Bidirectional Job & Resume Matching System

—Practice Module of Intelligent Reasoning Systems

Group Leader： LI YURUI

Group Members:

LI FANGQING

PRADEEP KUMAR ARUMUGAM

Oct 2022

Abstract

In the current economic environment where there is a huge demand for talent, the problems of how job seekers find their desirable job efficiently, and how enterprises choose the right candidates they need, have become a social issue that needs to be further addressed. The existing diverse platforms, such as LinkedIn and JobStreet, do provide a variety of information, but it is always difficult for users to accurately find the jobs or candidates they want or match their requirements from such a large amount of information, which leads to a low efficiency for job search and recruitment.

In this project, a Bidirectional Job & Resume Matching System is proposed which will provide a platform for both HR managers and job seekers to upload their profiles, make a quick search, and get the recommendation results of best matching candidates and jobs. Doc2Vec model and Term Frequency-inverse Document Frequency (TF-IDF) model were selected and trained to process the data and achieve the function of accurately matching by extracting the features from information from the upload files. A website based on the Flask framework was also built to provide a simple interactive interface. It is hoped that through this streamlined interface, the unnecessary information and steps in the job search and recruitment process can be reduced.

This report will introduce and analyse the project through 6 sections, which are the Business Problem Background, the Market Research, the Problem Description, the Project Solution, the Project Implementation and the Project Performance & Validation to present the project concept and project results.

Keywords: Bidirectional Recommendations, Jobs, Resumes, Doc2Vec, TF-IDF

Contents

1. Business Problem Background

1.1 Background

With the growing economy, the demand for talent has increased to meet the rapid changes in the market and to cope with the rapid employment changes.[1]

Take Singapore's Labor Market Statistics as an example, as shown in Figure 1, according to the employment data of Singapore provided by CEIC, since January 2019, there has been a significant increase in the overall employee population, which means that the demand for jobs also shows a trend of increasing.

Figure 1. The employment data of Singapore from January 2021 to June 2022.[2]

What increased the demand for job search and recruitment is the appearance of various job search and recruitment websites, which provides a large database for the research of recruitment. In the meanwhile, the request for a new generation of recommendation systems for service personalization and precision requirements is becoming increasingly strong.

1.2 Requirements Analysis

To find out the breakthrough points of a good recommendation system, some analyses are made to discuss the optimized requirements of recommendation systems from the perspective of job seekers, human resources (HR) managers and the operator of the recommendation system.

From the point of view of job seekers, a more intelligent and personalized resume recommendation system can guide them to complete a faster and more accurate resume that fully expresses their abilities and strengths. More importantly, it is also more efficient for a job seeker if the recommendation system can be able to target and recommend the interested jobs more precisely from a long list of jobs according to their characteristics and preferences. In addition, the system should provide easier and more user-friendly services to enhance the user experience.

From the perspective of an enterprise's HR manager, the recommendation system is expected to be able to quickly filter the most suitable collection of candidates from a large number of resumes and provide as much detailed information about the applicant as possible to evaluate the candidate in all aspects.[3] Through the set evaluation system of the recruitment website, the internal HR management system of the company can be connected with this information as well, which can also provide a reference for decision-making on the system related to talent pooling, training and appraisal to meet the development strategies of the company.

For the operator of the recommendation system, the optimization of the systems should be able to improve the success rate of the system job search and recruitment docking, strengthen the humanized service of the system, enhance the user experience, attract different kinds of users and aggregating users is the basic requirement. In addition, it is also the responsibility of a good recommendation system that the system can provide the current related data, such as the release of annual statistical reports on applicants and the forecast reports of applicants. This visualization of data should make it able to show an overview of labour market supply and demand in terms of industry, region and time dimensions, through which the enterprises and government economic planning can get convincible references, win a reputation for system operation and generate more socio-economic benefits in the further development.[4]

Starting with the above requirements, a bidirectional job recommendation and resumes screening system is decided to be designed and built to provide a personalized and convenient platform for both job seekers and HR managers

2. Market Research

2.1 Analysis of Existing Platforms

To meet the market demand, various job search and recruitment websites have emerged, and their widespread utilization provides a perfect database for market research.

To find the breakthrough, point for this project, some mainstream job search and recruitment platforms in the market have been tested and researched by our group. Currently, popular job boards and recruitment websites can be broadly classified into five categories according to the technology they use.[5-6] As shown in Figure 2, the type of websites contains traditional information portal websites, professional social websites, recommendation websites, typical industry websites and manual assistance websites. Traditional information portal websites can be divided into specialist portal websites and general portal websites, which are based on static information display and publishing. However, traditional information portal websites are facing technical transformation challenges due to the lack of personalized services.

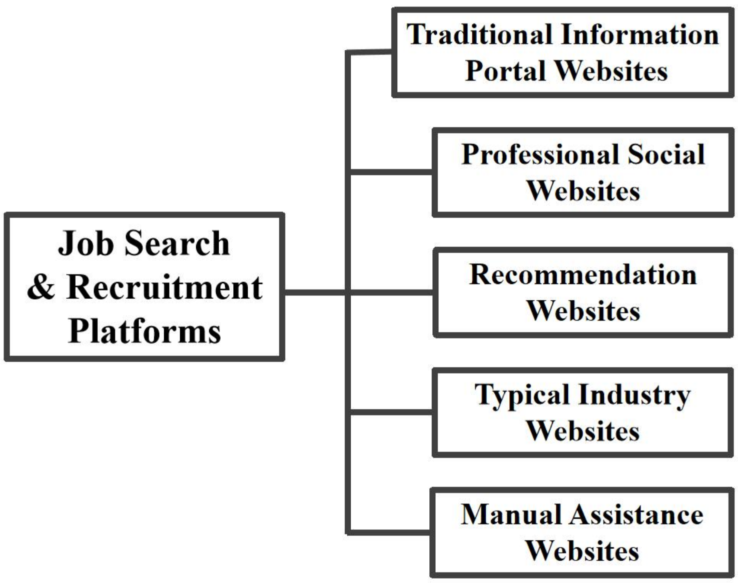


Figure 2. The categories of current platforms.

Compared with traditional ones, professional social websites are relatively more flexible. Take LinkedIn as a typical example, which builds exclusive professional contacts for each user, the users can contact interesting positions or job candidates through these contacts and then personalized recommendations will be made to match the information. Unlike the first three types of websites, typical industry websites are more specialized recruitment platforms, they focus on specific job opportunities, making full use of industry features and knowledge to filter candidates and make job recommendations, such as Lagou in the internet industry. Manual assistance websites are recruitment websites that always provide information sites that work with real humans, such as the Blind. Regardless of the type of job search and recruitment platforms, intelligent and efficient recommendation methods always seem to be the most important and worthwhile part of the research.[7]

2.2 Development Perspectives

After the above research, our team believes that the recommendation system can be optimized from the following perspectives.

1. Build personalized resume templates and job templates for different industry sectors.
2. Extracting and modelling user characteristics based on multi-dimensional heterogeneous data and operation behaviour within the system and interactions between users.
3. Bidirectional recommendation algorithm based on matching job preferences of job seekers and recruitment preferences of enterprises.
4. Personalized and humanized information collection strategy, data statistics, and multiple data visualization displays.

2.3 Literature Review

Before deciding on the detailed research direction and proceeding with the work, our team looked through several established papers, patents, and projects.

The bidirectional recommendation was first proposed by Pizzato L et al in 2010.[8] The bidirectional recommendation is a unique branch of personalized recommendation systems, and the algorithm applies to systems where there are preferences between the two users of the recommendation.[9] It considers the mutual preference relationship between the two users of the recommendation so that both users can achieve their purpose efficiently, and the accuracy of the recommendation system can also be improved. Malinowski J et al built a job recommendation system for individual users and corporate them based on the preference information that considered the recommendation results of both users.[10] Yu H et al proposed a similarity calculation method that combined the explicit and implicit preference information of both users and further explored the correlation between the preference degree. [11]

Considering the objective factors such as project duration, the construction of bidirectional recommendation systems and the optimization of data visualization will be focused on in this project for implementation and research. A SWOT analysis is developed to evaluate the feasibility and scope of circulation in the market, and the result is shown in Figure 3.

Figure 3. The SWOT Analysis

3. Problem Description

3.1 Project Objectives

Based on the above market analysis, a Bidirectional Job and Resume Matching System is decided to be designed and built in our project. The system should allow both HR managers and job seekers to upload their profiles, and when an HR manager screening the candidates or a job seeker trying to find the appropriate job, the system can provide an accurate matching result, which would offer a convenient and efficient environment to achieve the recruitment in a primary step.

3.2 Project Scope

This project is based on a content-based recommendation algorithm that matches the job requirements provided by the HR manager with the resumes uploaded by job seekers. The most critical aspect of the content-based matching approach is the formulation of matching rules, such as the "job requirements" attribute of the job and the "skills" attribute from the resumes, which are crucial to the accuracy of the matching. The majority of these attributes are long text, and it is still challenging to make full use of the deep semantics of long text items for feature matching. With the development of natural language processing technology, the vectorized representation of long text provides technical support for deep semantic mining. Therefore, this project will combine deep semantic features to build a more accurate Job and Resume Matching System to make full use of the information. The rich semantic information contained in the long text in the features will enable accurate matching between resumes and job information.

As a bidirectional system, the whole system is expected to be divided into two separate systems for both HR Managers to upload job information and job seekers to upload their resumes.

Through researching various online job boards, job seekers, always want to know the job information posted by the recruiters, while the recruiters are concerned with the basic information of the job seekers and their job search intention. Therefore, once the job information content and resumes are uploaded, the job features and job seeker features needed to be constructed based on the information provided in the uploaded file, so that further matching can be done, and high-matching recommendations can be displayed after the matching is completed.

3.2 Success Measurements

1. The HR managers can upload the job information.

2. The job seekers can upload their resumes.

3. Extract the features from the contents of job information and resumes and achieve bidirectional matching and recommendation.

3. Quick search for both HR managers and job seekers through keywords.

4. Successfully show the recommendation results of resumes or job information

4. Project Solution

In this project, a bidirectional recommendation website for resumes and jobs is planned to be implied. To achieve a lightweight model and to speed up the response time on the back end of the site, both recommendation functions are decided to be achieved by the same model.

Through this model, job seekers will be able to find the best match according to the information provided in their resumes and the recruiters will be able to find the employees among the applicants by uploading job descriptions.

As the resumes and job descriptions consist of a large amount of text, to implement the recommendation system and achieve the function of bidirectional recommendations, the features from the text need to be extracted and the similarity of the features needs to be compared to match the job with the information of the resumes. The primary system design is shown in Figure 4 below.

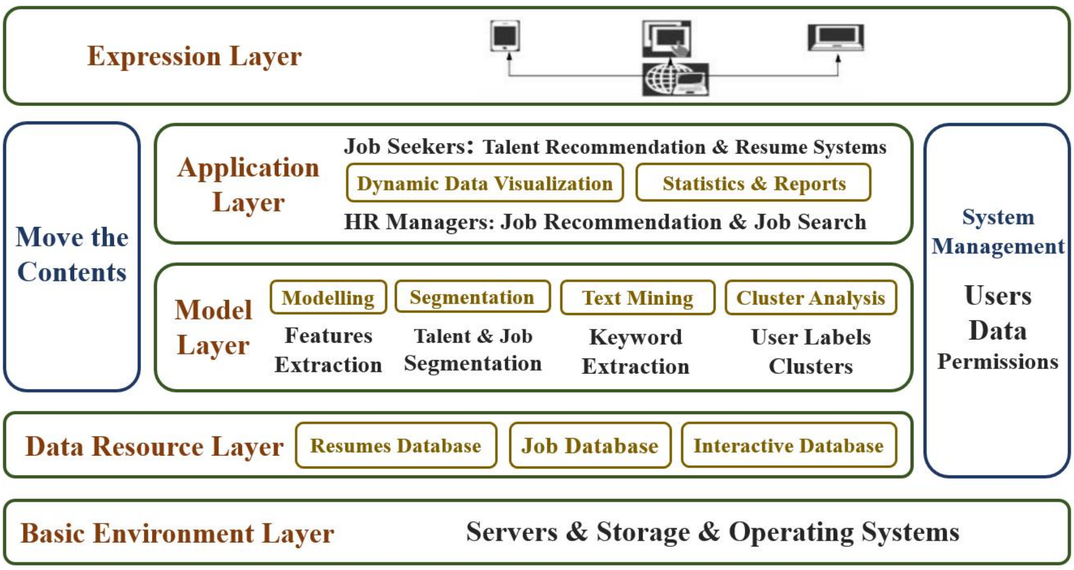


Figure 4. The design framework of system architecture.

Through preliminary model research, two models that can meet our needs for extracting vector features from the text are selected, which are the Term Frequency-Inverse Document Frequency (TF-IDF) Model and the Doc2Vec Model.

4.1 Term Frequency-inverse Document Frequency (TF-IDF) Model

Machine learning with natural language is faced with one major hurdle is the algorithms usually deal with numbers, while the natural language is text. It is necessary to do text vectorization, which is a fundamental step in the process of machine learning for analyzing data, and different vectorization algorithms will drastically affect the final results.

The Term Frequency-Inverse Document Frequency (TF-IDF) Model is a common weighting technique used in information retrieval and data mining. It gives a statistical measure that evaluates how relevant a word is to a document in a collection of documents. The algorithm is simple and efficient, which is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents. The most important use of it is automated text analysis which is very useful for scoring words in machine learning algorithms for Natural Language Processing (NLP). TF-ID model provides a way to associate each word in a document with a number that represents how relevant each word is in that document, and documents with similar or relevant words will have similar vectors. Once the words are transformed into numbers in a way that machine learning algorithms can understand, the TF-IDF score can be fed to algorithms such as Naive Bayes and Support Vector Machines, which greatly improves the results of more basic methods like word counts.

The TF-IDF Model can be considered into two components, one is "Term Frequency" (TF) and the other is "Inverse Document Frequency" (IDF).

TF can count the deactivated words and filter them. Once the high-frequency words have been filtered, the words with real meaning are remained to be considered. There are several ways of calculating this frequency, with the simplest being a raw count of instances a word appears in a document. Then, there are ways to adjust the frequency, by the length of a document, or by the raw frequency of the most frequent word in a document.

The IDF of the word across a set of documents can conduct how common or rare a word is in the entire document set. The extent of the number being close to zero will illustrate how common a word is. This metric can be calculated by taking the total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm.

Once the values of TF and IDF have been obtained, the TF-IDF of a word can be calculated by multiplying these two values together. The larger the TF-IDF value of a word, the more important that word will generally be. By calculating the TF-IDF of each word in an article and sorting them from largest to smallest, the words at the top of the list can be defined as the keywords of that article.

In more formal mathematical terms, the TF-IDF score can be calculated by formula 2-1, 2-2, and 2-3.

$$tf\,idf\,(t, d, D) = tf\,(t, d)\ .\ idf\,(t, D) \tag{2-1}$$

$$tf\,(t, d) = log\,(1 + freq\,(t, d)) \tag{2-2}$$

$$idf\,(t, D) = log\,\left(\frac{N}{count\,(d \in D{:}t \in d)}\right) \tag{2-3}$$

where,

t: The target word.

d: The document that the word t belongs to.

D: The document set that document d belongs to.

N: The number of documents under document set D.

4.2 Doc2Vec

Doc2Vec is a technique based on Word2vec which was proposed in 2014 by Mikilov and Le. Word2vec uses neural networks to establish word embeddings and attempt to decide the importance of a word by breaking down its neighbouring words from the context and thus resolving the context loss issue. Word embedding is a popular method for representing words as vectors. It is suitable for catching the context of a given word, finding the likeness between the semantics and syntactic, or extracting the connection with different words.

The two major architectures for Word2vec are the Continuous Bag-Of-Words (CBOW) and the Skip-Gram (SG).[12]

SG model iterates over the words in the corpus and predicts the context. Figure 5 shows the architecture of the SG model, the current word is used as the input layer, and the context words are present in the output.

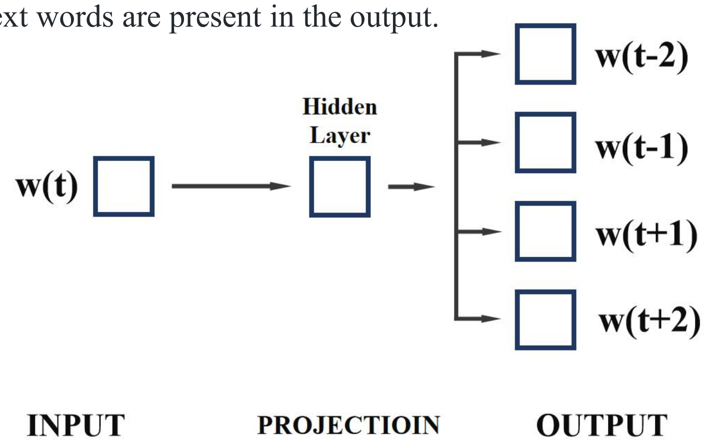


Figure 5. The architecture of Skip-Gram model.

The CBOW model uses the context to predict the current word. Figure 6 shows the architecture of the CBOW model, the context words are present in the input layer, and the current word is present in the output layer.

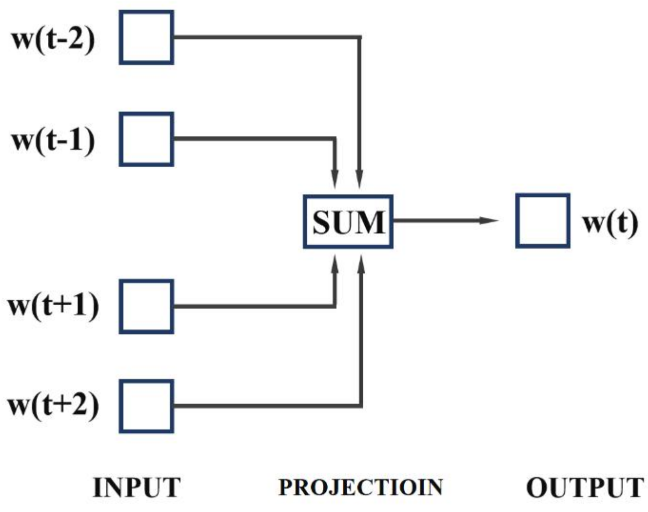


Figure 6. The architecture of CBOW model.

In these two architectures, the number of dimensions used to depict the current word is presented in the hidden layer. Both CBOW and SG models are shallow neural networks, which means they only contain one hidden layer. In general, the essential thought of word embedding is that the words that occur in a comparative context will be nearer to one another in the vector space.

The initial motivation behind building Doc2vec was the unstructured nature of documents as compared with individual words. Doc2vec uses an unsupervised learning approach to better understand documents as a whole. Compared with Word2vec , the architectures have an additional parameter called paragraph_id, which is also known as a paragraph vector. It is added to portray missing data from a document's context.

For Doc2vec, there are also 2 architectures which are PV-DM and PV-DBOW, respectively.

4.2.1 Distributed Memory Version of Paragraph Vector (PV-DM)

The architecture of PV-DM is similar to that of the CBOW model in Word2vec, the schematic diagram of which is shown in Figure 7. [13]

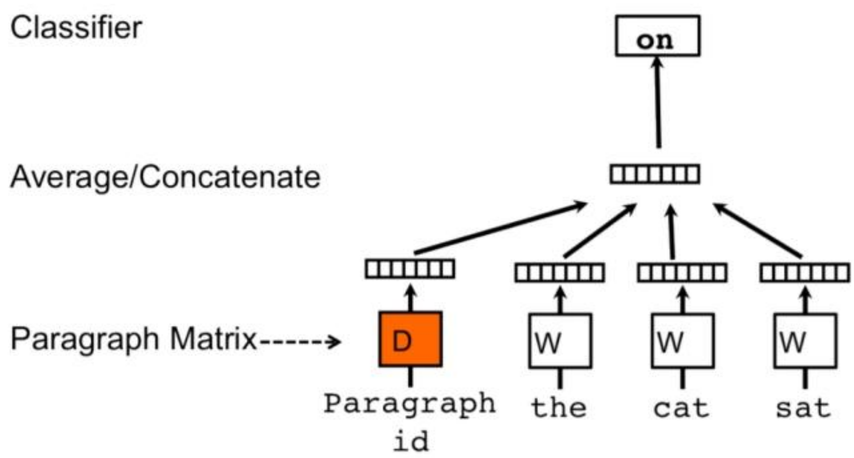


Figure 7. The architecture of PV-DM model.

The architecture of PV-DM contains three parts. The first part is the paragraph matrix, which consists of the vector columns for a paragraph. Secondly the Average/Concatenate part signifies whether the model would take the average or concatenate the paragraph and the word vectors. Finally, the Classifier part will take the input from the last hidden layer and tries to forecast the center word.

Compared with the architecture of CBOW, the addition of the parargaph_id and word vectors are arbitrarily initialized. Each paragraph_id is relegated to a single record while word vectors are shared among all records. Hence, paragraph_id is unique for each document. Thus, the document vector D is also trained while training the word vectors W. Eventually, document vector D will contain the numeric representation of the record.

It seems that a PV-DM model serves as a memory that recalls what's absent from the current context. Through PV-DM, consecutive words are randomly sampled from a paragraph, and the model then tries to predict a center word from the randomly sampled set of words, considering the input, which are the context words and a paragraph_id.

4.2.2 Distributed Bag of Words Version of Paragraph Vector (PV-DBOW)

The architecture of PV-DBOW utilizes a paragraph vector to classify all the words in the record rather than forecasting the next words, and the schematic diagram of it is shown in Figure 8.[13]Compared with the SG model, DBOW utilizes the paragraph_id and predicts randomly sampled words from the record. While training, a list of words will be sampled and then a classifier will be formed to classify whether a word belongs to the document so that word vectors can be learned. The model disregards the context words in the input but forces the model to forecast words arbitrarily sampled from the paragraph in the output.

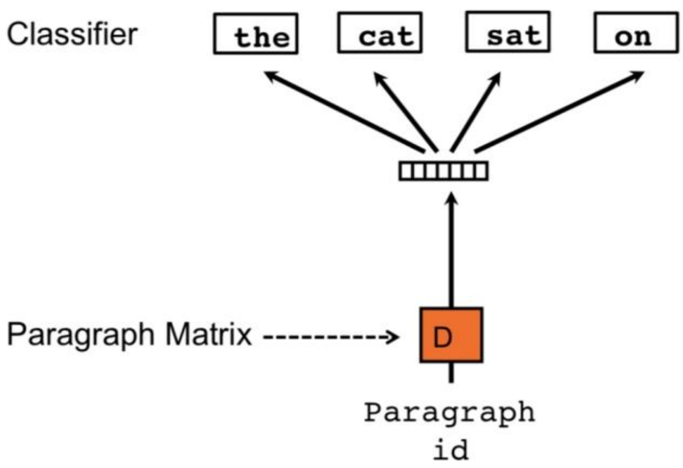


Figure 8. The architecture of PV-DBOW model.

4.3 Conclusion

By analyzing the properties and characteristics of TF-IDF and Doc2vec, the Doc2vec model is finally selected for text vector generation. The reasons are shown as follows.

TF-IDF model is simple, fast, and easy to understand, but sometimes it is not comprehensive enough to use word frequency to measure the importance of a word in an article, and the appearance of important words may not be frequent enough. In that case, the calculation results that are used to reflect the location information or the importance of words in context may not be accurate and convincing. So it may be a better choice to use the Doc2vec model to reflect the contextual structure of words and achieve their contextual relationship in a paragraph.

Secondly, the dimensionality of the vector results generated by TF-IDF depends on the number of keywords in the text. If TF-IDF is used to get the overall text content of the resume documents for vector generation, the dimensionality of the generated vectors will have more than several thousand dimensions, resulting in an explosion of dimensions, and the amount of computation may be too large in the next calculation of vector similarity

Thirdly, the dimension of feature vectors generated by TF-IDF is not fixed. To achieve the function of bidirectional recommendation and vector similarity calculation, the job text-generated feature vectors and the vectors of the resumes always need to keep in the same dimension. By Doc2ve, the dimensionality of the generated vectors can be controlled, and the subsequent matching can be facilitated.

5. Project Implementation

5.1 Job Dataset and Preprocess

5.1.1 Pre-Processing

The dataset selected in this project is a pre-crawled dataset, taken as a subset of a bigger dataset containing more than 9.4 million job listings, and it was created by extracting data from a leading job board called Naukri.com. As shown in Figure x, the original job dataset has 21996 rows and 13 columns of data, including company, education, experience, job description and other information.

company	education	experience	industry	jobdescription	jobid	joblocation_address	jobtitle	numberofpositions	payrate	postdate	site_name	skills
MM Media Pvt Ltd	UG: B.Tech/B.E. - Any Specialization PG:Any Po...	0 - 1 yrs	Media / Entertainment / Internet	Job Description Send me Jobs like this Quali...	210516002263	Chennai	Walkin Data Entry Operator (night Shift)	NaN	1,50,000 - 2,25,000 P.A	2016-05-21 19:30:00 +0000	NaN	ITES
find live infotech	UG: B.Tech/B.E. - Any Specialization PG:MBA/PG...	0 - 0 yrs	Advertising / PR / MR / Event Management	Job Description Send me Jobs like this Quali...	210516002391	Chennai	Work Based Onhome Based Part Time.	60.0	1,50,000 - 2,50,000 P.A. 20000	2016-05-21 19:30:00 +0000	NaN	Marketing
Softtech Career Infosystem Pvt. Ltd	UG: Any Graduate - Any Specialization PG:Any P...	4 - 8 yrs	IT-Software / Software Services	Job Description Send me Jobs like this - as ...	101016900534	Bengaluru	Pl/sql Developer - SQL	NaN	Not Disclosed by Recruiter	2016-10-13 16:20:55 +0000	NaN	IT Software - Application Programming
Onboard HRServices LLP	UG: Any Graduate - Any Specialization PG:CA Do...	11 - 15 yrs	Banking / Financial Services / Broking	Job Description Send me Jobs like this - Inv...	81016900536	Mumbai, Bengaluru, Kolkata, Chennai, Coimbator...	Manager/ad/partner - Indirect Tax - CA	NaN	Not Disclosed by Recruiter	2016-10-13 16:20:55 +0000	NaN	Accounts
Spire Technologies and Solutions Pvt. Ltd.	UG: B.Tech/B.E. - Any Specialization PG:Any Po...	6 - 8 yrs	IT-Software / Software Services	Job Description Send me Jobs like this Pleas...	120916002122	Bengaluru	JAVA Technical Lead (6-8 yrs) -	4.0	Not Disclosed by Recruiter	2016-10-13 16:20:55 +0000	NaN	IT Software - Application Programming

After removing the useless columns of data, only the remaining ten columns of data was processed.

company	education	experience	industry	jobdescription	joblocation_address	jobtitle	numberofpositions	postdate	skills
MM Media Pvt Ltd	UG: B.Tech/B.E. - Any Specialization PG:Any Po...	0 - 1 yrs	Media / Entertainment / Internet	Job Description Send me Jobs like this Quali...	Chennai	Walkin Data Entry Operator (night Shift)	NaN	2016-05-21 19:30:00 +0000	ITES
find live infotech	UG: B.Tech/B.E. - Any Specialization PG:MBA/PG...	0 - 0 yrs	Advertising / PR / MR / Event Management	Job Description Send me Jobs like this Quali...	Chennai	Work Based Onhome Based Part Time.	60.0	2016-05-21 19:30:00 +0000	Marketing
Softtech Career Infosystem Pvt. Ltd	UG: Any Graduate - Any Specialization PG:Any P...	4 - 8 yrs	IT-Software / Software Services	Job Description Send me Jobs like this - as ...	Bengaluru	Pl/sql Developer - SQL	NaN	2016-10-13 16:20:55 +0000	IT Software - Application Programming
Onboard HRServices LLP	UG: Any Graduate - Any Specialization PG:CA Do...	11 - 15 yrs	Banking / Financial Services / Broking	Job Description Send me Jobs like this - Inv...	Mumbai, Bengaluru, Kolkata, Chennai, Coimbator...	Manager/ad/partner - Indirect Tax - CA	NaN	2016-10-13 16:20:55 +0000	Accounts
Spire Technologies and Solutions Pvt. Ltd.	UG: B.Tech/B.E. - Any Specialization PG:Any Po...	6 - 8 yrs	IT-Software / Software Services	Job Description Send me Jobs like this Pleas...	Bengaluru	JAVA Technical Lead (6-8 yrs) -	4.0	2016-10-13 16:20:55 +0000	IT Software - Application Programming

For skills and education, where there is a lot of NA data, the function of sklearn.impute.SimpleImpute was utilized to fill each NA data position in the dataset with the most frequent value.

```
from sklearn.impute import SimpleImputer
to_fill = ['education', 'skills']
for col in to_fill:
    imputer = SimpleImputer(strategy='most_frequent')
    job[col] = imputer.fit_transform(job[[col]])
```

As the same qualification is represented differently in the education column, the replace function was used to unify the representation of the qualification. The replacement format code is as follows.

```
job['Education'] = job['education'].str.split(' ')
job['Education'] = job['Education'].apply(lambda x: x[1] if len(x) > 1 else x[0])

job['Education'] = job['Education'].replace(('B.Tech/B.E.','Graduation','Other','-','Not','B.Tech/B.E.,','Postgraduate',
                                              'PG:CA','Diploma,','B.Com,','B.Pharma,','B.A,','BCA,','B.Sc,','MBA/PGDM','B.B.A,',
                                             'PG:Other','Doctorate:Doctorate','Post'),
                                             ('B.Tech','B.Tech','B.Tech','B.Tech','B.Tech','B.Tech','B.Tech',
                                             'CA','Diploma','B.Com','B.Pharma','B.A','BCA','B.Sc','MBA','BBA',
                                             'B.Tech','Doctorate','B.Tech'))
```

The same problem exists for the job_address, the following replacement was utilized to unify the location expression.

```
replacements = {
    'joblocation_address': {
        r'(Bengaluru/Bangalore)': 'Bangalore',
        r'Bengaluru': 'Bangalore',
        r'Hyderabad / Secunderabad': 'Hyderabad',
        r'Mumbai , Mumbai': 'Mumbai',
        r'Noida': 'NCR',
        r'Delhi': 'NCR',
        r'Gurgaon': 'NCR',
        r'Delhi/NCR(National Capital Region)': 'NCR',
        r'Delhi , Delhi': 'NCR',
        r'Noida , Noida/Greater Noida': 'NCR',
        r'Ghaziabad': 'NCR',
        r'Delhi/NCR(National Capital Region) , Gurgaon': 'NCR',
        r'NCR , NCR': 'NCR',
        r'NCR/NCR(National Capital Region)': 'NCR',
        r'NCR , NCR/Greater NCR': 'NCR',
        r'NCR/NCR(National Capital Region) , NCR': 'NCR',
        r'NCR , NCR/NCR(National Capital Region)': 'NCR',
        r'Bangalore , Bangalore / Bangalore': 'Bangalore',
        r'Bangalore , karnataka': 'Bangalore',
        r'NCR/NCR(National Capital Region)': 'NCR',
        r'NCR/Greater NCR': 'NCR',
        r'NCR/NCR(National Capital Region) , NCR': 'NCR'

    }
}
```

For job description, as it is an important variable that needs to be matched later, all data where the job description was empty is removed directly.

For industry, there is only one NA data, so the largest number of 'IT-Software / Software Services' values were used to fill in the data and eliminate the NA data.

All other data attributes remain the same and the final processed data set is stored as job_eda.csv.

5.1.2 Visual Analysis

The visual analysis results of the database data are shown in the figures below.

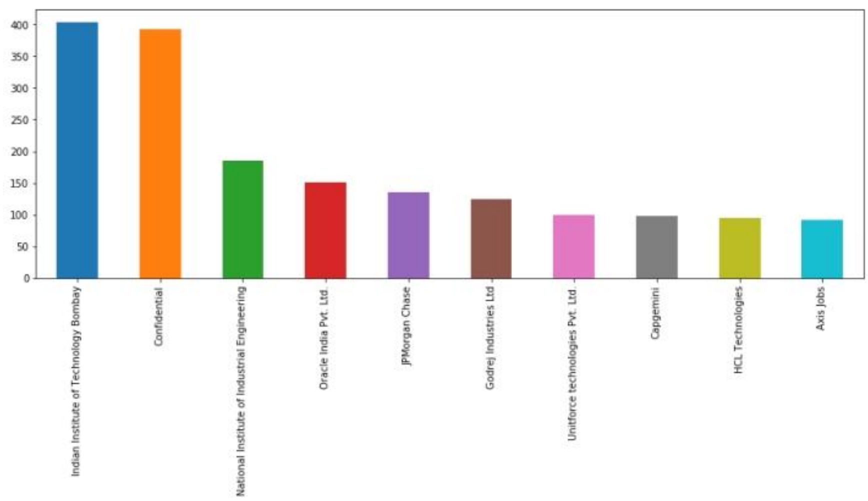


Figure 9. The bar chart of the Top 10 Companies from the dataset.

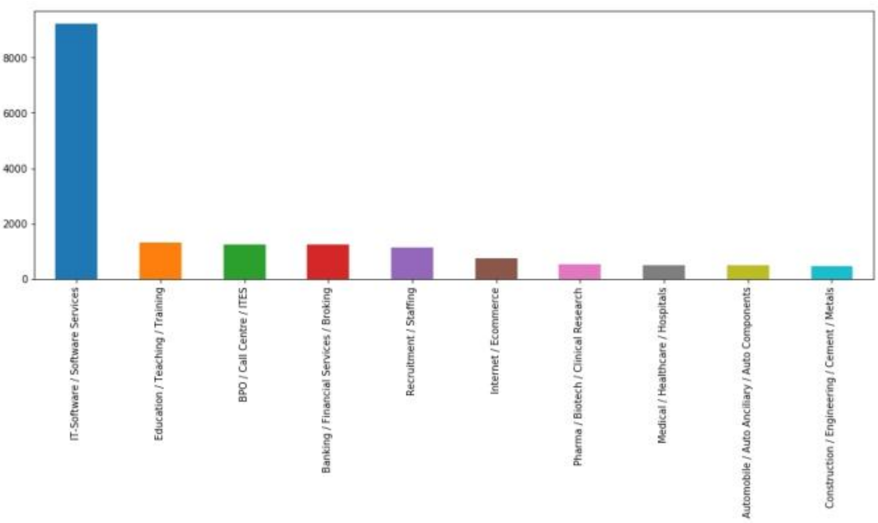


Figure 10. The bar chart of the Top 10 Industries from the dataset.

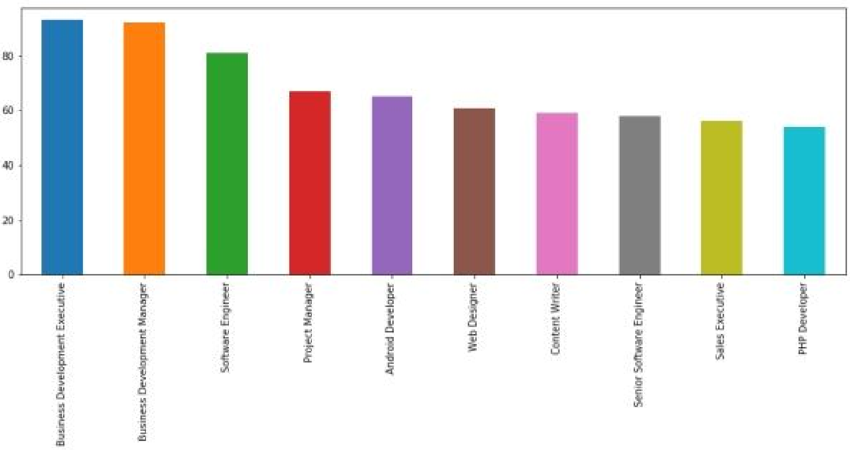


Figure 11. The bar chart of the Top 10 job_title from the dataset.

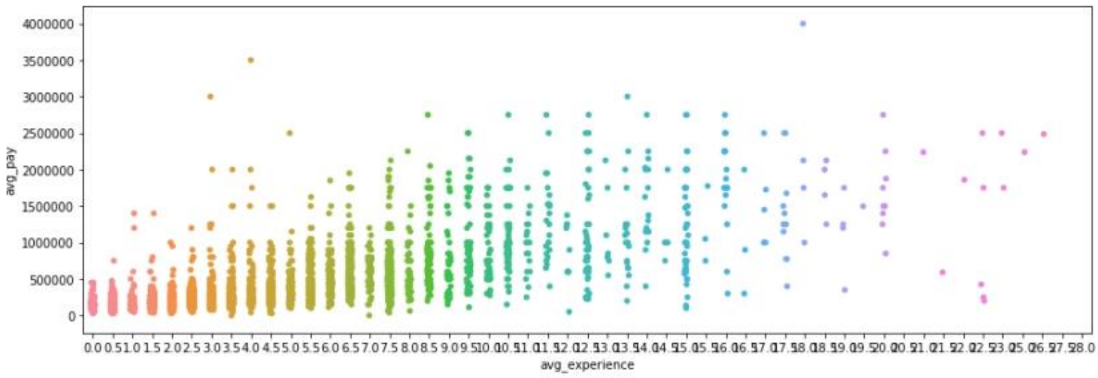


Figure 12. The relationship between average salary an work experience requirement.

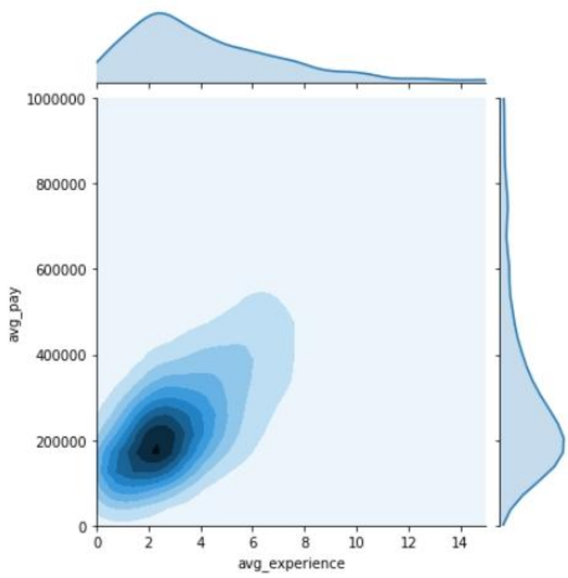


Figure 13. The relationship between average salary and work experience requirement.

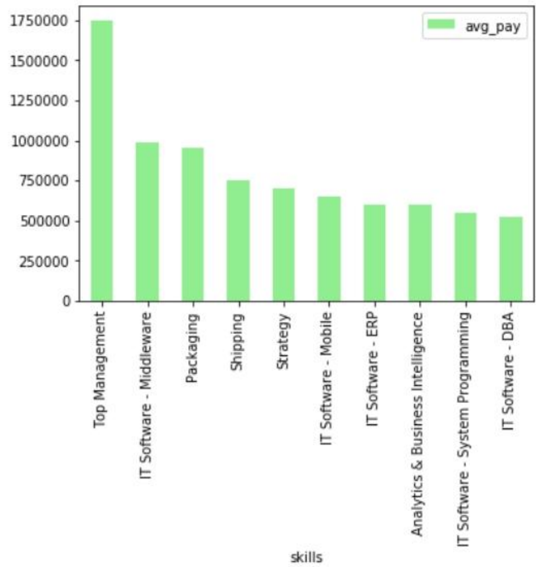


Figure 14. The relationship between average job salary and skills requirement.

5.2 Resume dataset and preprocess

The resume dataset is selected from a collection of Resume Examples taken from livecareer.com for categorizing a given resume into any of the labels defined in the dataset, which contains more than 2400 resumes in a string as well as PDF format. The PDF will be stored in the data folder differentiated into their respective labels as folders with each resume residing inside the folder in PDF form with the filename as the id defined in the CSV. Inside the CSV, there will be the contents as follow：

1. ID: Unique identifier and file name for the respective pdf.
2. Resume_str: Contains the resume text only in string format.
3. Resume_html: Contains the resume data in HTML format as a present while web scrapping.
4. Category: Category of the job the resume was used to apply for.

And inside the Categories, there would be job types including HR, Designer, Information-Technology, Teacher, Advocate, Business-Development, Healthcare, Fitness, Agriculture, BPO, Sales, Consultant, Digital-Media, Automobile, Chef, Finance, Apparel, Engineering, Accountant, Construction, Public-Relations, Banking, Arts, and Aviation.

To facilitate the subsequent use of the Resume_html column and all rows of

Resume_str were removed and the remaining 2482 rows of data were retained.

To facilitate the subsequent use of the Resume_html column and all rows of Resume_str were removed and the remaining 2482 rows of data were retained. And to facilitate the display of the subsequent search results, each resume was given a temporary name and a filename generated from the name, so that the final data set used for the subsequent search contains five columns, as shown in Figure 15. The ID is the filename that actually corresponds to the PDF of the resume in this data set. Based on the category of the resumes, a word cloud is generated for the resumes belonging to each category, and the output is shown in Figure 16.

Figure 15. The word clouds correspond to each category.

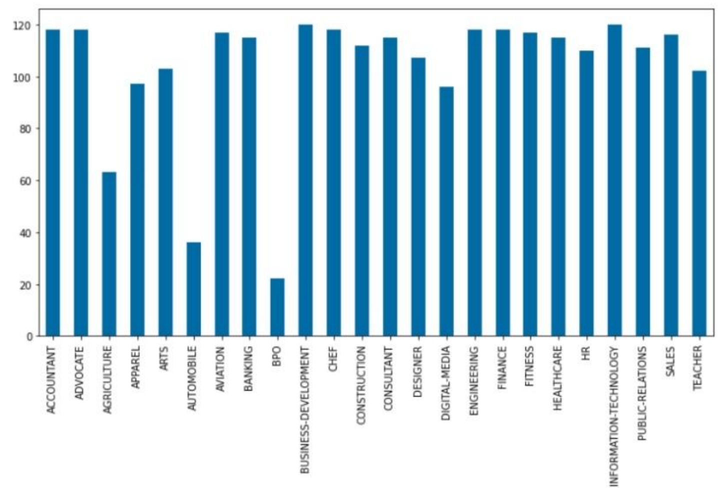


Figure 16. Statistical chart of the number of resumes in each category.

5.3 Implementation of Modelling

To ensure that the relevant text of the job description and the resume text are in the same domain, and in order to be able to compare the text similarity later by calculating the vector distance, the final job text and the resume text need to use the same model for feature vector generation. As the job dataset is much larger than the resume dataset, the job dataset data is considered for model training.

Firstly, the job description text is simply processed by removing many useless and repetitive sentences from the text using the string.strip() function as shown below.

```
job['jobdescription'] = job['jobdescription'].map(lambda x: x.strip().strip('Job Description').strip())
```

```
job['jobdescription'] = job['jobdescription'].map(lambda x: x.strip('Send me Jobs like this').strip())
```

Before process:

```
0    Job Description   Send me Jobs like this Quali...
1    Job Description   Send me Jobs like this Quali...
2    Job Description   Send me Jobs like this - as ...
3    Job Description   Send me Jobs like this - Inv...
4    Job Description   Send me Jobs like this Pleas...
Name: jobdescription, dtype: object
```

After process:

```
0    Qualifications: - == > 10th To Graduation & An...
1    Qualifications: - == > 10th To Graduation & An...
2    - as a developer in providing application desi...
3    - Involved with all stages of indirect taxatio...
4    Please share your Resume on : regina.mary@spir...
Name: jobdescription, dtype: object
```

In order to maximize the use of textual information in the job information, the textual information in the job database such as job title, job description, skills, and industry will be stitched together to form a long text for vector generation. The data set formed for model training is as follows.

	jobtitle	company	jd_combo
0	walkin data entry operator (night shift)	MM Media Pvt Ltd	walkin data entry operator (night shift) Quali...
1	work based onhome based part time.	find live infotech	work based onhome based part time. Qualificati...
2	pl/sql developer - sql	Softtech Career Infosystem Pvt. Ltd	pl/sql developer - sql - as a developer in pro...
3	manager/ad/partner - indirect tax - ca	Onboard HRServices LLP	manager/ad/partner - indirect tax - ca - Invol...
4	java technical lead (6-8 yrs) -	Spire Technologies and Solutions Pvt. Ltd.	java technical lead (6-8 yrs) - Please share y...

Due to the complex lexical composition of long texts, it is expected to be able to automatically define stop word lists based on the constituent words. In that case, the TF-IDF model was utilized to generate a count vector of all the words in the database, and identified the stop words that needed to remove from the word list and added to the list in nltk.corpus.stopwords.

```
#Transforms words to TFIDF
vectorizer = TfidfVectorizer(stop_words = stopwords)

#Fit the vectorizer to the data
vectorizer.fit(df_job_descriptions['jd_combo'].fillna(''))

#Transform the data
tfidf_scores = vectorizer.transform(df_job_descriptions['jd_combo'].fillna(''))
```

The sklearn.feature_extraction.text.TfidfVectorizer function was chosen to build the TF-IDF model, which uses jd_combo for training and count vector generation, and to reselect the deactivated words from the acquired bag of words.

After building a custom list of deactivated words, pre-process needs to be applied to each jd_combo text in the database for NLP. The procedures are as follows.

1. Split sentence into words with spaces.
2. Lowercase all letters in the words.
3. Use str.maketrans() to create a table for removing punctuation.
4. Use nltk.stem import WordNetLemmatizer to lemmatize.

To better transform word patterns, pos_tag() is also used for lexical annotation and lexical reduction based on word lexicality.

```
# defining tokenizer
def my_tokenizer(text):

    # 1. split at whitespace
    text = text.split(' ')
    #2. lowercase
    text = [word.lower() for word in text]
    #3. Remove puncutation
    #table to replace puncuation
    punc_table = str.maketrans('','',string.punctuation)

    #call translate()
    text = [word.translate(punc_table) for word in text]

    #4. remove stopwords
    text = [word for word in text if word not in stopwords]

    #5. lemmmatize
    tagged_sent = pos_tag(text)
    wnl = WordNetLemmatizer()
    lemmas_sent = []
    for tag in tagged_sent:
      wordnet_pos = get_wordnet_pos(tag[1]) or wordnet.NOUN
      lemmas_sent.append(wnl.lemmatize(tag[0], pos=wordnet_pos))

    #6. remove empty strings
    text = [word for word in lemmas_sent if word !='']

    return text
```

Finally, for model building and training, the TaggedDocument is selected to generate the paragraph id needed for training the Doc2vec model, and train 200 epochs to generate a vector of dimension 20.

```
# Convert tokenized document into gensim formated tagged data
tagged_data = [TaggedDocument(d, [i]) for i, d in enumerate(tokenized_doc)]
```

```
#train model - final******** with 200 epochs
epoch_logger = EpochLogger()
## Train doc2vec model
model1 = Doc2Vec(tagged_data, vector_size=20, window=2, min_count=1, workers=4, epochs = 200, callbacks=[epoch_logger])
```

```
Epoch #0 start
Epoch #0 end
Epoch #1 start
Epoch #1 end
Epoch #2 start
Epoch #2 end
Epoch #3 start
Epoch #3 end
Epoch #4 start
Epoch #4 end
Epoch #5 start
Epoch #5 end
Epoch #6 start
Epoch #6 end
Epoch #7 start
Epoch #7 end
Epoch #8 start
Epoch #8 end
Epoch #9 start
Epoch #9 end
Epoch #10 start
Epoch #10 end
Epoch #11 start
Epoch #11 end
```

Model storage and reading were implied after model training, and model.infer_vector() generates a twenty-dimensional feature vector for each line of text data.

```
# Save trained doc2vec model
model1.save("models/my_doc2vec.model")
```

```
## Load saved doc2vec model
model1= Doc2Vec.load("models/my_doc2vec.model")
```

```
## Get vector value
vec = np.empty([21996,20])

for k,i in enumerate(tokenized_doc):
    #print(i)
    vector = model1.infer_vector(i)
    vec[k] = vector

# reshape into 2D
new_arr = np.reshape(vec,(-1,20))
```

```
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   vec_1    21996 non-null  float64
 1   vec_2    21996 non-null  float64
 2   vec_3    21996 non-null  float64
 3   vec_4    21996 non-null  float64
 4   vec_5    21996 non-null  float64
 5   vec_6    21996 non-null  float64
 6   vec_7    21996 non-null  float64
 7   vec_8    21996 non-null  float64
 8   vec_9    21996 non-null  float64
 9   vec_10   21996 non-null  float64
 10  vec_11   21996 non-null  float64
 11  vec_12   21996 non-null  float64
 12  vec_13   21996 non-null  float64
 13  vec_14   21996 non-null  float64
 14  vec_15   21996 non-null  float64
 15  vec_16   21996 non-null  float64
 16  vec_17   21996 non-null  float64
 17  vec_18   21996 non-null  float64
 18  vec_19   21996 non-null  float64
 19  vec_20   21996 non-null  float64
dtypes: float64(20)
memory usage: 3.4 MB
```

The generated vectors and database will be re-stored for later work on the back end of the webpage, the matching process. For the data of resumes, the same NLP data pre-processing operations will be sampled to generate the same 20-dimensional feature vector using the previously trained model.

5.4 Implementation of Website

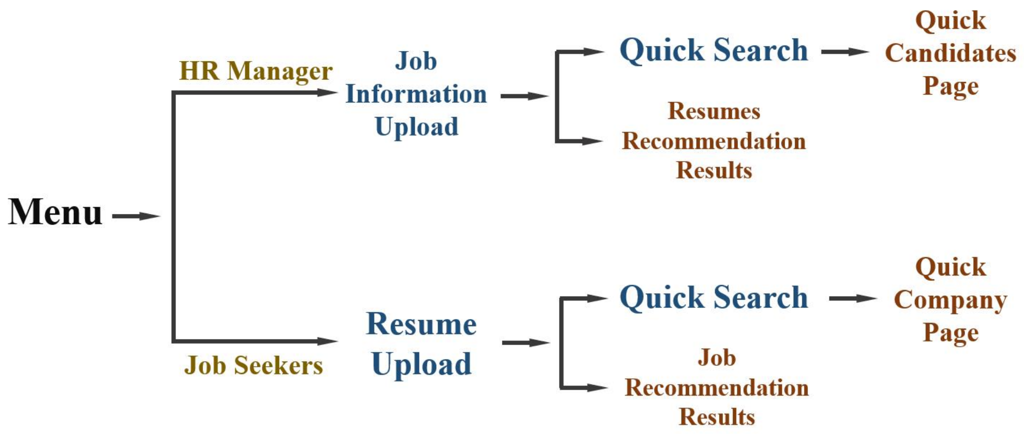


Figure 17. The design framework of website.

Figure 17 shows the architecture design of the website in general. For the website building part, the Flask web framework was chosen to enable local initialization of the web server and navigation to other pages, which is a list of routes and their functions in that specific route. The routes will be defined as @app.routes() and the functions below that route will be the function which needs to be done on that page. In Figure x shown below, some path variables have been defined.

```python

import os
import urllib.request
from flask import (Flask, render_template, request,
                   send_from_directory, jsonify)
from werkzeug.utils import secure_filename


app = Flask(__name__)
#app.secret_key = "secret key"
app.config['UPLOAD_FOLDER'] = './Resumes/'
app.config['JD_Folder'] = './JobDesc/'


ALLOWED_EXTENSIONS = set(['txt', 'pdf'])

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

def uploadFile(file):
    Filename = secure_filename(file.filename)
    file.save(os.path.join(app.config['UPLOAD_FOLDER'], Filename))
    return Filename


@app.route('/')
def index():
    return render_template('index.html')

@app.route('/resources/<path:path>')
def send_resources(path):
    return send_from_directory('resources', path)

@app.route('/uploadResumePage')
def uploadResumePage():
    return render_template('uploadResumePage.html')
```

As shown below, a case statement has been defined, when running a python file, the variable will be set with the value __main__, which means the server is in a specified port when running the file and the file will be waiting for the user work, and the web page will be active in the local server.

```python

@app.route('/updateJDAction', methods=['POST'])
def updateJDAction():
    name = request.form['name']
    Description = request.form['Description']
    # Following code is to job description file to server
    if Description:
        with open(f"{app.config['JD_Folder']}{name}.txt", "w", encoding = 'utf-8') as file:
            file.write(Description)
    else:
        result = "Please enter Job Description for profile match"

    return render_template('PageResult.html')

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=8080, debug=True)

```

After initializing the server, the web page can be able to access by the localhost web URL, which is http://localhost:8080/. Flask will launch the main home page through this link, and the server.py script will be called by using the render_template() function defined in the route, and the function is shown below

```python
@app.route('/')
def index():
    return render_template('index.html')
```

render_template() is a Flask function which is used to generate output from a template file based on the Jinja2 engine. Once the template folder is created in the same path as the file, the template will be visible as shown in Figure 18.

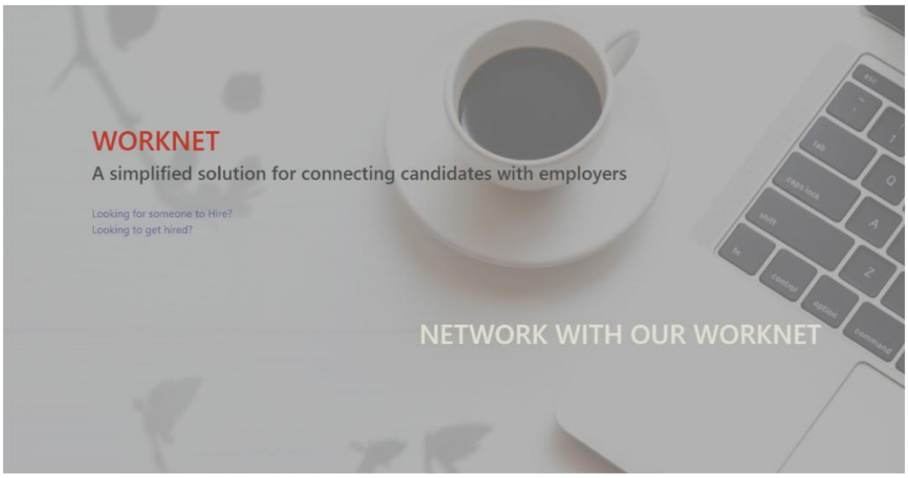


Figure 18. The Home Page.

What is shown below is two links are defined under the index page. The Flask route is also used to define the pages to upload the resume files and Job description. When an HR manager or a job seeker finish uploading, the respective route and functions will be called from the server.py file.

```html
<div class="row">
    <div class="col-sm"><a href="/updateJDPage"><FONT COLOR="#60679B">Looking for someone to Hire?</a></div>
    <div class="col-sm"></div>
    <div class="col-sm"></div>
</div>
<div class="row">
  <div class="col-sm"><a href="/uploadResumePage"><FONT COLOR="#60679B">Looking to get hired?</a></div>
  <div class="col-sm"></div>
  <div class="col-sm"></div>
```

For the information upload pages, once uploading is finished, the href reference will be used to route through different pages.

```python
# server.py > index
@app.route('/updateJDPage')
def classifierPage():
    return render_template('updateJDpage.html')
```

```python
@app.route('/uploadResumePage')
def uploadResumePage():
    return render_template('uploadResumePage.html')
```

As shown in Figure 19 and Figure 20 below, on this page, several fields are provided for users to type in the input text, and a submit option is set as a button.

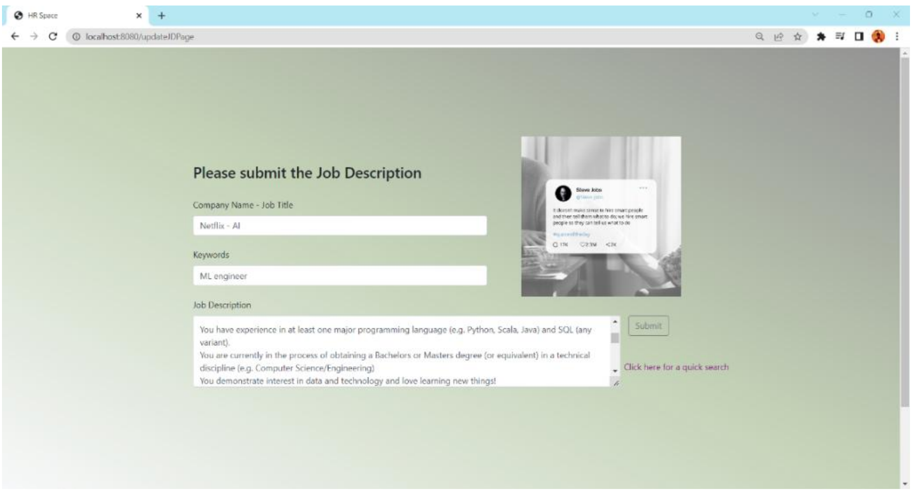


Figure 19. The information uploading page for HR managers.

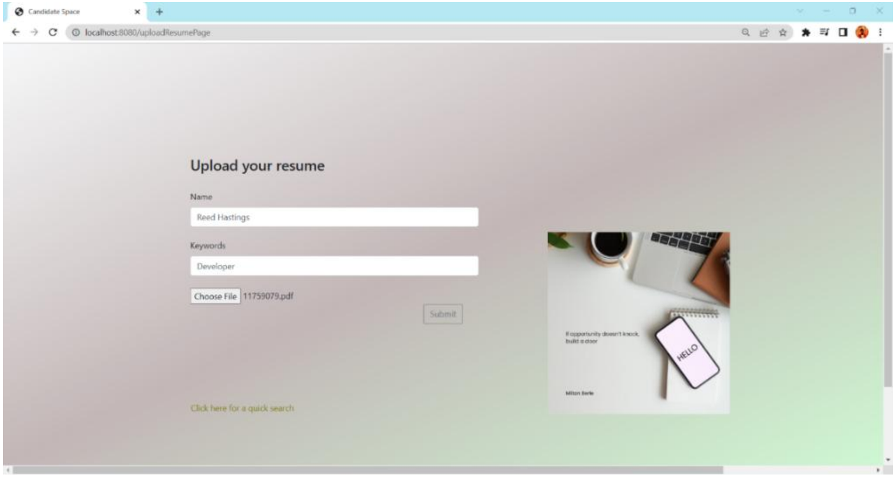


Figure 20. The information uploading page for job seekers.

The exact code to implement the successful submission function is shown below. Once the submission button is clicked, there will be an associated function via calling the route of "/updateJDAction" or "/uploadResumePage" and executing it.

```
div class="py-5 text-center">
  <form enctype = "multipart/form-data" action="/updateJDAction" method="post">
    <div class="form-group">
      <div class="row">
        <div class="col-sm"><h3 align="left">Please submit the Job Description</h3></div>
```

```
@app.route('/updateJDAction', methods=['POST'])
def updateJDAction():
    name = request.form['name']
    Keywords = request.form['keyword']
    Description = request.form['Description']
    # Following code is to job description file to server
    if Description:
        with open(f"{app.config['JD_Folder']}{name}.txt", "w", encoding = 'utf-8') as file:
            file.write(Description)

        resume_data = pd.read_csv('./resume_db.csv', index_col=0)

        result = viewing.dataframe(resume_data)
        #result = table.to_html()
        #result = result.to_html(classes='table table-stripped')

    else:
        result = "Please enter Job Description for profile match"

    return render_template('JDPageResult.html', results = result)
```

The name will be obtained from the job title field in the Name variable. The keyword and the description will be acquired from the text field and stored in the Description variable. A text file in the configured location will be created with the name of the file as the job title provided by HR.

```
<form enctype = "multipart/form-data" action="/uploadResumeAction" method="post">
    <div class="form-group">
        <div class="row">
            <div class="col-sm"><h3 align="left">Upload your resume</h3></div>
            <div class="col-sm"></div>
        </div>
        <div class="row">
```

```python
@app.route('/uploadResumeAction', methods=['POST'])
def uploadResumeAction():
    name = request.form['name']
    Keywords = request.form['keyword']
    # Following code is to upload resume file to server
    try:
        uploadfile = request.files['uploadfile']
    except:
        uploadfile = None

    if uploadfile:
        uploadfilename = uploadFile(uploadfile)

        resume_data = pd.read_csv('./jobs_db.csv', index_col=0)

        result = viewing.dataframe(resume_data)
    else:
        result = "No Profile to upload"

    return render_template('ResumePageResult.html', results = result)
```

For the information upload page of resumes, the name and the uploaded file from field variables will be obtained and the file in the server directory will be stored in the configured location, which is app. config['UPLOAD_FOLDER'] = '. /Resumes/'. The stored file will then be picked up by the model and processed to get a matched result data frame.

Once the script gets the data frame from the model, it will pass the result variable into the results page, which is JDResultsPage.html. and ResumePageResult.html. Finally, it will give a result page in a formatted way.

As an additional functionality, a quick search page is designed for both HR managers and job seekers.

For HR managers, this page provides a function to do a quick screening of candidates by typing a simple keyword, and it will be collected and applied a simple lookup on the available Database files.

```python
@app.route('/QuickCandidates')
def QuickSearchCandidatesPage():
    return render_template('QuickSearchCandidates.html')
```

```html
<form enctype = "multipart/form-data" action="/QSSkillsResults" method="post">
    <div class="form-group">
        <div class="row">
            <div class="col-sm"><h5 align="left">Please type any skills that you're looking for</h5></div>
            <div class="col-sm"></div>
        </div>
        <div class="row">
            <div class="col-sm"> </div>
        </div>
        <div class="row">
            <div class="col-sm-5" align="left"><label for="name">Keywords</label></div>
        </div>
```

```python
@app.route('/QSSkillsResults', methods=['POST'])
def QSSkillsResults():
    keyword = request.form['name']
    # Following code is to job description file to server
    if keyword:
        keyword = request.form['name']
        quickcv = pd.read_csv('./DB/Resume.csv', index_col=0)
        filtered_resume = quickcv.sample(frac=1)
        filtered_resume = filtered_resume.loc[filtered_resume['Resume_str'].str.lower().str.contains(keyword.lower())].head(10)
        filtered_resume = filtered_resume.drop(columns= ['Resume_str', 'Resume_html'])
        result = viewing.dataframe(filtered_resume)
        #result = table.to_html()
        #result = result.to_html(classes='table table-stripped')

    else:
        result = "Please enter Job Description for profile match"

    return render_template('QSCandidatesResult.html', results = result)
```

One example of quick search results is shown in Figure 21 below.

Resume ID	Resume category
12351749	SALES
12011623	ENGINEERING
21156767	CONSULTANT
13586069	CONSULTANT
10985403	ENGINEERING
27080812	DIGITAL-MEDIA
28923650	ENGINEERING
28505854	ENGINEERING
22946204	AUTOMOBILE
34953092	BANKING

Figure 21. The example of quick search results.

For job seekers, this page provides a function to do a quick screening of jobs, and it will also be collected and applied a simple lookup on the available Database files.

```
@app.route('/QuickJobs')
def QuickJobs():
    return render_template('QuickSearchJobs.html')
```

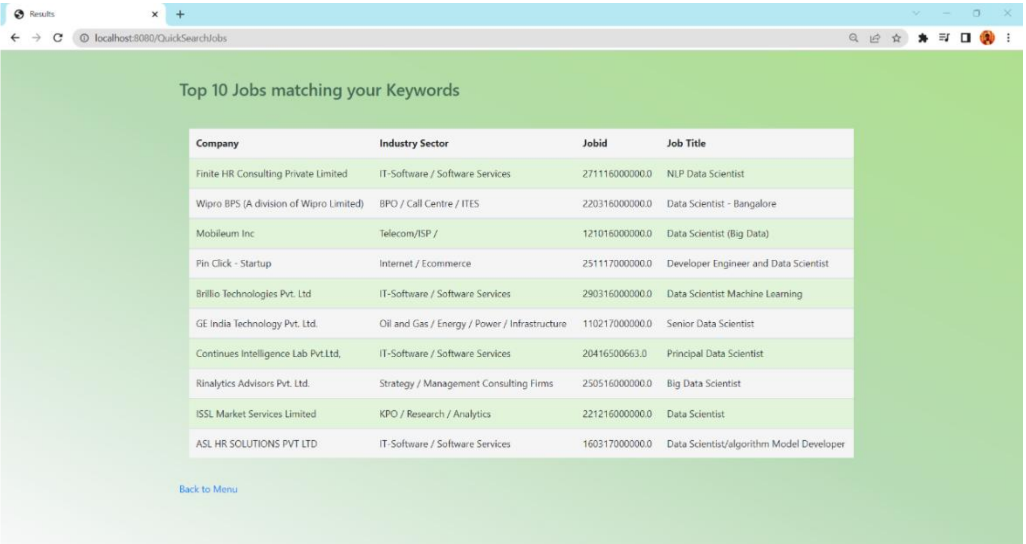
```
<form enctype = "multipart/form-data" action="/QuickSearchJobs" method="post">
    <div class="form-group">
        <div class="row">
            <div class="col-sm"><h3 align="left">Please type a Job title</h3></div>
            <div class="col-sm"></div>
        </div>
        <div class="row">
            <div class="col-sm"> </div>
        </div>
        <div class="row">
            <div class="col-sm-5" align="left"><label for="name">Keywords</label></div>
        </div>
        <div class="row">
```

```
@app.route('/QuickSearchJobs', methods=['POST'])
def QuickSearchJobs():
    keyword = request.form['name']
    quickjob = pd.read_csv('./DB/jobs.csv', index_col=0)
    filtered_jobs = quickjob.sample(frac=1)
    filtered_jobs = filtered_jobs.loc[filtered_jobs['jobtitle'].str.lower().str.contains(keyword.lower())].head(10)
    filtered_jobs = filtered_jobs.drop(columns= ['education', 'experience', 'jobdescription',
                                                 'joblocation_address', 'numberofpositions', 'payrate',
                                                 'postdate', 'site_name', 'skills', 'uniq_id'])
    # Following code is to job description file to server
    if keyword:
        result = viewing.dataframe(filtered_jobs)
        #result = table.to_html()
        #result = result.to_html(classes='table table-stripped')

    else:
        result = "Please enter keyword for a quick profile match"

    return render_template('QSJobsResult.html', results = result)
```

One example of quick search results is shown in Figure 22 below.



Company	Industry Sector	Jobid	Job Title
Finite HR Consulting Private Limited	IT-Software / Software Services	271116000000.0	NLP Data Scientist
Wipro BPS (A division of Wipro Limited)	BPO / Call Centre / ITES	220316000000.0	Data Scientist - Bangalore
Mobileum Inc	Telecom/ISP /	121016000000.0	Data Scientist (Big Data)
Pin Click - Startup	Internet / Ecommerce	251117000000.0	Developer Engineer and Data Scientist
Brillio Technologies Pvt. Ltd	IT-Software / Software Services	290316000000.0	Data Scientist Machine Learning
GE India Technology Pvt. Ltd.	Oil and Gas / Energy / Power / Infrastructure	110217000000.0	Senior Data Scientist
Continues Intelligence Lab Pvt.Ltd,	IT-Software / Software Services	20416500663.0	Principal Data Scientist
Rinalytics Advisors Pvt. Ltd.	Strategy / Management Consulting Firms	250516000000.0	Big Data Scientist
ISSL Market Services Limited	KPO / Research / Analytics	221216000000.0	Data Scientist
ASL HR SOLUTIONS PVT LTD	IT-Software / Software Services	160317000000.0	Data Scientist/algorithm Model Developer

Figure 22. The example of quick search results.

6. Project Performance & Validation

6.1 Home Page

Figure 23 shows the home page of the website after successfully running the server. Two types of services are set for users to choose from. One is for HR managers who looking for employees, and the other is for job seekers to find the desired job.

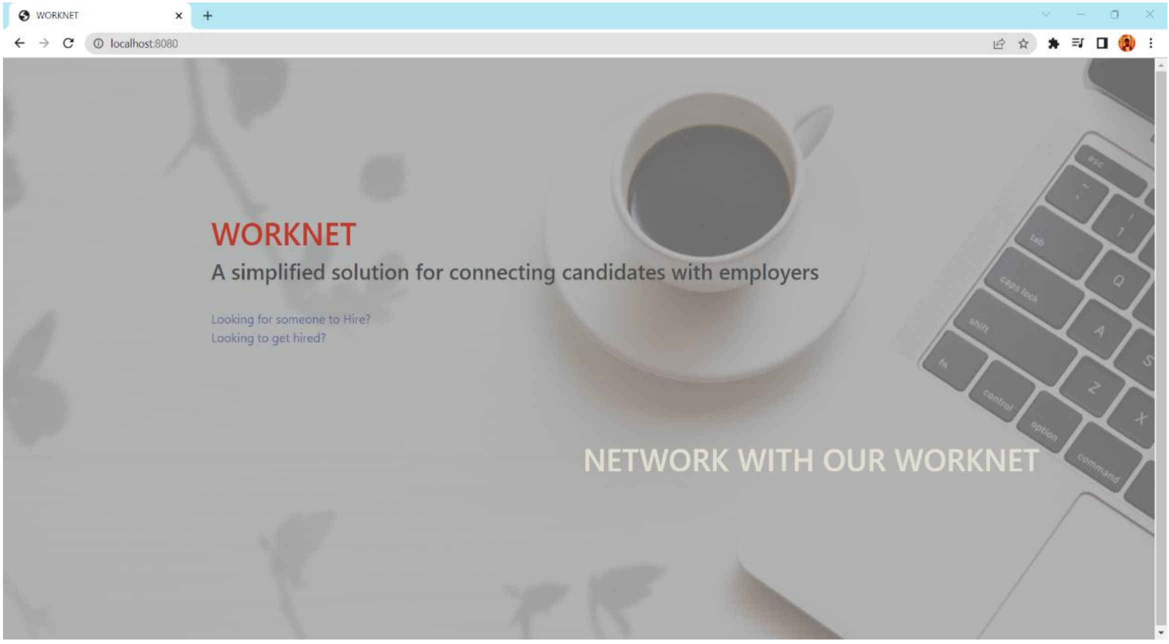


Figure 23. The Home Page.

6.2 Information Upload Page

6.2.1 For HR Manager

As shown in Figure 24, a page for HR managers to upload their job descriptions is set. Once a job title and description are uploaded successfully, a text file will be created in the folder structure of the websites, and the model will be called to get the recommendation results that match the job requirements.

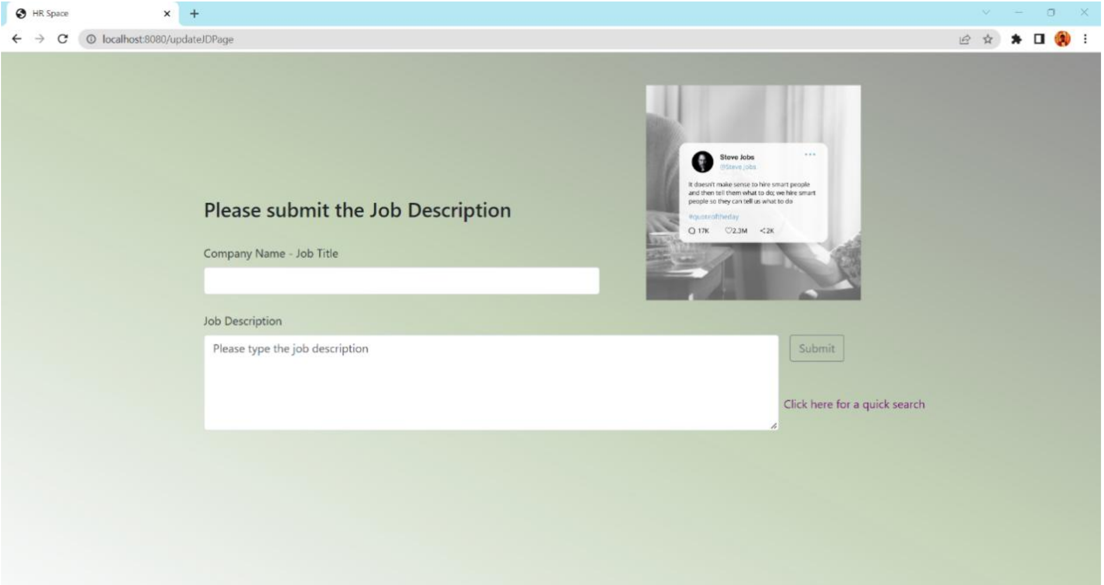


Figure 24. The information uploading page for HR managers.

6.2.2 For Job Seekers

As shown in Figure 25, a page for job seekers to upload their resumes and personal information profile is set. Once a resume is uploaded successfully, a text file will be created in the folder structure of the websites, and the model will be called to get the recommendation results that match the personal information of job seekers

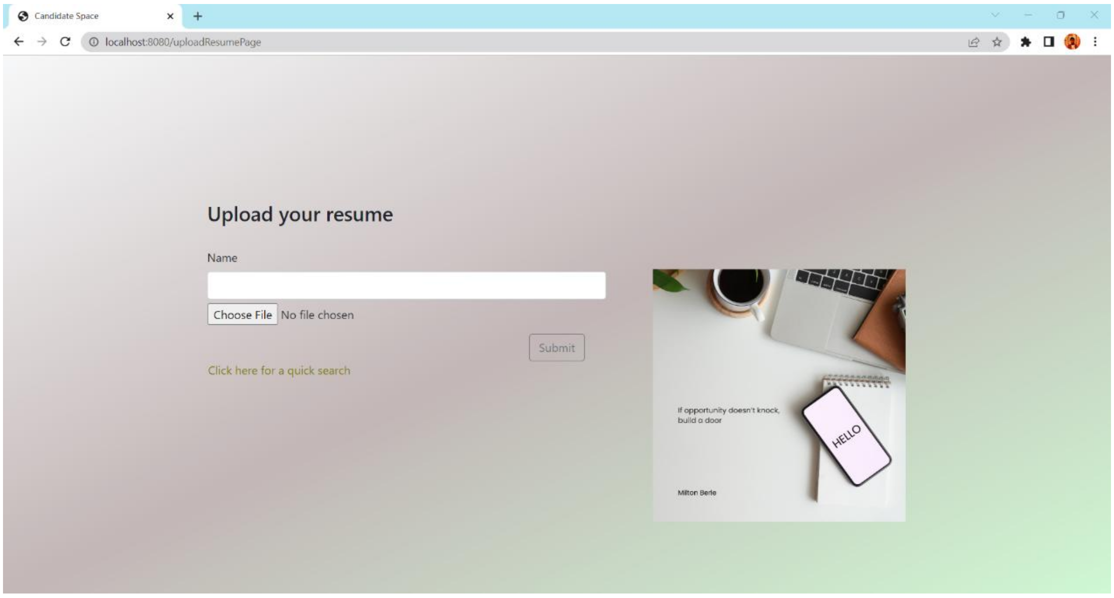


Figure 25. The information uploading page for job seekers.

6.3 Recommendation Result Pages

After the model works successfully, the top 10 recommendation candidates that match the job description or the top 5 jobs that match the individual information of job seekers will be shown on the recommendation result pages, the examples are shown in Figure 26 and Figure 27.

Resume ID	Resume category
28505854	ENGINEERING
22946204	AUTOMOBILE
13586069	CONSULTANT
23015611	PUBLIC-RELATIONS
10985403	ENGINEERING
12351749	SALES
34953092	BANKING
28923650	ENGINEERING
50328713	ENGINEERING
27080812	DIGITAL-MEDIA

Figure 26. The example of recommendation result page for HR managers.

Company	Jobid	Job Title
Premium	101000000000.0	Business Development Manager
Convate hiring for Retail/e-commerce domain	61016003187.0	Opening for Android Developer-bangalore-4-8 yrs
Sinai Global	131000000000.0	Full Stack Web Application (php) Developer
Mathematical Computing Software Company	131000000000.0	Looking for an Application Engineer-fpga
Accenture	121000000000.0	Revenue Assurance
PKS & Associates	121000000000.0	Senior Java Full Stack Developer - Java/ Spring 4
SMACera Technologies Consulting and Services Pvt L hiring for Smacera Technologies	30916001587.0	Data Entry Executive
Janak Vidya Consultancy Pvt. Ltd.	131000000000.0	PHP Developer_banashankari II Stage
Strivex Consulting Pvt Ltd	81016900140.0	Bluetooth Host Software Development Engineer - Automotive Infotainment

Figure 27. The example of recommendation result page for job seekers.

6.4 Quick Search Page

For HR managers or job seekers to search for expected skills and abilities or desired jobs by typing keywords, a quick search page is set to provide a more direct and fast way to get the recommendation information. The model will be called after receiving the instruction, and then get the top 10 candidates that match the keywords from the database, the examples are shown in Figure 28 and Figure 29.

6.4.1 For HR Manager

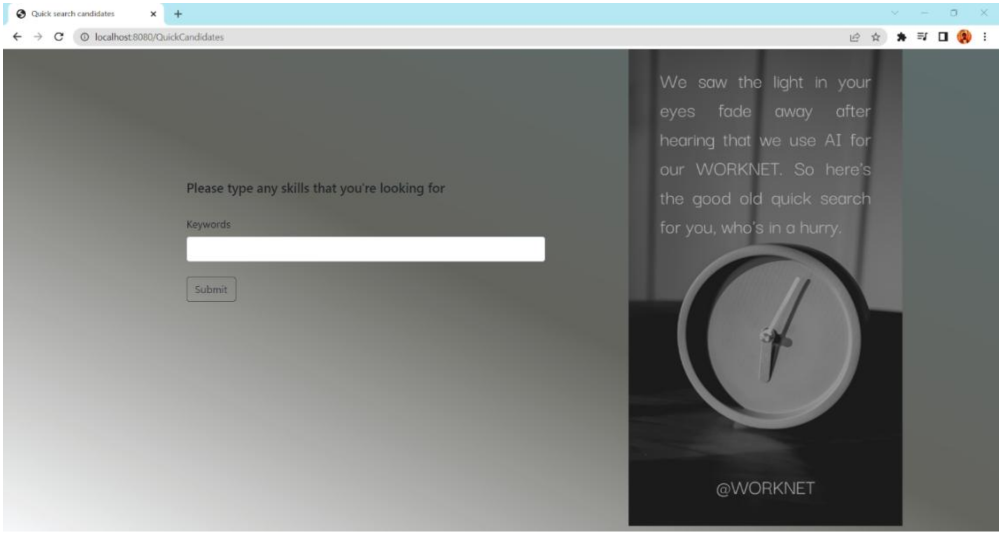


Figure 27. The quick research page for HR managers.

6.4.2 For Job Seekers

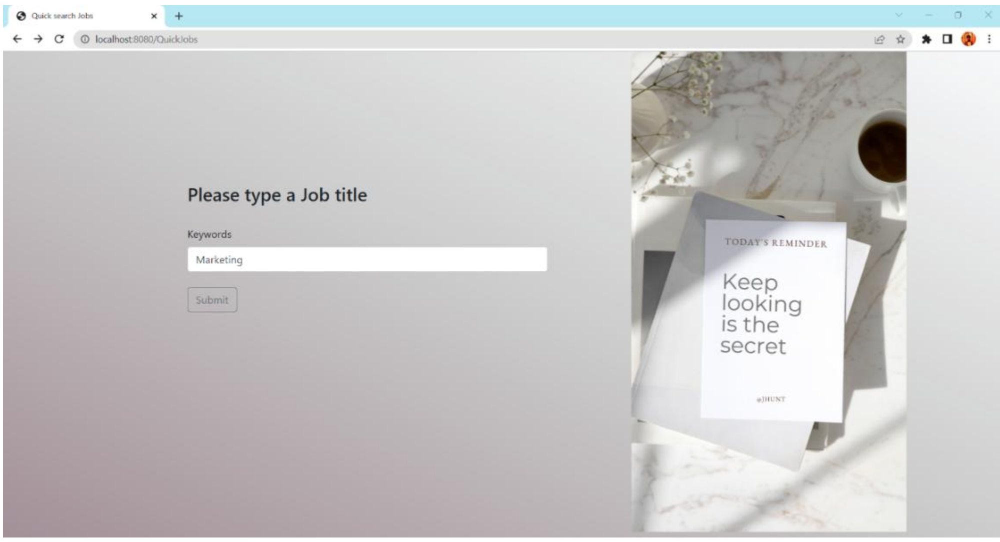
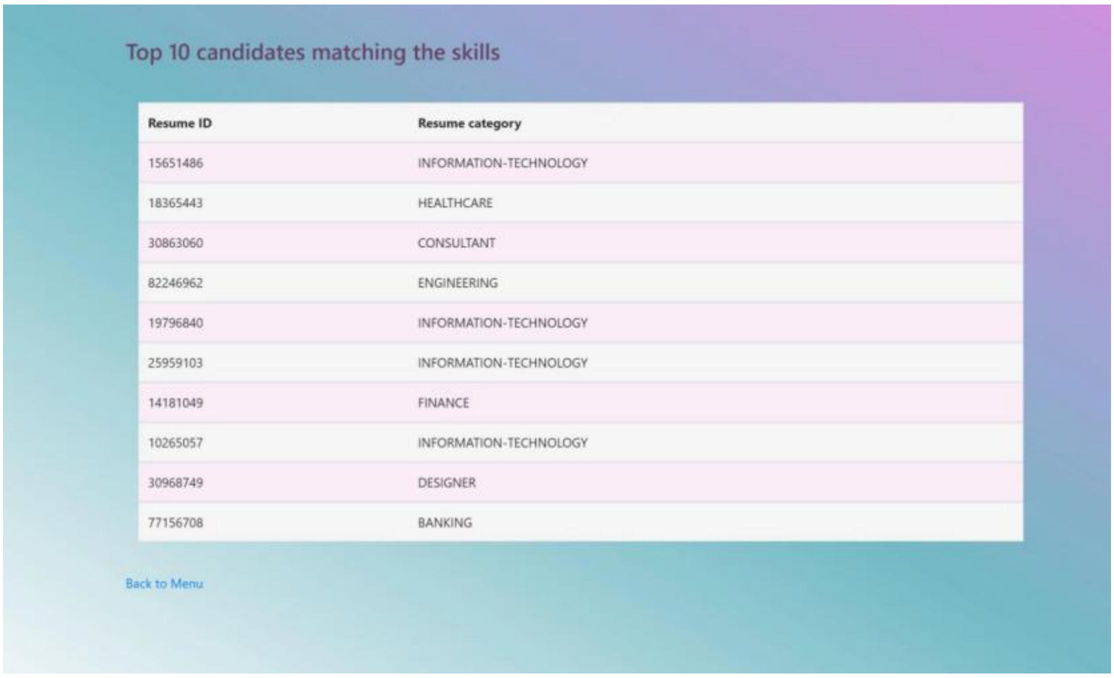


Figure 28. The quick research page for job seekers.

6.5 Recommendation Result Page for Quick Search

The examples of quick search results are shown in Figure 29 and Figure 30 below.

6.5.1 For HR Managers



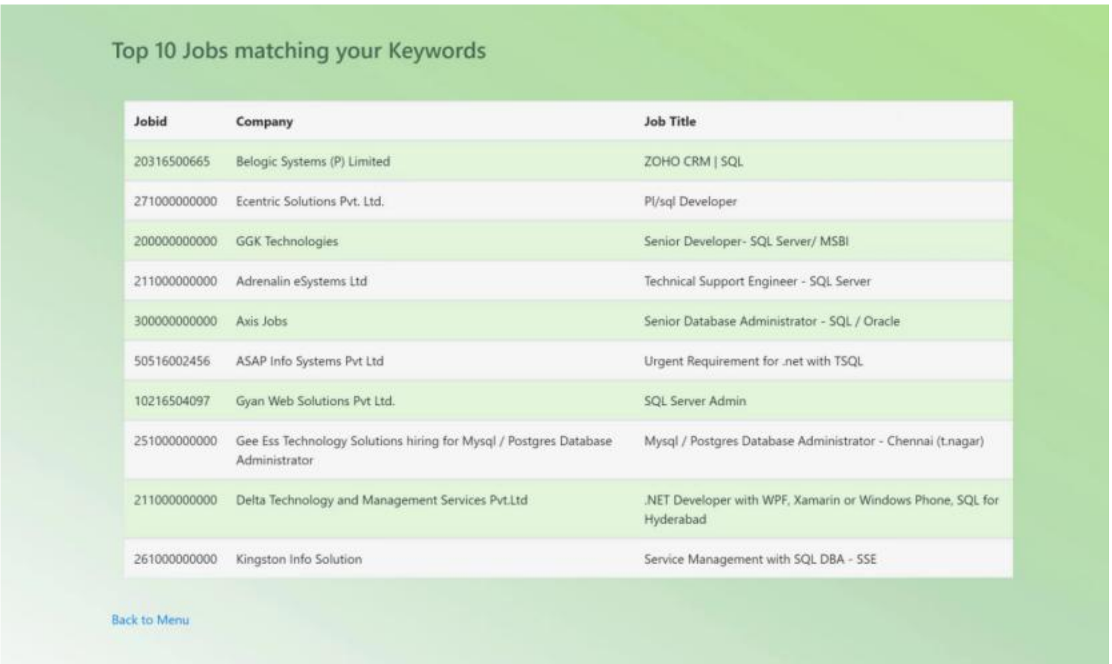
Top 10 candidates matching the skills

Resume ID	Resume category
15651486	INFORMATION-TECHNOLOGY
18365443	HEALTHCARE
30863060	CONSULTANT
82246962	ENGINEERING
19796840	INFORMATION-TECHNOLOGY
25959103	INFORMATION-TECHNOLOGY
14181049	FINANCE
10265057	INFORMATION-TECHNOLOGY
30968749	DESIGNER
77156708	BANKING

Back to Menu

Figure 29. The recommendation page of quick research for HR managers.

6.5.2 For Job Seekers



Top 10 Jobs matching your Keywords

Jobid	Company	Job Title
20316500665	Belogic Systems (P) Limited	ZOHO CRM | SQL
271000000000	Ecentric Solutions Pvt. Ltd.	Pl/sql Developer
200000000000	GGK Technologies	Senior Developer- SQL Server/ MSBI
211000000000	Adrenalin eSystems Ltd	Technical Support Engineer - SQL Server
300000000000	Axis Jobs	Senior Database Administrator - SQL / Oracle
50516002456	ASAP Info Systems Pvt Ltd	Urgent Requirement for .net with TSQL
10216504097	Gyan Web Solutions Pvt Ltd.	SQL Server Admin
251000000000	Gee Ess Technology Solutions hiring for Mysql / Postgres Database Administrator	Mysql / Postgres Database Administrator - Chennai (t.nagar)
211000000000	Delta Technology and Management Services Pvt.Ltd	.NET Developer with WPF, Xamarin or Windows Phone, SQL for Hyderabad
261000000000	Kingston Info Solution	Service Management with SQL DBA - SSE

Back to Menu

Figure 30. The recommendation page of quick research for job seekers.

7. Project Conclusion

From the content of Project Solution, Project Implementation, and Project Performance & Validation introduced above, a bidirectional job and resume matching and recommendation system is designed, the objectives of this project have also been achieved successfully. Both HR managers and job seekers can upload their requirements, and the matching job information or the candidates will be recommended according to the features of text extracted by the pre-trained model. The recommendation function by quick search is also set inside, which provides a more efficient and simple way to conduct the information screening.

For the perspective of this project, some optimization suggestions will be made as follows.

1. More visual displays can be added, such as showing the full information on the web page for recommended resumes.
2. Improving the dynamic resumes and jobs database to capture the data in real time and to be able to update and modify it according to the uploaded content.
3. Using more information in job and resume matching, such as working location.
4. Optimize the functions of the website, such as user registration and login.
5. Acquire the domain name and make the website publicly available on the internet.

8. References

[1] Zeng Chun, Xing Chunxiao, Zhou Lizhu. A review of personalized service technology[J]. Journal of Software, 2002, 13(10):10.

[2] The employment of Singapore, CEIC DATA, 2022. <https://www.ceicdata.com/zh-hans/indicator/singapore/employed-persons>

[3] Hong WX, Wang N, Chen YW, et al. Talent recommendation system in the era of big data[J]. Big Data, 2017, 3(2):6.

[4] GONZALEZ T, SANTOS P, OROZCO F, et al.Adaptive employee profile classification for resource planning tool [C]. 2012 Annual SRII Global Conference, 2012.

[5] HONG W X , LI L , LI T, et al. iHR:an online recruiting system for xiamen talent service center [C]. The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2013.

[6] Li Tao. Application and practice of data mining: case studies in the era of big data [M]. Xiamen University Press, 2013.

[7] ADOMAVICIUS G , TUZHILIN A. Toward the next generation of recommender systems:a survey of the state-of-theart and possible extensions[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(6):734-749.

[8] PIZZATO L, REJ T, CHUNG T, et al, Reciprocal recommenders[C]. The 8th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems, June 20-24, 2010, Hawaii, USA.

[9] PIZZATO L, REJ T, CHUNG T, et al, Recon:a reciprocal recommender for online dating[C].The 4th ACM Conference on Recommender Systems, September 26-30, 2010. Barcelona, Spain, New York: ACM Press.

[10] MALINOWSKI J, KEIM T, WENDT O. et al. Matching people and jobs:a bilateral recommendation approach[C]. The 39th Annual Hawaii International Conference on System Sciences, January 4-7, 2006, Kauai, Hawaii, USA, New Jersey: IEEE Press, 2006:137c.

[11] YU H, LIU C, ZHANG F, Reciprocal recommendation algorithm for the filed of recruitment[J].Journal of Information& Computational Science, 2011,8(16):4061-4068.

[12] Mikolov T , Chen K , Corrado G , et al. Efficient Estimation of Word Representations in Vector Space[J]. arXiv e-prints, 2013.

[13] Le Q V , Mikolov T . Distributed Representations of Sentences and Documents[J]. arXiv e-prints, 2014.

APPENDIX A

PROJECT PROPOSAL

GRADUATE CERTIFICATE: Intelligent Reasoning Systems (IRS)

PRACTICE MODULE: Project Proposal

Date of Proposal: 26th October 2022
Project Title: IRS Project: Bidirectional Job & Resume Matching System
Sponsor/Client: *(Name, Address, Telephone No. and Contact Name)* Institute of Systems Science (ISS) at 25 Heng Mui Keng Terrace, Singapore NATIONAL UNIVERSITY OF SINGAPORE (NUS) Contact: Mr. GU ZHAN / Lecturer & Consultant Telephone No.: 65-6516 8021 Email: zhan.gu@nus.edu.sg
Background/Aims/Objectives: The objective of this project is to design and build a Bidirectional Job and Resume Matching System. The system should allow both HR managers and job seekers to upload their profiles, and when an HR manager screening the candidates or a job seeker trying to find the appropriate job, the system can provide an accurate matching result, which would offer a convenient and efficient environment to achieve the recruitment in a primary step.
Requirements Overview: 1. Marketing research and feasibility evaluation 2. Data extraction and elicitation 3. Ability of Programming with Python 4. NLP Knowledge 5. Ability in website building: HTML, CSS, JavaScript 6. Environment configuration 7. Understand and design the machine learning model
Resource Requirements: (Please list Hardware, Software and any other resources) Hardware proposed for consideration: CPU Software proposed for consideration: Hardware proposed for consideration: CPU Software proposed for consideration: platform: win-64

flask=2.2.2

python=3.7.13

pip=22.1.2

pillow=9.2.0

pandas=1.3.5

numpy=1.21.6

matplotlib=3.5.1

ipykernel=6.9

json5=0.9.6

Number of Learner Interns required: (Please specify their tasks if possible)

Number of Team Members: 3

Name	Student Number	Tasks
LI YURUI	A0261750J	System Architecture Design Data Acquisition and Processing Retrieval Model Development
LI FANGQING	A0261793W	Product Prototype Design: Business Flow Design Report Writing and Video Recording
PRADEEP KUMAR ARUMUGAM	A0261606J	User Interface Development Website Development Video Recording

Team Formation & Registration

Project Title: **Bidirectional Job & Resume Matching Systems**
System Title: **WORKNET**
Team Member 1
Name: LI YURUI
Matriculation Number: A0261750J
Contact (Mobile/Email): +65 87905967/E0983144@u.nus.edu
Team Member 2
Name: LI FANGQING
Matriculation Number: A0261793W
Contact (Mobile/Email): +65-87906039/E0983187@u.nus.edu
Team Member 3
Name: PRADEEP KUMAR ARUMUGAM
Matriculation Number: A0261606J
Contact (Mobile/Email): +65 83019905/ e0983000@u.nus.edu
Advisor Assigned
Contact: Mr. GU ZHAN / Lecturer & Consultant Telephone No.: 65-6516 8021 Email: zhan.gu@nus.edu.sg

APPENDIX B

Mapped System Functionalities against knowledge, techniques, and skills of modular courses: MR, RS, CGS

- **Pre-processing of Text**
- **Data Clean by TF-IDF**
- **Data Clean by SQL**
- **Natural Language Processing**
- **Doc2Vec Model**
- **Similarity & Distance Measures**
- **Content Base Recommendation**

APPENDIX C

Installation and User Guide

Installation

1. In Local Machine

- git clone <https://github.com/lyrrrr/IRS-PM-2022-IS04FT-GRP-WorkNet>
- pip install the components from requirements.txt
- cd C:\Users\sampleuser\Downloads\WORKNET\webapp_model
- Run the server.py with
  C:\Users\sampleuser\Downloads\WORKNET\webapp_model>py server.py
- Go to URL using web browser <http://localhost:8080/ > or
  <http://192.168.10.106:8080/>

2. In VM

- git clone <https://github.com/lyrrrr/IRS-PM-2022-IS04FT-GRP-WorkNet>
- pip install the components from requirements.txt
- $cd C:\Users\sampleuser\Downloads\WORKNET\webapp_model$
- Run the server.py with
  $python server.py
- Go to URL using web browser <http://localhost:8080/ > or
  <http://192.168.10.106:8080/>

User Guide

The user guide has been shown in the Project Performance & Validation section.

APPENDIX D

Individual Reports

Individual Report

Li Yurui A0261750J

Personal Contribution

System Architecture Design: After going through so much information online, my teammate and I decided the topic of this project together. To solve both the job recommendation and resume screening problems in nowadays society, I want to finally provide user a website where they can type in information and find the results. For the job seeker, the web needs to provide functions to upload resume, and shows the top recommended job results. For the hr, some text input box need to be provided to type in job description and job title, and then search for the matched resumes. I also designed the quick search pages for both sides to provide the related results just by keywords.

Data Acquisition and Processing: To realize the matching between job and resume, both two dataset must have enough text data for job description and resume string. I did some search to get dataset with enough data for model training and also have enough information for matching. In the preprocessing for data, I tried many methods to clean and normalize each useful column of data. The normal process of NLP preprocess is conducted including lower casing, tokenization, lemmatization, etc. To customize the stop word list for the dataset, the TF-IDF is used to generate the word list of dataset and filter extra stop words.

Model design and implement: I planned to build a model that can complete a bidirectional matching process. Therefore, I chose the cosine similarity calculation method to match vectors and complete recommendation. To control the dimension of feature vector, I developed the Doc2vec model using the python package of gensim.models.doc2vec.Doc2Vec. Besides, I also realize the connection between web and model, using the job or resume submitted by users to match with the vector in dataset.

Lessons Learned

This is my first time to design and implement a model based on NLP knowledge. It is really helpful to fully understand theories learned in class by coding and realizing these models by myself. In this project, I feel more about the importance of data. At the beginning, I spent lots of time on finding the suitable dataset and how to properly preprocess the data I get. It is important to always change the design plan with the dataset I find.

Meanwhile, this project also taught me more in the Doc2Vec model and website development. In class, we had a quick mention of Doc2Vec. In my development, I read its original papers, trying to study deeper into the structure and principle. After the model training, I also responsible to connect the model to website, and complete the data recommendation through model.

Future

This project has given me engineering experience in NLP. I got familiar with how to use package like nltk and genism. to deploy projects and publish our recommendation systems. I also learned how to connect the trained model to the backend of website. Besides, I had a lot of debug experience in this project. In addition to the technical aspects, I also learned a lot about the market, how to research user needs. How to analyze the market situation, and so on.

In the future, I believe this experience will be strongly useful for me. I can use this technology to analyze the text, and conduct preprocess for sentences and words. All in all, with the cooperation between me and my team members, we were able to complete this relatively complex project. And everyone delivered their own module and learned something new, which I believe is meaningful for every member of the group.

Individual Report

Li Fangqing A0261793W

Personal Contribution

In this project, I worked with our group leader LI YURUI and another group member, PRADEEP KUMAR ARUMUGAM, to complete a Bidirectional Job & Resume Matching Systems. I was responsible for part of the project design, business analysis and survey, the overall report writing, and the videos design and recording. Our group leader, LI YURUI, helped me a lot throughout the project and helped me to complete the task under great pressure. I benefited a lot from working with such great people.

Learning Reflection

For a transfer student from, this project is the first AI-related project that I have been fully involved in which including designing, building, testing and optimizing. In terms of system design and coding skills, my group members got much more experience than me. I was actively learning in all parts and trying to understand the knowledge of the models used and understand the architecture of the system.

Firstly, in this project, I have learned setting clear project goal in each project phase is very important in pushing forward the project. To design a product with real business value, I firstly make a survey of the market and evaluate the feasibility of this projrct. And in the practical sections, I learnt how to use the Flask framework for website building and how to built front and back-end by HTML and CSS. Moreover, by understanding models such as Doc2Vec and TF-IDF. I was able to gain a clearer understanding of how to apply my knowledge of NLP for data cleaning, text pre-processing and text feature extraction in practice. By writing the overall content of the report and producing the video, I was able to get a comprehensive view of the overall project structure and design from a holistic perspective. Through this project experience, I have been acutely aware of the importance of lifelong learning and I will keep learning and practicing in my journey in AI fields in order to improve my abilities more practically.

Further Perspective

1. In the further study, I will trying to gain an in-depth understanding of data crawling, data cleaning, clustering and other data processing techniques and apply them to practical examples

2. In the further work, I will trying to optimize the database to make it systematic and more standardized by using how the front-end should interact with the back-end, in conjunction with the database.

3. Through this experience of teamwork, I will continue to give full play to my strengths in my own work and in the promotion of the whole project, and play a better coordinating role in future teamwork projects to help the team achieve their goals in a more efficient and focused manner.

.

Individual Report

Pradeep Kumar Arumugam A0261606J

Individual reflection of project journey

After the "Intelligent Reasoning Systems" module was over, we began to discuss the ideas about all the possible existing problems and the solutions which we can identify and solve using the knowledge provided. Among all possible projects, we chose to build a bi-directional, intelligent, jobs / candidate recommendation system. This was due to the fact, as an ISS student, we were desperately looking for jobs, internships and also for a very simple and compact platform to find it. Finding such platform was our immediate problem and we decided to build one Intelligent system for that. Because, it is more exciting to build a solution for your own problem.

Personal contribution to group project

After deciding the topic, since our team comprised of just three members, we decided to divide tasks for more efficient way in operational point of view. I was responsible for finding the dataset to use for our model and after which is settled, I started working on designing and building the web UI and the framework for operating it as a complete package. Though it was my first time working on design and building a website, by doing extensive research online and reaching out to my team whenever there was a blocker during the building phase, we made a webapp as it can be seen today. With this I got introduce to applications and language like Flux, Webflow, Flutter, dart, html, php and css. Finally, worked on creating the introduction video with the team to make a very simplified, product selling video for customers/users.

What learnt is most useful for you

As a beginner in this AI field, everything we did as part of this project is very useful for me. From building the frontend to fitting the backend and deploying it. Amongst all, I consider the actual model which does the recommendation the most useful, because working with that gave me an insight into what a real-world AI looks like in the backend. Additionally getting a hands-on experience on how to identify a problem, how to identify the methods to solve such problems and how to implement the solution is another valuable experience for me.

How you can apply the knowledge and skills in other situations or your workplaces?

We can use the similar method for various recommendation systems, like product recommendation for shopping customers, movie and series recommendation for subscribed viewers, news and article recommendation for readers and much more. Finally, by doing the web building, with some additional functions and features, we can build even a better interface for all the above-mentioned use cases.