Automatic Resume Segregation

Group Name: Group-3

Group Members:

First name	Last Name	Student number
Nithin	Tata	C0789951
Ramkiran	Sampathi	C0793773
Sai Charitha	Paleru	C0787250
Sree Rukmini	Thumu	C0795474

Submission date: ***Aug-14-2020***

Contents

Abstract **2**

Introduction **3**

Literature Review: 4

Intended Experiments 5

Dataset 5

System Design 6

Conclusion 11

REFERENCES 12

Abstract

Every year when a company posts about a job posting, thousands of people will apply. Due to this tremendous response, identifying the perfect resume is more time-consuming. The recruiter had to go through every information that is on the resume/ CV. Furthermore, they need to arrange them in detail on the excel sheet or the database so it would be easy to identify the correct person for the job postings. This process is tiresome, and often it is repetitive, and sometimes it also leads to errors. Also it is time taking and also an inefficient one.Aim is to build a system that evaluates resumes of any file format according to the given restrictions or the following demand provided by the client firm, using NLP (Natural Language Processing) and ML (Machine Learning) to rank them according to the given constraint. We will primarily receive the bulk of the input resumes from the client organization, which will also supply the requirements and restrictions by which our system should rate the resumes. We will read the candidate's social profiles (such as LinkedIn, Github, and others) in addition to the information provided by the resume, which will offer us with more actual information about that candidate.

Keywords— NLP, ML, MySQL, Elastic Search, Ranking, Scrapy, Scikit learn, SQL

Introduction

In the current system, candidates must fill out a manual form with all of their resume information, which takes a long time, and then they are dissatisfied with the position that the current system likes based on their qualifications. Let me explain through an example what a 5:1 ratio means: if five employees are hired, just one of them will be satisfied with his or her job. Let me give you an example: If I am a skilled python developer and a company hires me and requires me to work in Java, my python talents are rendered largely useless. On the other hand, if a position in a firm becomes open, the owner of the company will choose the best potential candidate for that position. As a result, our technology will serve as a handshake for these two entities. The firm that likes the best candidate possible and the candidate who prefers the best employment possible based on his or her skills and abilities.

The issue is that current systems are inflexible, inefficient, and time consuming. If a candidate is required to fill out forms online, it is possible that you will not receive accurate information on the applicant.As technology advances daily, we can use this technology to convert this tiresome and time-consuming task into a faster and more efficient one. Using the concepts of Natural Language Processing, Machine learning at the back end, and a friendly user interface, we can address the problems of every recruiter. In short, we can say that we have developed a user-friendly application that is easy to use and can identify the top resumes as per the recruiter's requirements.

Where our system saves the candidate's time by allowing them to upload their resume in any format they prefer, in addition to all of the information in the resume, our system will detect all of the candidate's activity from their social profile, resulting in the best candidate for that

particular job, and the candidate will be satisfied because he will be hired by that company, which is truly appreciative. On the other hand we are providing the same kind of flexibility to the client company.

Thorough research is done prior to starting the project. We have gathered the requirements and based on which we have built the UI. The user interface for the job applicant is provided with the functionality of uploading the resumes/ CVs, whereas for the recruiter/HR/ admin, they should post the job descriptions, view and sort the resumes as per their requirements. Additionally, testing is performed rigorously. The sole aim of the UI is to display the final result, which means displaying the top resume candidate names as per the recruiter's requirements.

Literature Review:

Hiring Systems-1:

The Hiring Team would use this system to post job openings and invite applicants. Newspapers, television, and word of mouth were all used as means of distribution. The candidates that are interested would next apply by emailing their resumes. The hiring staff received and sorted these resumes, and shortlisted candidates were contacted for additional rounds of interviews. Finding the ideal individual for their job duties would take a long time and a lot of human work.

Hiring Systems-2

As the industries have developed, so have their hiring requirements. Certain consulting units have been established to meet these employment needs. They proposed a system in which the candidate would upload and submit their information in a specific format to the agency. These agencies would then conduct keyword searches on the candidates. These agencies served as a link between the candidate and the company. These systems were inflexible since candidates had to upload their resumes in certain forms, which varied from system to system.

Hiring Systems-3

This is the approach we offer, which allows candidates to post resumes in a variety of formats. Our technology then analyses these resumes, indexes them, and stores them in a certain format. Our search becomes much easier as a result of this. The analyzing system employs a Natural Language Processing method, which is a subdomain of Artificial Intelligence. It reads resumes and converts them into a specified format by understanding the natural language/format used by the candidate. The knowledge base is where all of this new information is saved. The system updates the knowledge base with more information about the candidate obtained from his social profiles, such as Linkedin and Github.

Method

For this project we divided the entire work into two parts, part 1 being the backend with the complete model building and part 2 is building the frontend web interface to utilize the backend functionalities with ease. In the backend we implemented Machine learning models like Multinomial Naïve Bayes, Decision trees, Linear SVM, and XGBoost to classify the resumes into individual categories and then apply the parsing and ranking techniques on top of those resumes. Using the KNN machine learning algorithm in combination with Cosine similarity we are able to rank the resumes that are more related to the job postings.

In the frontend we built a web interface on top of server-less Google cloud platform for better availability, ease of use and low cost maintenance. Frontend serves an input layer where an employee/candidate can register themselves and apply for the job using their Google account as authentication. A candidate can apply for a job only once with his login and can apply for multiple different jobs. Admin or HR can post a job and can be able to see the applied candidates through the interface designed for the Admin users. Once the resumes are uploaded they will get stored in the Firebase database as document objects.

Intended Experiments

With the resume data we first parse the skills of the candidate using NLP.

- Apply Machine learning models like Multinomial Naïve bayes, Decision trees, Linear SVM, and XGBoost to classify the resumes into individual categories.
- Clean the Resumes and job description text data by removing stopwords, stemming, lemmatization, and removing numbers, special characters.
- Convert the resume and job description text into numerical vectors and apply KNN with cosine similarity to rank the resumes more similar to the job description.
- Finally we display the parsed data and ranked resumes in order on the HR web interface to shortlist the resumes.

Dataset

A. Capturing & Identifying Data

With the resume data we first parse the skills of the candidate using NLP.

For this project we collected data from multiple sources like Indeed, LinkedIn, iNeuron company. The total number of resumes we collected from all these sources is around 2500. We have collected different formats of resumes like pdf, word, txt files and using NLP, Regular expressions we converted them to normalized text data. All the resumes are collected using web scraping techniques. The dataset is completely filled with textual data with different formats in each resume and are taken from different candidates with various profiles, so the dataset is pretty diversified and good enough to train the model.

Following are the reasons why we chose iNeuron, Indeed, LinkedIn for this project:

- Sufficient number of resumes for creating robust and accurate model
- Good number skills in all the resumes
- Covers various textual representations of same words
- Covers people from various backgrounds and regions

System Design

The system architecture comprises two important phases with outer phase and the ranking phase. The outer phase will consist of modules like the client or the HR/client team, Social profile, and Candidates resume stored in the Firebase database. The client company is the one that will provide us with the majority of the resumes or C.V.s, as well as the precise requirements and constraints that should be used to rank them.

The resume ranking system consists of algorithms built using the KNN and Cosine similarity, and the resume parser system.

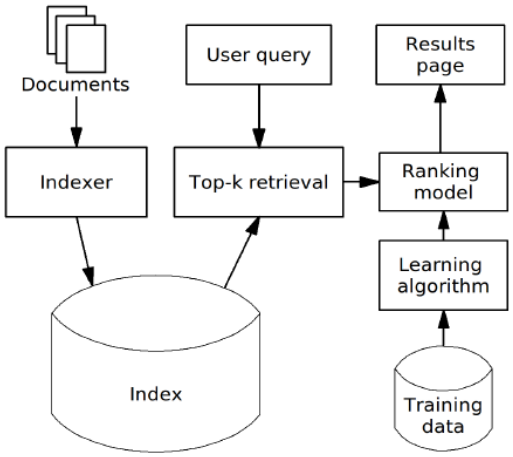

Figure1: Resume Ranking System Overview

Our project completely relies on two important input data files which are the resumes that are uploaded by candidates for applying for a job and the job description which is posted by the HR team. So our model will first parse both the resume and the job description and convert them to machine understandable format like numerical vectors. For parsing the resumes and job description files we are utilizing the Natural language processing. The applicant resumes and social profiles were parsed using NLP by the parsing system. That is without the need for any

manual intervention. This is how we're going to parse the resume one by one using Natural Language Processing.

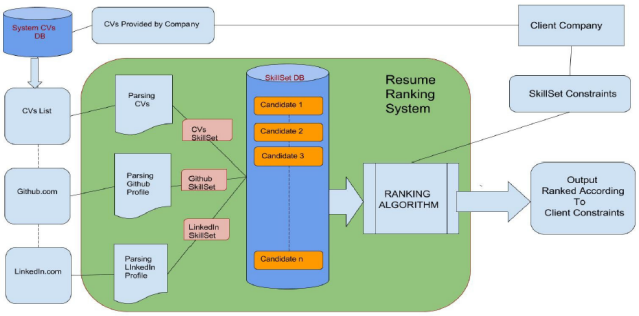

Figure2: Resume Ranking System Architecture

B. Cleaning & Structuring Data

The next step and the most time-consuming part would be structuring the data and cleaning it. Steps Followed are:

- Removal of stopwords using NLTK library
- Removal of special characters, numbers, and Non-ASCII characters
- Applying stemming to make similar words to same root
- Applying lemmatization to make related words into same common words
- Tokenizing and converting the text data into numerical vectors
- The aim of data preparation and cleaning is to create a final normalized data that can be used to achieve the goals of this project.

C. Building Model for resume classification

After having a proper dataset in place, we split the data into train and test with 80:20 percent ratios. We applied four algorithms namely Multinomial Naïve bayes, Decision trees, Linear SVM, and XGBoost. We used metrics like Accuracy score to evaluate the model performance to classify the resumes uploaded for particular job posts. First we checked if the accuracy scores of training and testing differ by huge margin to understand if the model is overfitting or working fine. Out of all the models that were used to classify we got better performance with Linear SVM and XGBoost with minute differences in Cross validation scores.

D. Cloud technologies used:

All the cloud services used in this project are provided by google and they all are free.

Firebase: It's a No-SQL database provided as part of the firebase project to store data in the form of documents and in JSON format. All the data created is assigned with a unique document id and it's used to do CRUD operations. It charges users based on read-write operations as our project aligns under limits; it costs nothing.

Firestore: Resumes uploaded by users need to be stored somewhere in the cloud where Firestore comes into picture. Firestore is a bucket storage service provided as part of the firebase project. Rules to access these objects are given in json files.

Google Cloud Console: its a on-demand computing service provided by google with various internal services like virtual machines, SCM, cloud functions, messages etc.

Google Cloud Functions: Our project was designed to work on a server-less environment where the code will be triggered only when a user requests it via HTTP. This sort of architecture is a bit slower compared to traditional server architecture when a server/ computing resource stays online for the whole life even if no one is using it.

a. We had basically 2 cloud functions:

Parser function: This function triggers when a user uploads the resume and extracts the required insights like name, experience, contact information, skills etc and stores it in the firebase as a json document.

Ranking function: This function holds all the machine learning logic to rank or score resumes by comparing it with the job descriptions.

These functions are configured on 2 gb ram and 5 gb disk space servers.

b. Challenges faced while using cloud:

Configuring serverless configurations is a complicated and tedious process because requirements need to be installed every time and maximum run time needs to be decided before deploying servers.

Angular is a bit out of scope for us but learning it is fun and interesting.

Results

A. Model Performance:

S.No	Model	Accuracy
1	Multinomial Naïve Bayes	57.57
2	Decision Tree	58.76
3	Linear SVM	63.84
4	XGBoost	64.98

B. Front-end results:

URL for admin: https://capestone-945f7.web.app/admin

URL for user login: https://capestone-945f7.web.app/jobs

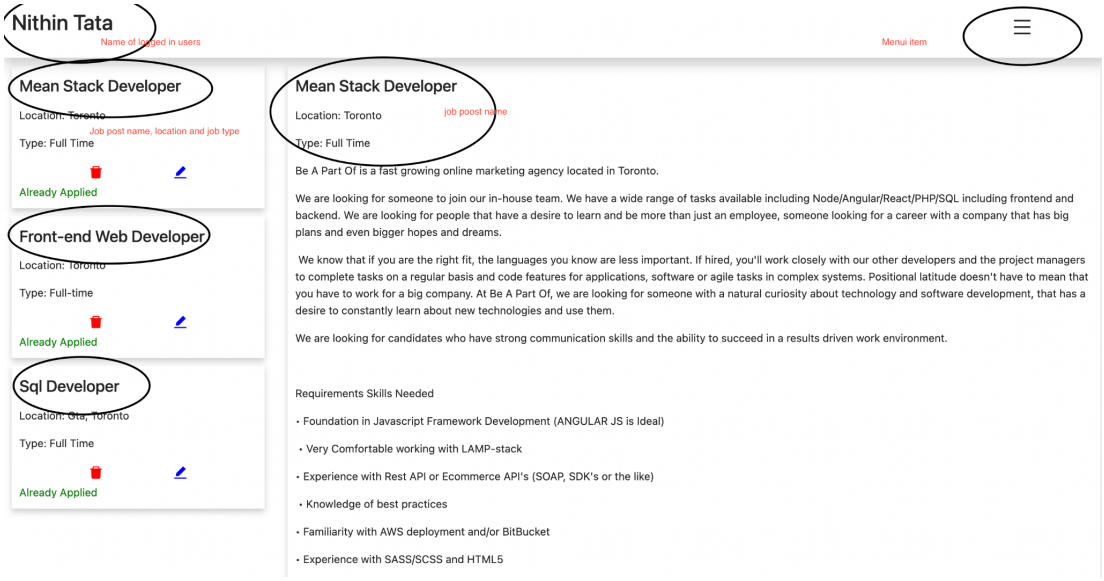

Fig: UI Screenshot for user to select job post and apply

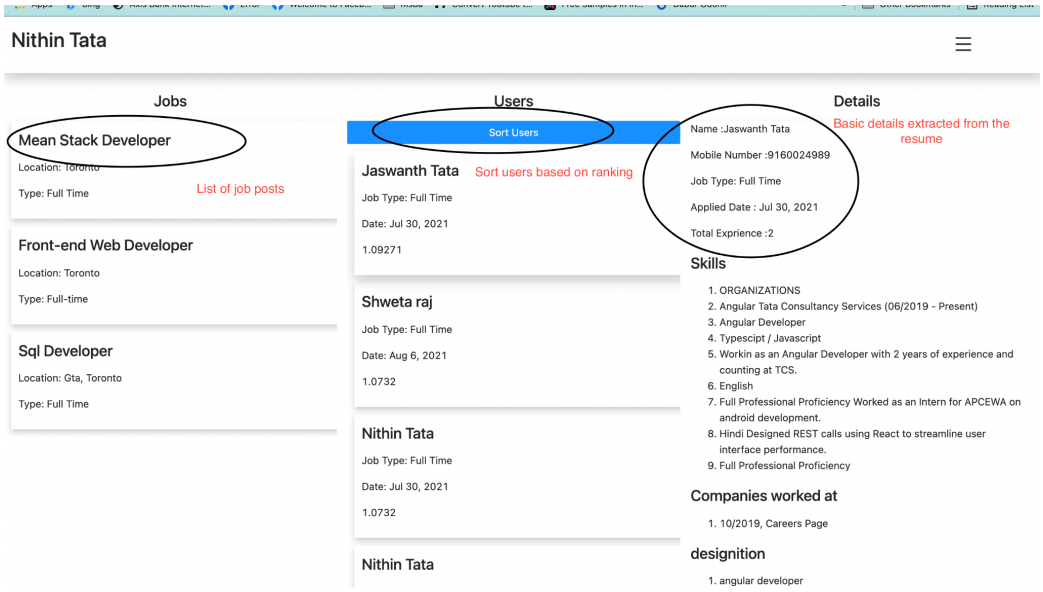

Fig: UI Screenshot for admin to select job post and view, sort users

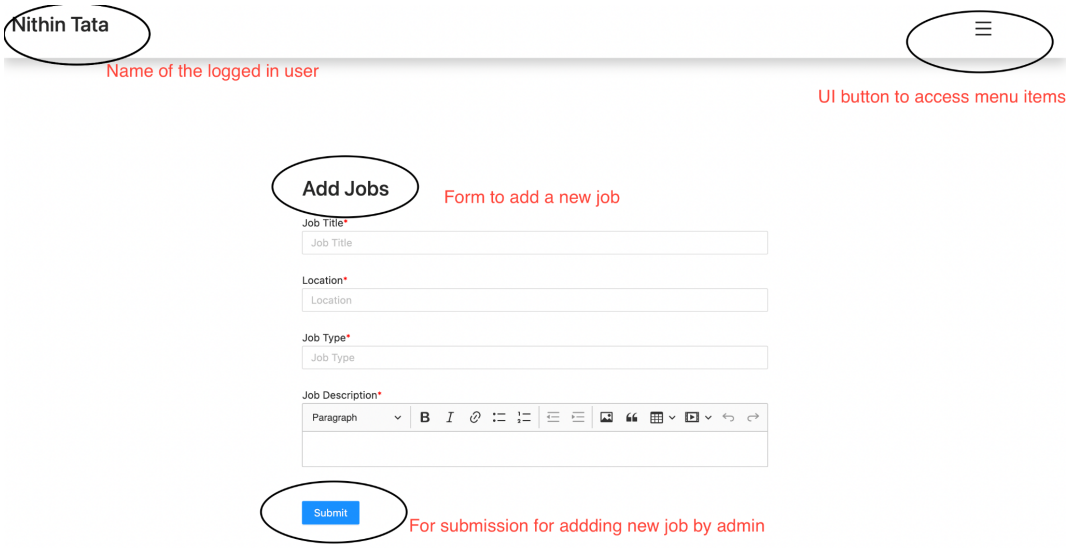

Fig: UI Screenshot for admin to add a new job

Conclusion

Our system will provide a better and efficient solution to the current hiring process. Accuracy can be improved if the data size is more to train the model since train and test data are not in Correlation. Linear SVM is taking more time compared to XGBoost which made us choose XGBoost over SVM for the UI integrationThis will provide potential candidates to the

organization and the candidate will be successfully placed in an organization which appreciates his/her skills and ability. The application can be extended further to other domains like Telecom, Healthcare, Ecommerce and public sector jobs.

REFERENCES

Xu, Shuo & Li, Yan & Zheng, Wang. (2017). Bayesian Multinomial Naïve Bayes Classifier to Text Classification. 347-352. 10.1007/978-981-10-5041-1_57

Abbas, Muhammad & Ali, Kamran & Memon, Saleem & Jamali, Abdul & Memon, Saleemullah & Ahmed, Anees. (2019). Multinomial Naive Bayes Classification Model for Sentiment Analysis. 10.13140/RG.2.2.30021.40169.

Malik, Shubham & Harode, Rohan & Kunwar, Akash. (2020). XGBoost: A Deep Dive into Boosting (Introduction Documentation). 10.13140/RG.2.2.15243.64803.

Jijo, Bahzad & Mohsin Abdulazeez, Adnan. (2021). Classification Based on Decision Tree Algorithm for Machine Learning. Journal of Applied Science and Technology Trends. 2. 20-28

Srivastava, Durgesh & Bhambhu, Lekha. (2010). Data classification using a support vector machine. Journal of Theoretical and Applied Information Technology. 12. 1-7

Evgeniou, Theodoros & Pontil, Massimiliano. (2011). Support Vector Machines: Theory and Applications. 2049. 249-257. 10.1007/3-540-44673-7_12

Gomaa, Wael & Fahmy, Aly. (2013). A Survey of Text Similarity Approaches. international journal of Computer Applications. 68. 10.5120/11638-7118

Prasetya, Didik & Wibawa, Aji & Hirashima, Tsukasa. (2018). The performance of text similarity algorithms. International Journal of Advances in Intelligent Informatics. 4. 10.26555/ijain.v4i1.152

Maake, Benard & Zuva, Tranos. (2018). A Comparative Analysis of Text Similarity Measures and Algorithms in Research Paper Recommender Systems. 10.1109/ICTAS.2018.8368766

Wang, Lishan. (2019). Research and Implementation of Machine Learning Classifiers Based on KNN. IOP Conference Series: Materials Science and Engineering. 677. 052038. 10.1088/1757-899X/677/5/052038

Contributions

Nithin Tata (C0789951):

- Discussing and designing the Flow Chart in architecture design.
- Setting up and configuring google cloud functions which will be triggered when a new user uploads the resume
- Creating backend API using REST protocol.
- Firebase configuration to add documents using REST API from user interface.
- Selecting and researching Cloud providers to host our webapp.
- Extracting text from PDF/DOC files using doc files, where this text is processed by NLP techniques.
- Collection creation in firebase for user and admin to do CRUD operations
- Designing User and admin panel using figma. This acts as a prototype to build using code.
- Creating and writing angular related code for user and admin GUI.
- UI testing to verify uploading process
- Creation of git repo and source code management using GIT on Github
- Contribution in creation of final presentation and report.

Sree Rukmini Thumu (C0795474):

Following are the tasks accomplished by me in the project

- Planning and designing workflow
    - produced the communication plan for the project
    - system architecture design through flow chart is given
    - Database designing -firestore
    - selecting the cloud provider to host app-firebase
- For identifying the dataset I contributed along with my other team mates.
- Contributed for the software requirements analysis.
- Preprocessing of the dataset is done through the following steps along with other teammates
    - Extracting text from PDF/DOC files
    - Extracting required details from resume according to the job requirements
    - Performing NLP operations
    - Text Preprocessing
    - Converting word to Vector
    - Chart Updates for visualization
- Contributed for User Interface development along with Nithin
    - Developed in angular 9
    - Two different components are built one for user and other for the admin
    - For job posting the privilege is given to the admins only. Also edit, delete and view facilities are given.
    - For the user to upload a resume option was given.
    - For admin/recruiter privilege of viewing the best match resumes is given.
- Handling the response (resumes) from UI is done in json format.
- Measured the efficiency of the model to filter out the best among the other resumes.
- Final unit test and integration test is done thoroughly to identify bugs.
- In the process of testing, I have identified a few bugs and as a team we made a decision in solving them.
- Assisted in deploying the code to google firebase
- I have contributed myself in preparing the presentation slides and writing the report.

Sai Charitha Paleru (C0787250):

- Researched on different databases and finalized Firebase's Firestore service to use as a database and hosting service,storage service in Firebase to host websites and store resumes.
- Performed text preprocessing and vectorization on resume content to do semantic analysis on the vectors using cosine similarity.
- Created and designed a schema for admin and users to store files in google firestore.
- Discussed with the team and finalised the machine learning model by comparing various performance metrics.
- Managing and connecting angular routes between pages on web app/GUI.
- Performed performance testing of API calls using postman.
- Testing and finding the optimal server configuration to run on demand functions in google cloud.
- Contributed myself in doing powerpoint presentations and final reports.

Ramkiran Sampathi(C0793773):

- Data Collection: Collected data from multiple resources by directly contacting the company iNeuron for resume data and also used platforms like LinkedIn and Indeed to scrape data
- Extracting required details from resume by parsing them using python libraries according to the job requirements
- Chart Updates: Gantt chart updates are monitored and updated on weekly basis to track tasks done by every teammate for project completion
- Training and Testing the clustering models: All the Machine learning models are trained and tested with resume data for classification
- Hyper parameter tuning of the Machine learning models for better performance and classification of the uploaded resumes
- Implemented the KNN and Cosine similarity algorithm for obtaining Semantic and Syntactic similarity between the uploaded resumes in the firestore bucket and the job description
- Verifying model performance and accuracy by going through each sorted resume manually
- Communication plan: Prepared a proper team communication plan and acted as a scrum master to discuss about weekly deliverables