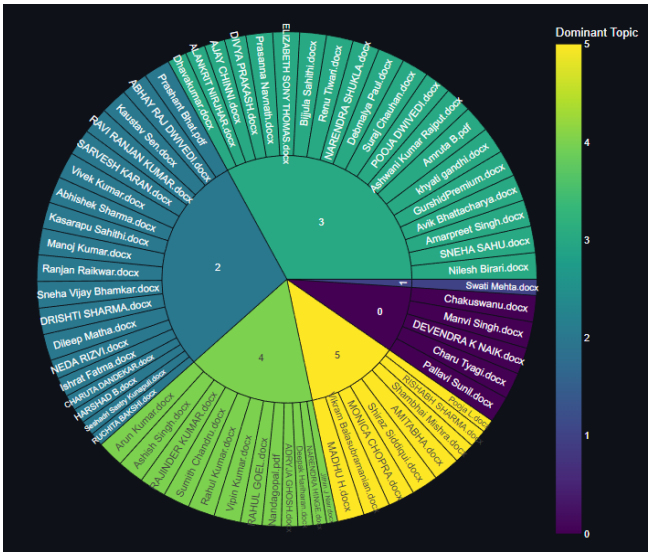

Fig 5. Topic distribution graph

Fig 6. The word cloud for the resume with match score

VII. CONCLUSION

Insanely high volume of resumes sent to the company for each open position. Employers today face a mountain of resumes, and sifting through them to find the right employee is a time-consuming and laborious process. Manually sorting through resumes is a time-consuming and resource-intensive procedure. We present a Natural Language Processing-based automated model that does just that, recommending resumes of qualified candidates to HR based on a particular job description. In this work, we introduced a resume screening technology that streamlines the recruitment process by removing the many obstacles experienced by human recruiters when shortlisting candidates for a specific job opening. Our method is effective in two ways. The first step is to utilise NLP to sift through the varied and chaotic formats of the resumes for useful nuggets of information. In the end, it generates a condensed version of each resume that includes only the data points that matter for making the final cut. The screening officials' job is much easier without all the extraneous details, and they are able to analyze each résumé more thoroughly. To counter this, we have built in a scoring system that compares the extracted resume attributes to the requirements listed in the job description, using a content-based algorithm based on the BERT Model and similarity. Each resume is assigned a similarity score, and the results are used to rank the best N applicants in order of how well they match the job description. The suggested method successfully captures the semantics and insights of resumes, resulting in an accuracy of 91.85%.

VIII. REFERENCES

[1] Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E.D., Gutierrez, J.B. and Kochut, K., 2017. A brief survey of text mining: Classification, clustering, and extraction techniques. arXiv preprint arXiv:1707.02919.

[2] Arguello, J., 2013. Vector space model. Information Retrieval September, 25.

[3] Berry, M., 2001. Computational Information Retrieval. Philadelphia: Society for Industrial and Applied Mathematics,121-144.

[4] Bird, S., Klein, E. and Loper, E., 2009. Natural Language Processing With Python. Bejing: O'Reilly, 264.

[5] Faliagka, E., Ramantas, K., Tsakalidis, A. and Tzimas, G., 2012, May. Application of machine learning algorithms to an online recruitment system. In Proc. International Conference on Internet and Web Applications and Services.

[6] Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina (11 October 2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding".

[7] "Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing". Google AI Blog. Retrieved 2019-11-27..

[8] Horev, Rani (2018). "BERT Explained: State of the art language model for NLP". Towards Data Science. Retrieved 27 September 2021.

[9] Jabri, S., Dhabi, A., Gadi, T. and Bassir, A., 2018, April. Ranking of text documents using TF-IDF weighting and association rules mining. In 2018 4th International Conference on Optimization and Applications (ICOA), 1-6. IEEE.

[10] Huang, A., 2008, April. Similarity measures for text document clustering. In Proceedings of the sixth new Zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand, 4, 9-56.

[11] Dr.Parkavi A, Pooja Pandey, Poornima J, Vaibhavi G S, Kaveri BW, “E-Recruitment System Through Resume Parsing, Psychometric Test and Social Media Analysis”, 2019, IJARBEST.

[12] Satyaki Sanyal, Neelanjan Ghosh, SouvikHazra, Soumyashree Adhikary, “Resume Parser with Natural language Processing”, 2007, IJESC.

[13] Stenella, B., 2020. What Is TF-IDF? [online] MonkeyLearn Blog. Available at: <https://monkeylearn.com/blog/what-is-tf-idf/>.

[14] Mansouri, A., Affendey, L.S., and Mamat, A., 2008. Named entity recognition approaches. International Journal of Computer Science and Network Security, 8(2), 339-344.