Resume Retrieval System using a combination of NLP and Clustering Techniques

Anirudh Ganguly, Riddhi Jain, and Shreya Goyal

Group - MinersGO

GitHub Link - <https://github.com/rridhi-jain/resume_retrieval>

Project Report for course

CMPE-256 Web Data Mining

First Semester, 2020

San Jose State University, California

**Project 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. Our project proposes a system using natural language processing 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 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: the 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.

**Table of contents**

[**1 Introduction**](#_heading=h.v1k5xd88xr3f) **3**

[**2 Requirements**](#_heading=h.tm5dtmknv1d3) **3**

[2.1 Software requirements](#_heading=h.antocamjvlg0) 3

[**3 KDD process**](#_heading=h.cb9zyssx1i43) **3**

[3.1 Data Collection](#_heading=h.c9sy3n6p6fnd) 3

[3.2 Data Integration](#_heading=h.bcvjqisjij4o) 3

[3.3 Data Transformation and Selection](#_heading=h.x5lpjdjw9byc) 4

[3.4 Data Mining](#_heading=h.c3cwwwjnd0j2) 4

[3.5 Evaluation](#_heading=h.4rterxu239ho) 4

[3.6 Knowledge Presentation](#_heading=h.t6112zs27m45) 4

[**4 Feature Engineering**](#_heading=h.qmtzxoxs0p70) **4**

[**5 High Level Architecture**](#_heading=h.u771dmkb1dp1) **4**

[5.1 Backend server](#_heading=h.4ts3dxtmvsm) 4

[5.2 User Interface](#_heading=h.wflzq7gl83qk) 5

[**6 Data Flow Diagram & Component Level Diagram**](#_heading=h.52kmwzn9rxwa) **5**

[**7 Sequence & Workflow Diagram**](#_heading=h.d9tbvimg45fy) **5**

[**8 Data Science Algorithms & Features Used**](#_heading=h.rp1npokd4ir) **5**

[**9 RESTFul interfaces**](#_heading=h.9vse2m3xfd6o) **6**

[**10 Client Side Design**](#_heading=h.p50s25tv1gd4) **6**

[**11 Design Patterns**](#_heading=h.qj80131wjxi3) **6**

[11.1 Client Server Design Pattern](#_heading=h.tm0pw4mqyn7j) 6

[**12 Testing**](#_heading=h.grcc4bavojq9) **6**

[**13 Model Deployment**](#_heading=h.5qwkovogccfl) **6**

[**14 Data Engineering**](#_heading=h.uaa25h3qcwaz) **6**

[14.1 Data Acquisition](#_heading=h.7cs21trscqi9) 6

[14.2 Data Transformation](#_heading=h.7bisljqme9wq) 6

[14.3 Data Load](#_heading=h.3e0di48xbhuf) 6

[**15 Conclusion**](#_heading=h.2n9aim11ip4y) **6**

# 

# 

# 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 report is on the project that 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, our project work 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 makes the workflow more streamlined and effective. Our project solution can save an enormous amount of time allowing an organization's 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 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).

The dataset used for our project is a collection of resumes from varied profiles and backgrounds that we took from the internet.

URL:- <https://drive.google.com/file/d/17M9oDPip5JFFFNJhDCBQKy8BMqoyxajU/view>

Credits: JAIJANYANI

# 2 Requirements

There are two requirements of the resume retrieval system:

1. Given a job description, profiles of matching candidates should be retrieved ranked by relevance.
2. Given a resume or profile, retrieve similar profiles.

## 2.1 Software requirements

The software requirements for this project include:

* Python
* NLTK

# 3 KDD process

## 3.1 Data Collection

In this phase of the KDD process, the primary focus is on collecting a large sample of resumes that will be required to support our information retrieval system. Resumes related to every domain such as Software Engineers, Accountant, Human Resource, Sales, Administrative, etc is collected from all over the internet. All the dataset collected is in unstructured format for processing it using NLP techniques.

## 3.2 Data Integration

This is the process in which data from different sources and different formats are clubbed together. The unstructured resume dataset is in pdf, docx and doc format. All the resumes are stored under one bucket for simple access and further processing. The collection of documents information is stored in a database where every row represents the document name and root path of the document location.

## 3.3 Data Transformation and Selection

In the transformation step, we need to convert unstructured data into a structured format which will be used for data modeling. We read the text data in the document and split the words with spaces, punctuations and sentences. The text data is tokenized using word tokenizer which splits the data with spaces and stores them into an array. Natural Language Toolkit is then used for further transformation of the words. All the stop words present are removed from the documents as these words are commonly available in all the documents and don't contribute much towards the outcome of the model.

After removing stopwords, any alphanumeric words if present is also dropped from the words set. Next, we perform lemmatization and stemming on the words. In this, the porter stemmer algorithm is applied to convert all the words to their natural or base forms for better processing. For example, ran is converted into run and treated equally while matching the similarity of the document.

## 3.4 Data Mining

In this step, data mining algorithms need to be implemented on the transformed data available to us.

Firstly, we will apply natural language processing techniques to convert the array of words into a TF-IDF vector space model. This TF-IDF model is called the vector space model. The vector model is then stored inside the database corresponding to each document. The vectorizer model developed after fitting the dataset is saved as a pk file in the system.

Secondly, KMeans unsupervised clustering technique is used to group similar documents into a single cluster. One number is assigned to each cluster. The cluster number so assigned is then stored in the database corresponding to every document row.

## 3.5 Evaluation

The model is evaluated manually by analyzing the result of a set of around 100 keywords. Resumes returned are manually checked with the occurrence of search string passed present in the document.

## 3.6 Knowledge Presentation

Users are given a user interface with a search bar. They can enter the job description and hit the search button to get the top resumes. The resume retrieved after each search process is rendered in the form of a list view to the user. The list contains the top 5 matched resumes based on their cosine similarity. Information about the document name and the link to the document is displayed to the user.

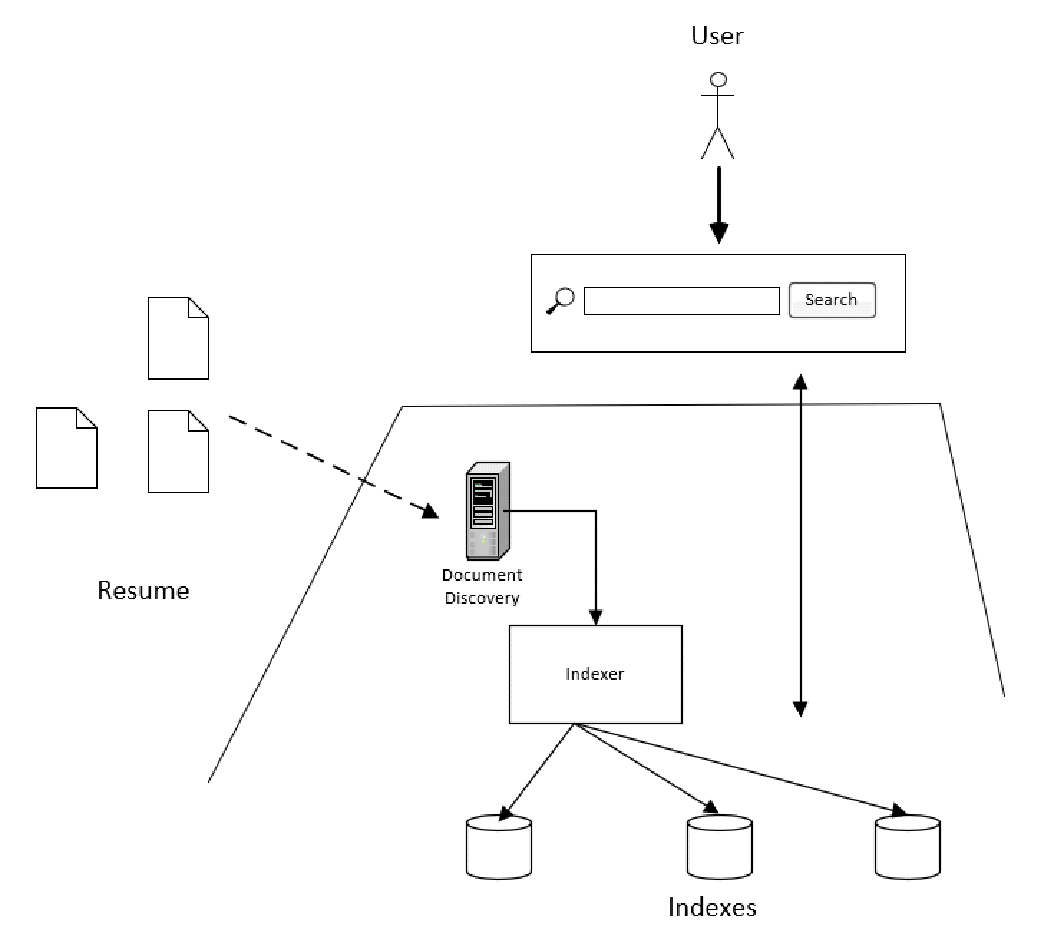
A ‘similar documents’ button is also visible to the user in the resume list. On clicking, the list of all documents belonging to the same cluster as that of the document being passed is returned as the result. The result is again shown as the list with the information regarding the name and the URL of the document.

# 4 Feature Engineering

Feature engineering is an important step in the development of the model. In NLP, feature engineering involves converting raw text data into a vector space model. The occurrences of each word found in the document dataset are counted and stored into a 2D array where a row represents the document and the columns represent the terms describing the document. The frequency of each word is then used to calculate the term frequency and the inverse document frequency of the respective words. The TF-IDF/vector model is developed by calculating the term frequency of all the terms describing the resume. The model is converted into a JSON and saved inside the database for quick processing.

In KMeans, feature engineering is to group the document with high similarity under the same cluster. The cluster information is then saved in the database. For getting the efficient value of K, we used the elbow method heuristic to determine. It draws the graph showing the relation between the percentage of variance on the y-axis and the K value on x-axis. Value closer to the origin is selected for grouping the similar corpus.

# 5 High level Architecture



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.

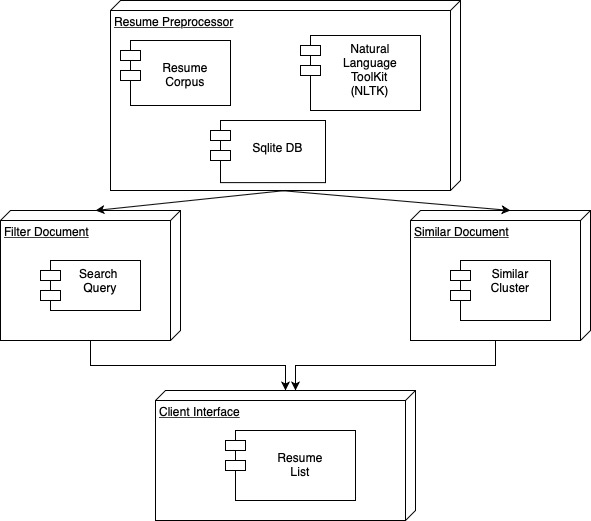
## 5.1 Backend server

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 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. An 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.

## 5.2 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 requests the web server and displays the response - list of matching profiles, to the user.

# 6 Data Flow Diagram & Component Level Diagram



The system consists of 3 main components:

1. Client Interface

The user is given a client-side interface to visualize the filtered documents list returned. It also helps in displaying the list of similar documents in the dataset.

1. Filter Document

A search query string is passed based on which the documents matching the similar criteria are fetched from the database.

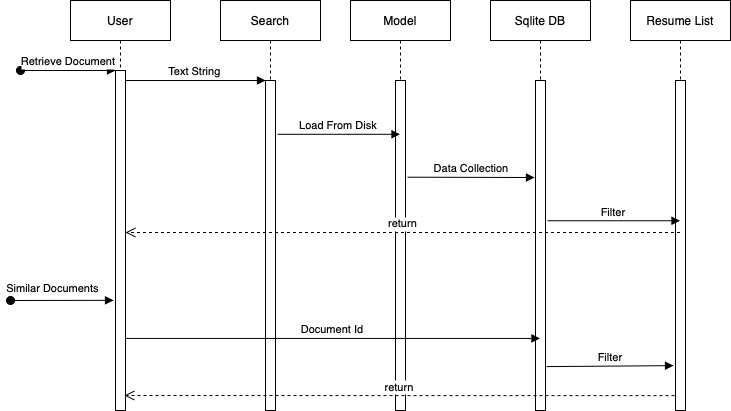
1. Similar Document

Based on the document passed as an input, the list of other related documents are returned to the user.

1. Resume Processor

The resumes are stored locally in a folder and are processed at the time of initial setup. Using NLTK the documents are scanned and converted into a vector space model.

# 7 Sequence & Workflow Diagram



The following is the data flow diagram of the application:

**Retrieve Resume based on the search string**

1. The user provides a text query string to the search API.
2. The API loads the TF-IDF vector model from the disk.
3. The query string is passed to the model and converted into a vector space model.
4. All the documents in the SQLite database are then compared with the query string TF-IDF vector.
5. Documents with high cosine similarity are then returned to the user as the filtered list.

**Retrieve similar resumes based on a resume passed**

1. The user clicks on a similar document button and passes the document id to the API.
2. The API filters all the documents stored in the database and results in only those documents that have high similarity with each other.

# 8 Data Science Algorithms & Features Used

To convert the unstructured text data from the resume to a vector space model, the term frequency - inverted document frequency feature engineering algorithm is used. TD-IDF is an important feature that is used in many text mining applications. The TF-IDF of a term in a document tells the significance of the term in the document.

To find the similarity of between the resumes and the user query, cosine similarity is used. Cosine similarity is the best measure for this use case as the size of the document does not affect the result.

In order to cluster the resumes, the KMeans algorithm is used. KMeans is an unsupervised learning algorithm that is used to cluster data points. Cosine similarity is used as a measure of the distance between the documents.

# 9 RESTFul interfaces

Following are REST APIs exposed by the backend system to deliver the features in the system

1. /setup POST - This API is called by the client to initiate the document preprocessing step. The documents are converted into TD-IDF features and the model is saved in a database file.
2. /search GET - This API is called to get matching resumes for a user provided search query. The query could be certain keywords or a job description.
3. /similar GET - This API is called to get similar resumes for a given resume.

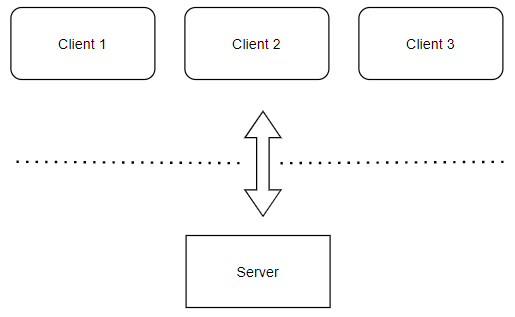
# 10 Client Side Design

The client interface is built using the Python Flask framework. The homepage consists of a blank search box to enter the string similar to which resumes are required. Retrieved resumes are shown in the list template.

A ‘Find Similar’ button in front of every resume is used to get all other similar documents. The resumes returned are displayed in a list view. ‘Do it again’ button is available to take the user back to the homepage with a search box.

# 11 Design Patterns

## 11.1 Client Server Design Pattern

****

For our project, we have used the client-server design pattern to break our system into two parts. The advantage of using this design is that we can run the client on the browser from anywhere as long as it knows the address of the server. Also, we can have multiple clients making simultaneous requests to the server. Any change in the backend server will not have an impact on the client code as long as the interface between the client and the server is not changed.

# 12 Testing

The search model so developed is analyzed by manually verifying the search api response. A job description string is passed, processed and the resumes returned are verified by opening and going through its content.

Implemented testing with around 20 keywords set. Modified the process repeatedly to improve the accuracy of the model. In the future, we can automate the manual process with a python script to cover a large set of keywords.

# 13 Model Deployment

The model is deployed on a local server. The prerequisite for the requirements is to get the latest codebase, latest python version, and run the flask deployment script.

# 14 Data Engineering

Data engineering is the field of science that focuses on collecting, modifying, and storing the data in a data warehouse. We have implemented it in a 3 step process

## 14.1 Data Acquisition

Resumes of users working in different domains such as Software, Accountant, Marketing, etc are collected from worldwide web.

## 14.2 Data Transformation

The collected resume documents are available in various formats such as docx, pdf, etc. Each resume is loaded to the memory and the text in the document is transformed first by removing all the stopwords. As a second step, stemming is applied to every word. The transformed text is converted into a vector space model using the TF-IDF saved model.

KMeans model is used to cluster all the resumes. Clusters are formed using high intra similarity and low inter similarity. The resumes under a cluster are assigned one value to uniquely identify all the clusters in the group.

## 14.3 Data Load

The vector formed after transforming the text document is stored in a SQLite DB. The database contains information about the resume name, URL of the resume location, the vector model and the cluster number.

# 15 Conclusion

This project has provided a model that can rank resumes according to job description or keywords very efficiently, saving time for human resources to do it manually. It has automated the process. NLP techniques are used to convert unstructured resumes to structured term space models. The model is used in fetching the top 5 matching resumes using cosine similarity. KMeans clustering is used to retrieve similar resumes based on similar cluster numbers.

# Acknowledgment

We would like to thank our professor Chandrasekar Vuppalapati for giving us the opportunity to learn and do this project. We would also like to thank the dataset creator.

Dataset link: <https://drive.google.com/file/d/17M9oDPip5JFFFNJhDCBQKy8BMqoyxajU/view>

Credits: JAIJANYANI