Research Paper for Course

CMPE-256

Web Data Mining

First Semester, 2020

San Jose State University, California

**Group -** MinersGO

**Team Members**

Anirudh Ganguly [015268071]

Riddhi Jain [014600716]

Shreya Goyal [015251990]

Resume Retrieval System NLP And Distributed Computing Techniques

Anirudh Ganguly, Riddhi Jain, and Shreya Goyal

**Abstract**—One of the keys to success for a business in a competitive market environment is to find and retain the right talent. It is extremely challenging for human resources to find the right candidate for the right role in the organization. If done manually, the HR team needs to manually scan through resumes from multiple sources and try to categorize them with their intuition or experience. This is a time consuming and tedious process, prone to errors.

This paper proposes a system using natural language processing and distributed computing techniques to automatically rank and retrieve a plethora of profiles available, that are relevant to the employer’s needs. In the proposed solution, the employer will get a web based interface to enter a free text query for example “linux experts with experience in distributed computing” and will get a result of top ranked matching profiles. This will save the employer’s time to filter and narrow down on the best matching profiles. The task of ranking and retrieving the profiles based on user query has two parts: first part is to convert the unstructured resumes into a structured data format and the second part is to use the extracted features and apply similarity comparison. Finally, the paper describes how the Hadoop framework and Glove Embedding techniques can be used to improve the performance of the retrieval system.

**Index Terms**—Clustering, Information Search and Retrieval, Hadoop, MapReduce, Glove Embedding, Natural Language Processing

—————————— ◆ ——————————

# 1 Introduction

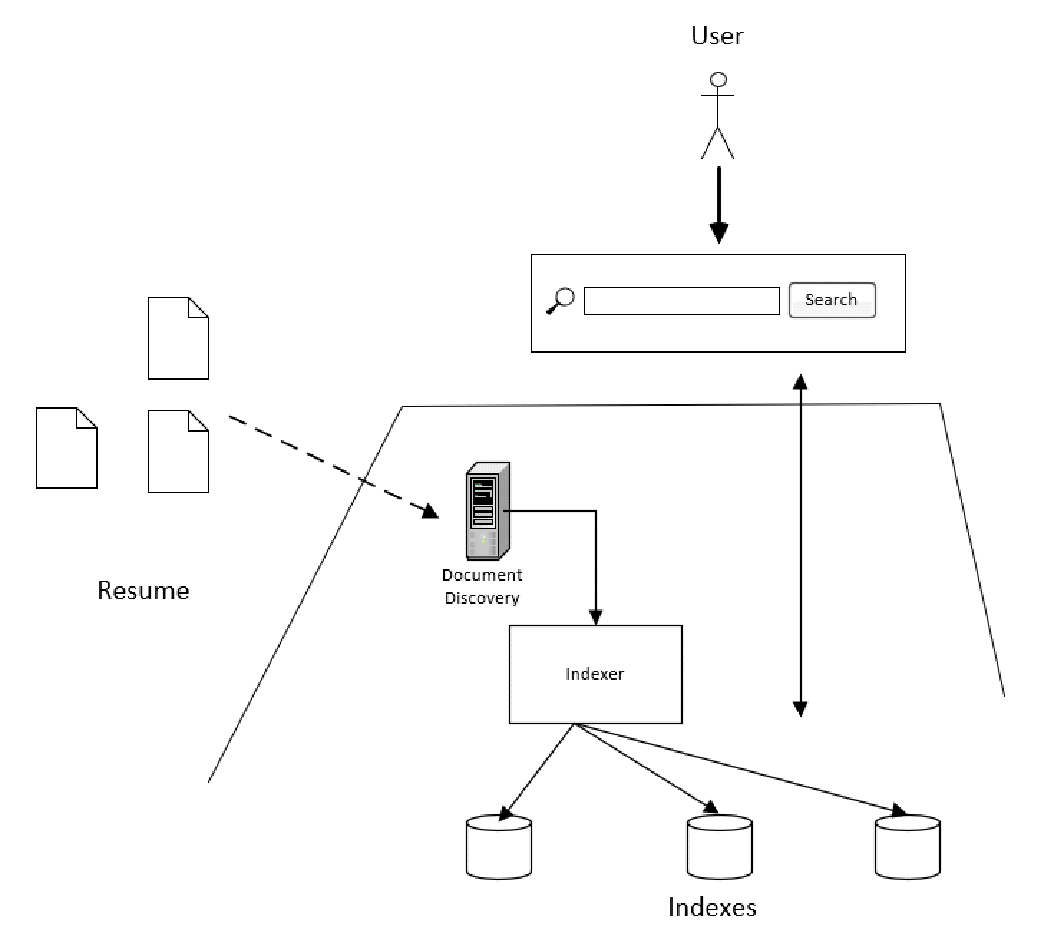
Searching for an ideal candidate to fit perfectly for a role in an organization has always been a challenging and time-consuming task. In recent years, the Human Resource department of an enterprise has started to receive enormous amounts of resumes from candidates having varying skill sets, work experience, and academic background. There is no standard format for writing resumes and candidates are showing their leading-edge in resume building with different styles( text, tables, etc) and formats. With companies tapping talent from the global market online market of resumes, the volume and diversity of resumes are growing exponentially with each day. As the number of potential candidates is growing, a significant challenge is to manually go through all the resumes and filter them based on the different profile requirements for different hiring managers. To handle this problem, job seekers are given out templates to fill at the time of applying for a role. It had helped the organization, but introduced an unnecessary overhead for the candidates as each time they needed to fill different forms while applying for a job. Additionally, organizations also have to invest resources and time for maintaining that application and updating it with the change in the market needs.

This paper focuses on developing an information retrieval system where a user will give some keywords and those words will be used for filtering out the ideally suited set of candidate profiles that best match the job description from the large pool set. Based on the keywords entered in the search bar, the system will quickly scroll through all the available documents with different formats and return the top-ranking profiles to the user. The query in the proposed system combines words defining job descriptions or key skill sets. The result obtained is the list of matching profiles. Also, the paper proposes a way to improve the efficiency and speed of the retrieval results further by using a clustering technique on the documents provided. Clustering will be used in finding the similarities between the unstructured data and then grouping them together for processing.

To analyze a huge amount of unstructured data, we propose a model that will use Natural Language Processing technique for information interpretation. It is a two-step model. In the first step, it will convert the unstructured data to structured data by extracting keywords. The structured data stored in the database is used to extract relevant information and find insightful patterns. Since the amount of data available is enormous, in the second step, an approach of feature extraction and ranking is a compelling direction to make the workflow more streamlined and effective. If successfully implemented, the proposed solution can save an enormous amount of time allowing an organisations HR team to focus on other useful tasks and will also speed up the recruitment process. However, developing a 100 percent automated system with high accuracy and consistency across all formats of data is a challenging task.

The available set of documents is called the corpus. Here, a document is a resume containing unstructured text data. From this point onwards, a resume and document will be referred interchangeably since this retrieval technique can be applied to any text data document. As the size of the corpus would be huge and diverse, the system needs to pre-process those documents for faster retrieval. The success of the system will be measured in terms of how precisely the system returns the relevant documents and at what cost (time and memory). Since, the corpus will be dynamic which means documents will get added or deleted at any point in time. Therefore, any type of processing on the document needs to be dynamic. The paper begins with briefly explaining the high-level system components and then the rest of the paper describes the retrieval process and mechanisms to improve the retrieval process using parallel computing and context based natural language processing technique.

# 2 High Level System Design



*High Level System Design*

At a high level, the resume retrieval system consists of two components: the backend server and the user interface as shown in the above figure.

## 2.1 Backend server

The backend server is responsible to discover documents and pre-process them. The documents can be collected from various sources and at any point of time. A document discovery service waits for any new document to be available. The document discovery service could be a simple service that listens to any changes to a folder in the file system. For the purpose of the project, we have implemented a simple folder scanner in place of the document discovery service. Once the document is available, it is dispatched to the indexer module to pre-process. Indexer is a component that performs the pre-processing operation. The pre-processed indexes are stored in a database file on disk. Lastly, the backend server runs a web service that listens to web requests for a front-end client.

## 2.2 User Interface

The user interface is delivered as a web application running on a browser. The web app provides a search bar where users can type in any text - it could be any key skillset or the job profile. The web app makes a request to the web server and displays the response - list of matching profiles, to the user.

# 3 Retrieval Process

## 3.1 Pre-process of resume text

First, the resumes are pre-processed by removing any non-alphanumeric characters. Next, stop words like ‘is’, ‘the’, ‘and’ are removed as they add no value to the algorithm. Finally, stemming and lemmatization is applied on each work to bring it into its base form. The above-mentioned operations on a text document can be done using the NLTK library in python.

## 3.2 TF-IDF features

TF-IDF stands for term frequency-inverted document frequency, and it is an important feature that is extracted for many text mining tasks. Term frequency of a term t refers to the ratio of number of occurrences of the term in a given document d and the total number of unique terms in the document. Here, the term refers to a single word in the document. Inverted document frequency refers to the ratio of total number of documents in the corpus and the number of documents containing the term t. The product of TF and IDF is the TF-IDF weight of a term in a document. The TF-IDF weight tells the significance of the term in a given document.

The following steps are performed to convert each document into a vector model.

* A vocabulary set of all unique words from the pre-processing step is created.
* A zero vector is created for each document with the same length as the length of the vocabulary set.
* TF-IDF weight is calculated for each word in every document and the value is updated into the corresponding index of the vector.
* For each document, a sparse vector is created and all the vectors are stored in a repository.

## 3.3 User Search

When a user provides a text query which could be some keywords or a job description, the query text is also converted into a feature vector as described in section 3.2. Then the cosine similarity is calculated between the query vector and each document vector from the repository. The documents with highest cosine scores are retrieved and returned as a result to the user.

## 3.4 Document clustering

The documents can be clustered by using the generated TF-IDF vectors and cosine score as a measure of distance between the documents. Once the TF-IDF vectors are generated for each document, unsupervised clustering algorithms like K-means can be used to cluster the documents. K-means is a centroid based clustering algorithm that groups data items closest to a calculated centroid as a single group.

Once the documents are clustered, query performance in terms of recall and precision can be improved. Moreover, the time takes to retrieve the results can be significantly reduced. This is because, if a document gets a high cosine score with the user query, it's likely that other documents in the same cluster will also have similar scores. The basic assumption is that resumes with similar profiles get grouped together.

# 4 The Hadoop Framework

Open source framework which works on distributed computing to process large dataset ranging from gigabytes to petabytes on multiple computers. The core component of Hadoop framework is Map reduce. It is a technique for distributed computing. In other words, used to split the large dataset into small parts to process the parts of the dataset parallely to speed up the execution. Two main tasks are Map and reduce jobs.

* Map stage: Mostly it works on input data and input data is used to be in the form of files stored in HDFS (Hadoop File System). Mappers job is to divide the large dataset as input and processes that dataset to return another smaller chunks of dataset in the form of a key value pair.
* Reduce stage: This is a shuffle and combine stage. Output from mappers job works as work as input to the reduce jobs which used to combine those multiple pairs into lesser no of pairs. Mapreduce implies that map task is always performed before the reduce task as output from the map will be given to the reduce job.

## 4.1 TF-IDF feature extraction using MapReduce

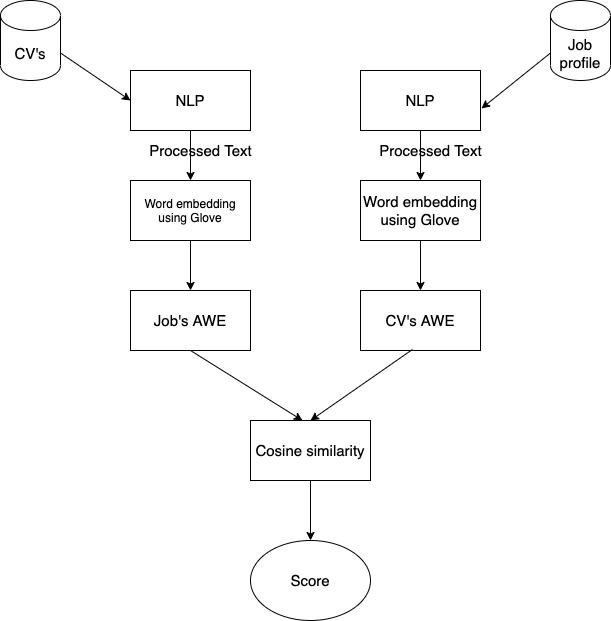
The main idea is to distribute work across multiple compute machines and get the work done parallely. The main task is to find the term occurrences in a document and the number of documents a term occurs in. This can be done in parallel by breaking down the documents into different bins. Once the term frequency is known, the inverted-document frequency can also be calculated in a distributed fashion.

Mapreduce works on the basis of key value pairs. TF-IDF values can be fetched using a key value pair to improve the performance of the model. Whole process can be divided into three mapreduce jobs. The first map reduce job will find the count of each word in each document. Once word frequency has been counted, the reducer job is to find the total number of occurrences of words in each document. Second map reduce job is designed to find the term frequency which can be passed as input to the third map reduce job. Third one calculates the inverse document frequency and combines the term frequency and inverse document frequency to calculate the score for each word.

# 5 Glove Embedding

It is one of the most popular techniques of natural language processing. It works on the dense vector representations. It is an unsupervised algorithm which works on the importance of word-word co-occurrence to extract the meaning from the sentence. This is used to convert words into vector space models with semantic context also. Words with similar context will take position close to each other. This way it will help us to find the word with the similar meaning.

After converting words into vector models using Glove embedding, cosine similarity can be applied to find degree of similarity between two words. It will help to find the similarity using the angle between the vectors not the magnitude.



*Architecture of model using Glove embedding*

1. Data in unstructured format is converted into structured format using NLP technique by removing extra spaces, tokenization , lemmatization etc.
2. Processed text is used to create a vector model for each word using glove embedding.
3. Use the word embedding model and take the average of word embedding to create a model for the document level.
4. Find the similarity to fetch matching resumes based on job description using cosine similarity.

# 6 Conclusion

This paper proposed a simple approach of using TF-IDF features to convert unstructured resumes onto a vector model and then rank the resumes based on user query. The paper also describes a mechanism to improve the performance of the retrieval process by clustering the resumes using an unsupervised KMeans clustering algorithm. Lastly, to improve the performance of the retrieval process, the paper talks about using Hadoop framework for document preprocessing and Glove embedding for context based document ranking.

**7 Acknowledgment**

We would like to thank our professor Chandrasekar Vuppalapati for giving us the opportunity to learn and write this paper.

**8 References**

1. <https://medium.com/@blogsupport/text-analytics-for-talent-acquisition-9e17a1043000>
2. <https://hadoop.apache.org/>
3. <https://en.wikipedia.org/wiki/Tf%E2%80%93idf>
4. CMPE - 256 course materials