A Document Vectorization Approach To Resume Ranking System(RRS)

by

NORUN NABI
19366001

A project submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
M.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
August 2022

Declaration

It is hereby declared that

1. The project submitted is my own original work while completing degree at Brac University.
2. The project does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The project does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Norun Nabi

19366001

Approval

The project titled \A Document Vectorization Approach to Resume Ranking System" submitted by

1. Norun Nabi (19366001)

Of Summer, 2022 has been accepted as satisfactory in partial ful llment of the requirement for the degree of M.Sc. in Computer Science on September 01, 2022.

Examining Committee:

Supervisor:
(Member)

Dr. Muhammad Iqbal Hossain

Associate Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Amitabha Chakrabarty

Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi

Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

The project submitted is my own original work, which has not been previously published elsewhere. Along with that, the paper is not currently being considered for publication elsewhere.

Abstract

Technology transformed the way how job seekers apply for a job and recruiter's hunting for a precise pick . Now, paper version of resume already become an outdated version of job application method. Electronic resume replaces the old method thanks to its easier access to technology. When it comes to a particular job requirement, screening a relevant resume among thousands is an exhaustive and time consuming recruitment process because the respective HR of an organization must have a proof read the entire resume set to select the right person in the right position, a key decision for any organization. Extracting the semantic meaning from resume is otherwise a daunting task. By making the selection process fast and accurate, organizations could save huge e orts and money. Using state-of-art-technology could be a way out. In the eld of NLP, there are a range of tools to classify documents. Document vectorization technique is a huge popular one among tech-communities. Documents like resumes could be categorized and ranked by applying such techniques and tools. Therefore choosing a most suitable vectorization algorithm is pivotal. It is aimed to build a custom trained model specialized in vocabulary of resume based on frequency based word2vec model such as TF-IDF.

However, to compare between job descriptions and resumes, Cosine-Similarity is considered to be the primary algorithm to nd matching resumes whereas k-nearest neighbor algorithm has been used to group the desired documents. But the limitation comes with using xed vocabulary size. TOPSIS is the most popular among Multi Criteria Decision Making algorithms. Along with vector similarity score, Other parameters like years of experience, university rankings could be normalized to consider for nal ranking score.

Keywords: TF-IDF; Word2Vec; Doc2Vec; Cosine Similarity; Resume Classi cation; Recommendation Systems

Dedication (Optional)

To my family members, whose relentless support have given me the momentum towards completing my MSc program at BRAC University.

Acknowledgement

I would like to express my gratitude to Dr. Muhammad Iqbal Hossain, Associate Professor, Department of Computer Science and Engineering at BRAC University, for his scholarly supervision and inspiration since undertaking the project work till completion. Firstly,He helped unlocking many obstacles confronted during conducting research on nding a proper project architecture and I came to know so many new things. Secondly, with regards to data-set collections to train the model, he gave me lot of ideas. Thirdly, I appreciate the teachers who taught me courses like data science and machine learning throughout the M.Sc classes. Those courses gave me lot of consequential ideas to layout the ground works of the project. Last but not least, I have to acknowledge the comfort and console given by friends and well-wishers throughout the venture.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

API Application Programming Interface

BOW Bag Of Words

CI Continuous Integration

CSS Cascading Stylesheet

GPU Graphics Processing Unit

HRM Human Resource Management

HTML Hypertext Markup Language

MCDA Multi-criteria Decision Analysis

MCDM Multi-criteria Decision-making

NLTK Natural Language TooklKit

RDP Dataset Parser

RRS Resume Ranking System

TF IDF Term Frequency - Inverse Document Frequency

TOPSIS Technique of Order Preference by Similarity to Ideal Solution

UCIRepository University of California, Irvine Repository

UI User Interface

Chapter 1

Introduction

1.1 Motivation

With the pace of industrial economic booming in Bangladesh, Job seekers are growing on an exponential scale than ever. Be it private or public service, organizations receive a ood of resumes even against a single job post; on top of that the scenario is real in Bangladesh due to her unemployment rate. However, to deal with such inundation, the HR department of organizations does not have e ective tools. In most organizations, human intervention is the only way to remove irrelevant and receive relevant resumes for the given job responsibilities. Two major issues that are dominant in the ongoing organizational resume screen processes.

ˆ Firstly, skills veri cation in candidates is most challenging. Unfortunately, it is a common scenario that the recruitment department/agency does not have people with such skills, so they can verify those skills in others.

ˆ Secondly, Sluggish recruitment process is responsible for wasting an organization's valuable time, money and human resources. Because this process demands a careful reading of thousands of resumes. Hence is vulnerable to inaccurate picking.

1.2 Background Study

When thinking about a pragmatic document classi cation project, some search has been carried out on the internet about how these kinds of works are dominating in market and research academia. However,only found some works related to this which is mostly research papers and journals. For example, a machine learning approach for automation of resume recommendation system is the most relevant one [3] is renowned one. In addition to the research paper, A project regarding resume is found categorizing in Brac University project paper - Analyzing CV/Resume using natural language processing[6]. So tried to implement the project gathering ideas and methods from these di erent sources.
In order to analyze the document ranking, most e cient approach now-a-days is with the help of vectorization. So vector modeling is chosen following lot of industry standards and the idea gets con dence by reading di erent machine learning community blog [9].

1.3 Project Scope

This project aims to facilitate faster screening of the most matching pro le from job applicant's resumes with a dashboard. Recruitment department can easily navigate through the dashboard and download desired documents. To serve the user stories, So, This project is assumed to provide a comprehensive solution for Resume screening based on machine learning algorithms. However, steps are a token to move from a simple working version to a more complex-comprehensive version. Firstly there is an attempt to make a workable python model on console output based python programs, and then integrating with an UI module. Hence, the project is broken down into three components primarily; browser based UI dashboard(RRS-UI), Restful API to feed data to dashboard from server(RRS-API), a python based decision making model(RRS-core) , and Data Parser as well.

Figure 1.1: System Components

RRS-UI is based on web frameworks like CSS-HTML and Javascript. As the Fastapi platform is a trendy one, so RRS-API is developed by the platform. VSCode is chosen as IDE to develop all these components. RRS-core is the center of RRS application, which requires signi cant computation, SO It has to develop on a free cloud like Google Colab as it provides free computation power and nearly 50MB+ bandwidth. Same goes for DP, as it requires a terri c internet speed to download GB sized dataset from web sources like Kaggle.

Though there have been assumptions of a comprehensive RRS solution, eventually the assumption resulted in a simpli ed but working version of it. Due to shortcomings of time and resources to build such a complex system, project undergoes for a minimal approach with as much generic version as possible. To be speci c, doing a research base project along with managing full-time job comes with lot of challenges. Another constraint, technologies weren't not familiar beforehand. So development goes an as much as learning process moving on.Every syntax and rules for language and framework. So, it took more time than expected.

Chapter 2

Resume Dataset

2.1 Data Collection

Data is inevitable for any machine learning project. For a predictive model, the more data it could be trained with, the better result the it would infer. Although the importance of data is so high, collecting data is a colossal challenge, particularly noise free data. However there have been a lot of open source platforms that facilitate data for free of cost but not free of distortion. Amidst them some are the most admired data like Google data set Search Engine, Kaggle data sets [10], Amazon Data-sets, UCI machine learning repositories. They contain tons of data set ranging from wide varieties of categories e.g. Image data set, Data set on Disease, data set on sentiment analysis and many more.

Figure 2.1: Data sources example

Searching for purposive text demands to know lot about how text would be analyzed through out the project[7]. As exploring for a project satisfying data-set associated with resumes and its job descriptions, the data found in renowned data repositories are insu cient. Unfortunately, any robust and meaningful dataset could not be managed to feed the project-model except that a tiny amount of resume dataset in repetitive nature and garbage text found in Kaggle. Another di culty is to get the data in the desired

format, for instance CSV le.

Document	Resume text	Experience	Ranking	Education	Category
Resume-1	Full text	4	10	5.5	Data Analyst
Resume-2	Full text	7	2	7.56	Programmer
⋮	⋮	⋮	⋮	⋮	⋮
Resume-N	Full text	0	6	4.9	System Admin

Table 2.1: Structure and number of data set in excel sheet

2.2 Data preparation

Big data empowered the world of machine based decision making. But leverage the e ort of dealing with loads of data. Here, In this venture, the resume-data-set handled from CS v les. Those les are swamped with a set of resumes. Files are organized with a number of columns, mostly in manual intervention. So that, data parser could read them without a major error.

However, Extracting meaningful text from those les is as crucial as expecting an optimized decisive-model. A signi cant e orts of project work is undertaken to develop a less noise training-data which would yield a the better result. However; in order to read bulk data, there is a few choice to mine big data for an ML project. A python library called Panda earned an outstanding performance benchmark score in ML community to mine huge data load into machine learning project.It load data from hard disk into a virtual relational table format, ending up facilitating query like syntax on a large data. After reading data from CSV to Pandas data stream , data are undergone the following text prepossessing steps.

- ˆ Tokenize data
- ˆ Cleaning data
- ˆ Building word dictionary

2.2.1 Tokenize data

Breaking down sentences into token should be done with an utmost importance across the NLP pipeline episode.Owing to its determining weight, the task is performed in more considerate manner. Underpinning the importance, this project uses an extremely popular python library called Gensim for tokenization job. Gensim is an excellent for processing texts like tokenizing among NLP community.

2.2.2 Cleaning Data

It is better to have no data than to have big data with immense noise. But in reality, there is no source of noise free data. De nition of data pollution are thought have a various aspects for instance, data with web tagged, unused numbers, stop words, articles , pronouns etc. This de nition also depends on the eld of speciality where data would be a used for particular target. If it is observed closely, resumes data are keywords surroundings skills and academic terms. Hence, a list of cleaning steps are undertaken, so data could be puri ed as much as possible. However; unlike literature data, dealing with resume data bring about a new challenge, because many of it holds a range of technical terminology not persisting in dictionary.

Word lemmatization also conducted applying NLTK API using WordNet; in spite of undertaking most the of cleaning process is enforced using Gensim API. After lowering all the letters, spaces and newlines are replaced. Before moving ahead for next stage, nal removal of one letter words and chunk of numbers is done.

Figure 2.2: Text pre-processing

2.2.3 Formation of Word Frequency Dictionary

Another python library Numpy was a peerless tool to conduct counting on a large scale data set. So, When necessary to build term frequency dictionary, there is a little choice other than using these two libraries. One of the main targets of data preparation is to construct a data dictionary. A word frequency dictionary that will be used through the project. However, a number of tools in python available to do this. Here in this project Numpy is primarily used when it comes to building a frequency dictionary and processing the words owing to its ease of use.

KEYS	WORD_1	WORD_2	WORD_3	.....	WORD_N
Resume-1	1	4	10	.....	0
Resume-2	9	7	2	.....	5
⋮	⋮	⋮	⋮	⋮	⋮
Resume-N	6	8	3	.....	23

Table 2.2: Bag of words matrix from resume data set.

Chapter 3

Product Description

RRS comes up with an interactive dashboard on conventional browsers. So it does not need a special setup for client PCs to browse the system. As it provides a web based solution, it could be accessed from anywhere and any device. The dashboard helps navigating across the resumes and reading a particular one and downloading it to print. Some dashboards are summary to represent overall ranking in other data set. Scroll-able details of bar charts, which could be sorted based on attributes provided.

There is a page for con guration purposes. A machine learning project based on different parameters. In order to make the project more controllable by HRM personnel, the con guration facilities will make it more exible and generic for future maintenance of the site.

3.1 Methodology Used

Due to work nature, Iterative methodology was adopted throughout the project development. It took a couple of iterations to make a nal release, and in each iteration a working version with some feature and x is used to be nalized.

Figure 3.1: Iterative model

Planning took a considerable time on how the project gets its shape mostly depends technology and time. At the beginning stage, it was decided to develop the project with core building model only; but later on a major shift in planning.

3.2 System Design

Identifying modules and components are crucial to separate the concern of business logic. This project has several components as below

- HRM-Dashboard
- RRS-Core
- RRS-API

Figure 3.2: RRS architectural diagram

3.3 Process ow diagram

Recognizing relationship among major components in an output based system plays an important role to build an e ective system.It helps delimiting the project scope and beginning system development task after post analysis of Resume Rannking System(RRS).

Considering that identi cation of critical system processes is crucial when developing an automation system. Resume Ranking System gured out some of its major components and interactions are depicted in order to visualize to have a birds eye view of data communication from process to process.

Figure 3.3: Process ow diagram for background process

Brainstorm produces two major process ow diagram (PFD). One for background processes and other for foreground processes. Background process mostly deal with data journey of data. The journey starts with submitting resume by a job seekers for a job posted by HR personnel of an organization. A data collector is sit to pull data from outside server to RRS server. At the time of data collection, data is scrutinized whether the document is compatible with system or not. then data bring into to RRS database. Once readable data is uploaded, there are a couple of steps undertaken like data cleaning , tokenizing, building frequency matrix etc.

Figure 3.4: Process ow diagram for foreground process

After background processes are successfully executed, a vectorized document is stored in le system. Vectorized document is not in its nal point of data ow. A lot of data communication needs to do with these data because users of the system may not aware of how document vector works. So this vectors should be expressed in more meaningfully way.

Considering that in mind, a distributed architecture is endorsed to manage a huge number of vector formatted data to its graphical view. HRM portal would be primarily responsible to support such a dashboard. This portal would fetch data from di erent web services. So user would be able to make criteria based searching.

3.4 Sequence Diagram

Di erent actors need to interact with Resume Ranking System in order to make those processes moving. Some are humans' actor for example Job candidate and HR personnel and some are system initiated for example scheduler or system operator could initiate the program to get the trained result. Though sequence of the system with human interaction happens with minimal exchange, communications with database and NLP components are inevitable. Below is presented a sequence diagram to have an abstract glimpse of the possible system-human interaction.

Figure 3.5: Actor-system sequence diagram

The RRS cores receive data from organization Job portal eventually stored in database. This an abstract view of NLP components where RRS-core,RRS-API and RRS-web reside interact between system actors and database. System should calculate document vector for given batch of dataset other than starting for each resume. Because of xed size of vocabulary, a renewed list of vector has to reload to start over the vectorization process.

3.5 Data ow diagram

The primary element of a NLP project is data. So in order to have a general view of how data is owing in di erent phase of the project carries a lot of signi cance. Following is given a data ow diagram.

Figure 3.6: Data ow diagram

Final version comes with the idea that, project will have interactive HR dashboard communicating with model via a service components.Also development environment has a great impact on product development lifecycle. First version comes with working in personal PC. But limitation with time constraint in personal computer to train data model leads to moving to free cloud environment like Google colab on later versions. When it comes to deployment, rstly deployed in Heroku through Github as Continuous Integration fashion. But when trial version is over, service gets disrupted. Therefore services deployed in the local pc to demonstrate.

Chapter 4

Development and Deployment

A micro service approach is taken when deploying components. Each three RRS-core, RRS-API, HRM-Dahsboard have required a diverse development environment.
Primary tools linked to the development of RRS-core. In terms of libraries here is the list from pip3 which had been frizzed from project. Most development done with VSCode Editor, because it has rich plugins of di erent development environment like Python development plugins. Numpy for creating frequency dictionaries and many others dependent libraries to include.

4.1 Development Tools and frameworks

Most development is done with VSCode editor, because it has rich plugins of di erent development environments like Python dev plugins. Numpy for creating frequency dictionaries and many others dependent libraries to include.

In terms of libraries here is the list from pip3 which had been frozen from the project

Figure 4.1: The list of used tools and libraries

4.2 Deployment through CI

Its is tempting to deploying a project through CI pipeline, hence It has been done in previous version of the development. While developing in a local machine, after commit and push to Github, the deployment server automatically pulls data from the Git server

and makes a successful deployment. If automatic deployment from a git push is not successful, then deployment is rejected and rollback to the previous successful commit. CI (Continuous Integration) approach was set up to deploy the project.

Figure 4.2: Continuous Integration

4.3 RRS-Core Deployment

Deploying a cloud application in local machine becomes a super easy approach like no other. Python supports creating a virtual environment with virtualenv library. Virtualenv leaves a number of steps to prepare a virtual environment for python and deploying an application on it as follows.

Figure 4.3: Instructions applied for local python development

4.4 RRS-API DevOps

RRS has a Restful client. The RRS system used its front-end as react application. The client-end is feed with data using Restful API. As Restful API is admired among developer community for its simplicity and easy-to-use approach, this project adopted Restful API for data transmission among client and server. Open API is embedded in FAST-API framework, so APIS can be accessible and tested from anywhere. Development tools mostly integrated as follows.

- Fast-API
- Swagger
- Python

- Nympy

However, APIs could be found in the following URL base-url/docs

Figure 4.4: RRS-API List

4.5 HRM-Dashboard

A lightweight development strategy is chosen for UI development, Unlike RRS-core, there is a intention of putting less e ort on this part. Easy to run approach is taken with the advantage of VScode that provides server can run on the y almost instantly. Therefore no con guration di culties to get an application server ready to deploy. Web technologies are best suited to work on this part like

- HTML,CSS
- Javascript, JQuery
- VsCode On the y Server

This web application receives data through RRS-API, A sample request response is given below. Response data helps the app navigate and forming HRM-Dashboard.

Figure 4.5: RRS-API sample request response

Chapter 5

Results

Outcome of the project is a series of vectors. TF-IDF vectors one of the most popular doc2vec representation. Unlike vectors from occurrence matrix, it yields more meaningful results close to semantic. Fixed number of words needs to be determine beforehand making those document vector, one the limitation of vector representation.

5.1 Document Vector

Deriving document vector needed lot of ideas for example, the vector could be sentence vector or document vector. The ideas are adopted from multiple sources [4]. Finally document is represented with vector other than sentence or word vector.

Words	resume _1	resume _2	job _1	job _2
machine	0.198042	0.000000	0.000000	0.000000
science	0.099021	0.000000	0.138629	0.000000
statistics	0.198042	0.000000	0.000000	0.000000
python	0.000000	0.000000	0.000000	0.000000
data	0.099021	0.000000	0.138629	0.000000
spring	0.000000	0.198042	0.000000	0.115525
development	0.000000	0.000000	0.138629	0.115525
engineer	0.000000	0.000000	0.000000	0.231049
analysis	0.198042	0.000000	0.000000	0.000000

Table 5.1: Bag of words matrix from resume data set.

5.2 Similarity Approach

Cosine similarity algorithm is the most popular technique when it comes to comparing two vector[8]. Similarity algorithm other than cosine like k-means, k-nearest algorithms are not only computationally expensive but also have the limitations compared to it.

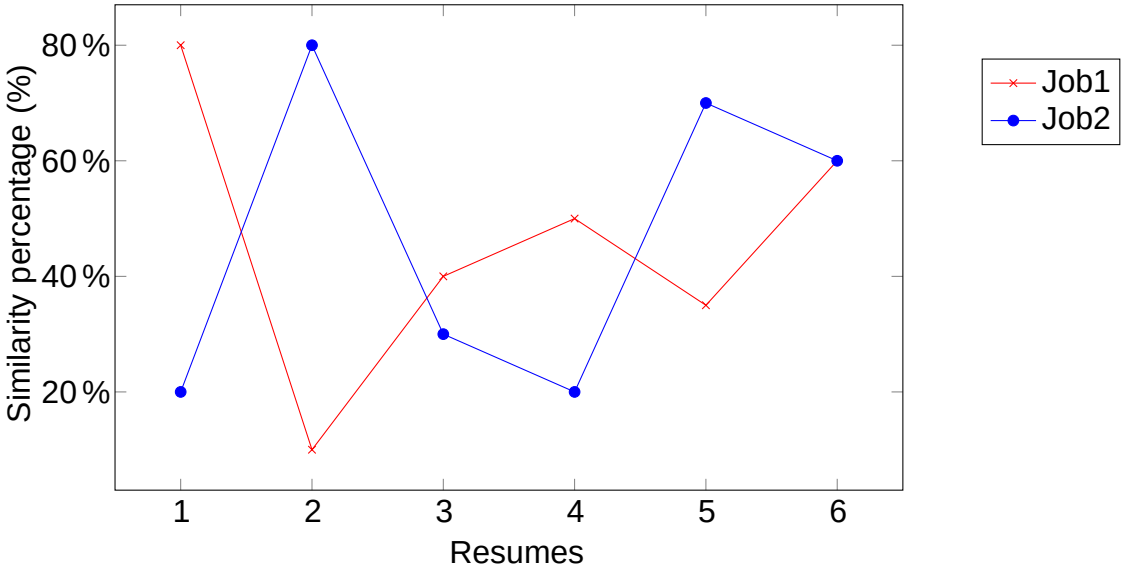

Table 5.2: An example of documents similarity with job description

5.3 Weight Distribution

Ranking scores are distributed based on a weight matrix [1] computed applying the TOPSIS algorithm[5]. In other words, A MCDM matrix[2] is built for individual jobs(e.g. job description) out of prede ned weight values to each criteria.

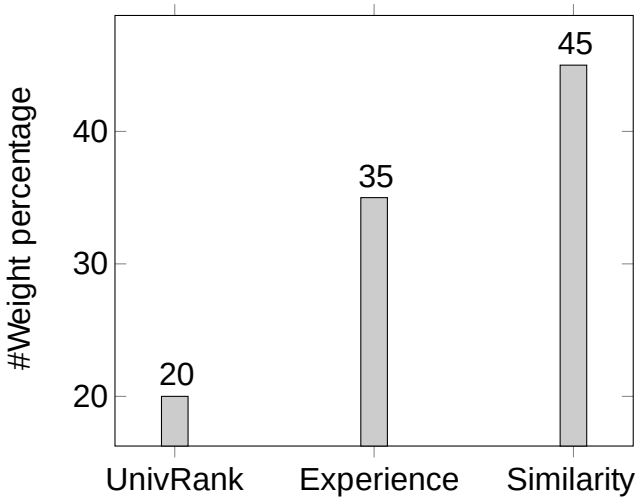

Table 5.3: Scale of importance of criteria

Then criteria vectors are normalized using sum of square method.
The matrix $(x_{ij})_{m\ n}$ is then normalized to form the matrix.
$R = (r_{ij})_{m\ n}$, using normalized method.

$$(r_{ij}) = \frac{x_{ij}}{\sqrt{\sum_{k=1}^{m} x_{kj}^2}}; \qquad i = 1; 2:::::m; \qquad j = 1; 2; :::::; n$$

Then this normalized vector is multiplied by each criteria weight value as above table. Which is called weighted-normalized decision matrix.For each resume, Euclidian distance is calculated between Ideal best (Dib) and Ideal worst (Diw) values selected for individual criteria vectors.
Now, performance score is determined from this distance vector using the following formula

$s_{iw} = d_{iw} = (d_{iw} + d_{ib}); \qquad 0 \quad s_{iw} \quad 1 \qquad i = 1; 2; ::::; m$
$s_{iw} = 1$, if and only if the alternative solution has the best solution;and
$s_{iw} = 0$, if and only if the alternative solution has the worst condition.

Chapter 6

Use Cases

Though most of this project work is intended for data visualization purposes on application/console log, this project extended its use cases for a user experience from web application. Where users like HR personnel could make an educated guess from a big resume data set by speculating an abstract view of all resumes at a glance.

Firstly, personnel from the Human Resource Department will enjoy an abstract view of having multiple charts from a long list of data sets. Secondly, use cases like visualizing the proportion of categories of resume streaming in the resume vault. However, downloading the desired resume is an additional feature incorporated in this program.

6.1 Visualize The Volume

With a view to categorizing the data-set, the project does not need to adopt any complex algorithm. Simple percentage from all data-set is calculated from the category column of the resume. Hence shown in the diagram

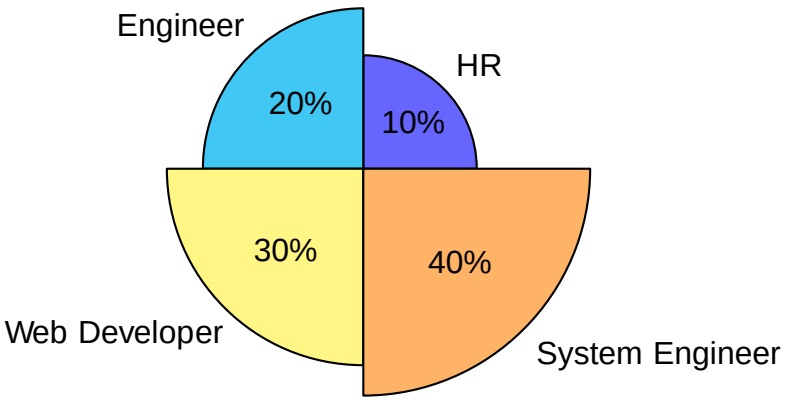

Table 6.1: Parentage of category of resume received

6.2 Ranked illustration

Following diagram showing data sorted by an overall rank. However this ranking could be customized by sorting any given attribute. Where, overall score is calculated using the TOPSIS algorithm, and similarity score is estimated by means of cosine-similarity of vector space. There shown an years attribute which is put in the diagram linearly from resume without any normalization, and this way other feature like university ranking, number of degree, number of certi cation, organizational reputation score where candidate has most of her professional experiences etc could be considered with the ranking criteria

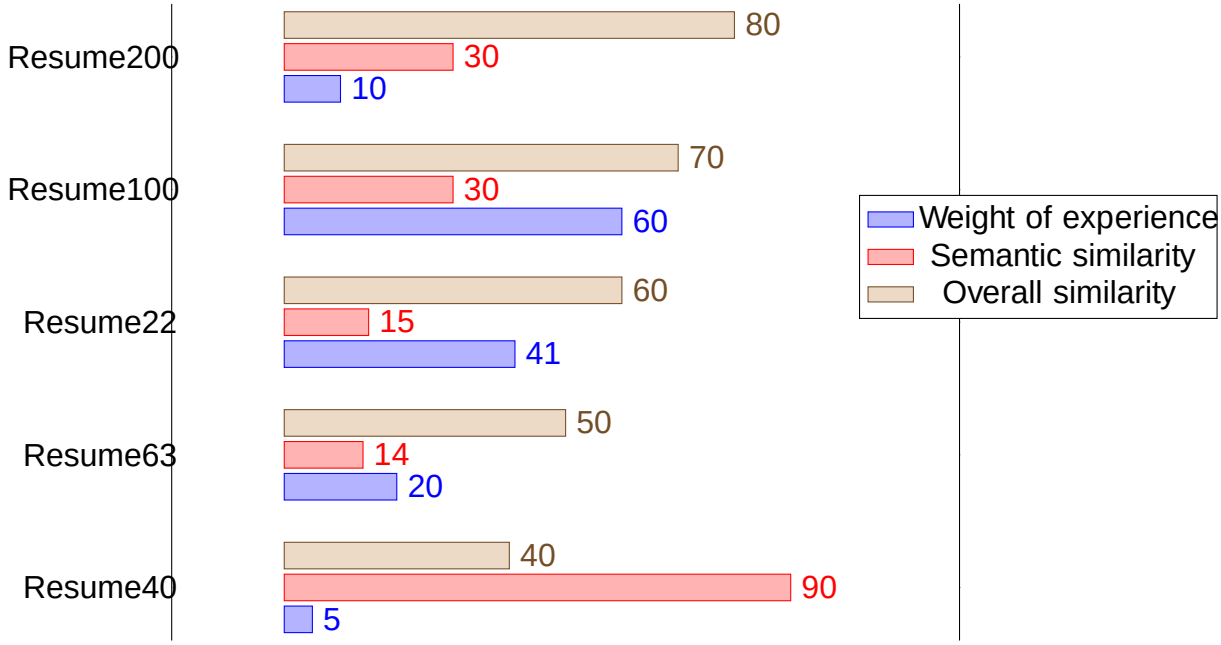

Table 6.2: Resume ordered by overall ranked score

6.3 Download Desired Resume

Resume could be downloaded by clicking the link on the bar. As the user clicks on the bar chart, it will open the particular resume in a new tab of the browser. So users could download the resume for the next phase of the recruitment process after a proof reading of every detail by comparing and contrasting with the job description attached on the job-circular.

Figure 6.1: Preview of downloaded resume

Chapter 7

Conclusion

Working on such a project has been a challenging journey throughout the development. An abstract level of document classi cation problem is attempted to solve but everything does not happen as expected during planning. Limitations are there at di erent levels. In terms of deployment, it has to use an open source platform. So an open source platform has its own pros and cons to deploy a project in a live server. UI could be a more attractive one. Considering all, future work will cover the limitations and projectile a comprehensive solution classifying right documents.

7.1 Limitations

Firstly, to train and build a machine learning model, a huge computational power is needed to process the data. As developing the project in personal laptop which is low con guration compared to GPU embedded machines, the model could not be trained with complex algorithms and big datasets. So Workspace limitation could be a bottleneck for ultimate optimization of project performance. Secondly, Project con guration could be more generic and exible so that each dataset might be processed. Currently, Only CSV le format is supported.

7.2 Future work

By minimizing the limitation of the project, future work could envision a better Resume Ranking System. As classifying documents comes with a considerable number of criteria and particularly when it comes to resume, this has the challenge of dealing with multi criteria. This project incorporated some criteria of resume like "years-of-experience", "skills-relevancy-score" and "Institution-ranking". There are a lot of other criteria that could be included in future to constitute a better resume screening system. For example, age of the candidate, academic grading, and reputation score of institutions where candidates worked earlier, national and foreign degree, publications, certi cations etc. So a comprehensive MCDM method could lead to more sophisticated and desired RRS. However, this project used a frequency based vectorization to classify the documents. Further improvements could be done using other types of algorithm vectorization algorithms. The UI dashboard for HRM could be deployed in BRAC University website.

Bibliography

[1] H. Byun and K. Lee, \A decision support system for the selection of a rapid prototyping process using the modi ed topsis method,"The International Journal of Advanced Manufacturing Technology, vol. 26, no. 11, pp. 1338{1347, 2005.

[2] R. V. Rao, Decision making in the manufacturing environment: using graph theory and fuzzy multiple attribute decision making methods. Springer, 2007, vol. 2.

[3] X. Yi, J. Allan, and W. B. Croft, \Matching resumes and jobs based on relevance models," in Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, 2007, pp. 809{810.

[4] Q. Le and T. Mikolov, \Distributed representations of sentences and documents," in International conference on machine learning, PMLR, 2014, pp. 1188{1196.

[5] R. Karim, C. Karmaker, et al., \Machine selection by ahp and topsis methods," American Journal of Industrial Engineering, vol. 4, no. 1, pp. 7{13, 2016.

[6] M. S. Reza Md. Tanzim Zaman, Analyzing cv/resume using natural language processing and machine learning, https://www.oreilly.com/library/view/applied-text-analysis/9781491963036/ch04.html, 2017.

[7] T. O. Benjamin Bengfort Rebecca Bilbro, Applied text analysis with python, https://www.oreilly.com/library/view/applied-text-analysis/9781491963036/ch04.html, 2018.

[8] S. Gupta, \Jaccard index and cosine similarity | where you should use what, pros and cons of each.," May 2018. [Online]. Available: https://towardsdatascience.com/overview-of-text-similarity-metrics-3397c4601f50.

[9] J. Alammar, \Visualizing machine learning one concept at a time.," 2019. eprint: http://jalammar.github.io/illustrated-word2vec/. [Online]. Available: http://jalammar.github.io/illustrated-word2vec/.

[10] S. BHAWAL, \Resume dataset,"Kaggle Dataset, vol. 1, pp. 1{1, Dec. 2021. [Online]. Available: https://www.kaggle.com/datasets/snehaanbhawal/resume-dataset?resource=download.

Appendix A: How to prepare the development environment

Devops on Colab

First couple of development phases are conducted on open source Google Colab. As follows:

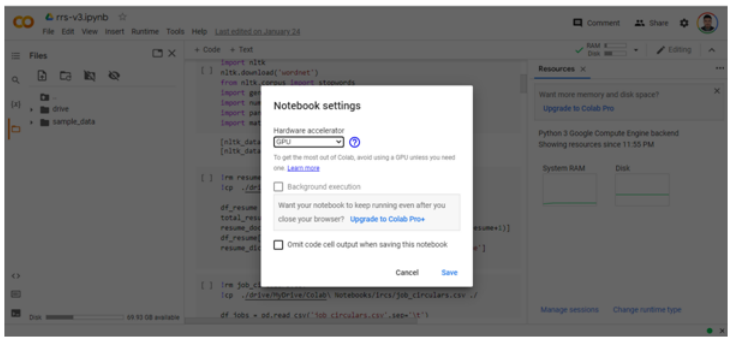

Figure 7.1: Google Colab workspace

Install Python in Windows PC

1. Download Pytnon from https://www.python.org/ftp/python/3.10.1/python-3.10.1-amd64.exe

2. Double click the file **python-3.10.1-amd64.exe** to begin installation and proceed default instructions till the installation is finished.

3. To check if installation is successful, Open windows command prompt and type the following:

```
$ python --version
> Python 3.10.1
```

4. However,You will find the default installation path as follows

```
C:\Python310
```

RRS-API project structures

1. Create a directory called "RRS-FASTAPI"
2. Under the RRS-FASTAPI, create file main.py
3. Under the RRS-FASTAPI, create a directory called "resources" where trained vector model are kept
4. Initialize FAST api application inside main.py and define necessary API interface.
5. Successful execution results in an open up swagger UI with exposed APIs

```python
app = FastAPI()


app.add_middleware(
    CORSMiddleware,
    allow_origins=['*'],
    allow_credentials=True,
    allow_methods=['*'],
    allow_headers=['*'],
)

pkl_file = open("./resources/resume_list.pkl","rb")
resume_list = pickle.load(pkl_file)
pkl_file.close()
total_resume = len(resume_list)

#Write APIS
@app.get('/')
async def root():
    return {'hello' : 'world'}
```

Figure 7.2: Fast API

Figure 7.3: API Document

Figure 7.4: API request response sample

RRS-WEB project structure

1. Under the workspace, create a directory named "RRS-WEB"
2. Under RRS-WEB, create directories named CSS, JS, Images.
3. Under RRS-WEB, create index.html and other realted html files.
4. Run the application on VSCode live server plugin.

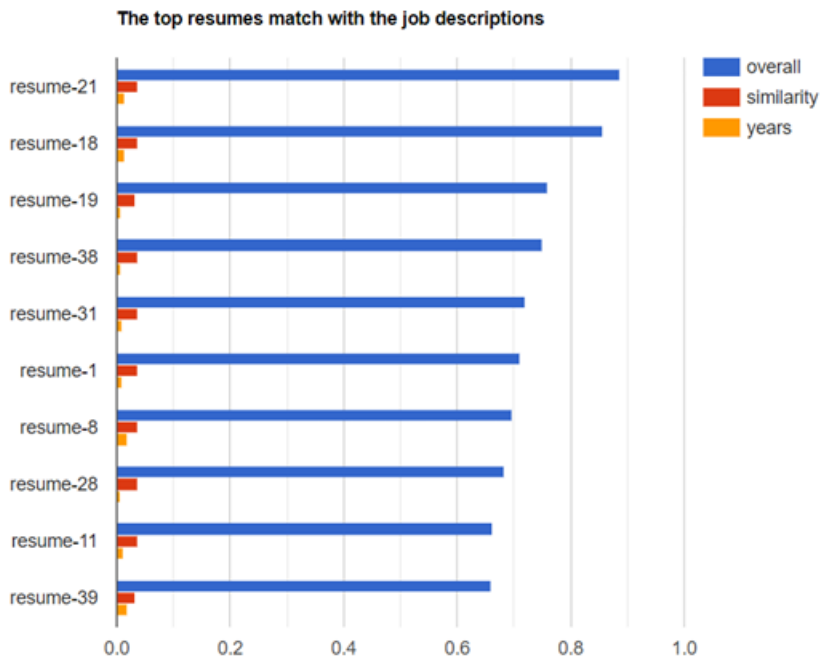

Figure 7.5: Rank visualization

Create RRS-Core project structure

1. Under the workspace, create a directory named "RRS-Core"
2. Under RRS-Core, create a file named "main.py" and "utils.py"
3. import following packages inside main.y

4. Under RRS-Core, create directory named "dataset", "resources"
5. Put resumes.csv and job-circular.csv under dataset and vector model data under resources directory.
6. Start the coding inside main.py
7. To run the project type as follows:

```
$python main.y
```

8. Successful execution should be end up saving vector model files under resources directory.

Run RRS-Core and RRS-FASTAPI on virtual environment

Executing a Python project with the help of virtual environment made the development pace even faster and easier. Unlike library dependency version chaos with global environment, The virtual environment helps creating a project specific snapshot for library dependencies. RRS-Core and RRS-API has followed the following instruction set to prepare an virtual environment.

Figure 7.6: Instructions preparing python virtual environment

Appendix B: Sample data

Sample Dataset

Data are download in CSV format which holds multiple columns as attribute to resumes and earch row represents an individual resume.

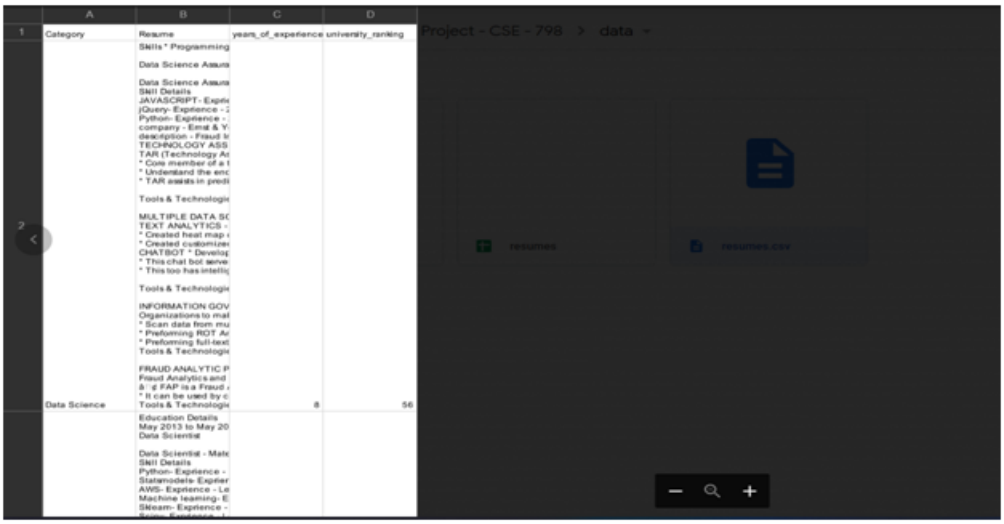

Figure 7.7: RRS Sample Data

Cleaning Steps

- NLTK is used to clean text
- Remove non-English text
- All text converting to lowercase
- Remove stop words
- remove articles and pronouns

```
%%writefile utils.py --append

def get_tokens(doc):
  tokens = []
  clean_text = doc.replace('\n', ' ').replace('\r', '')
  clean_text = remove_non_english_words(clean_text)
  clean_text = clean_text.lower()
  clean_text = remove_stop_words(clean_text)
  clean_text = remove_one_length_words(clean_text)
  clean_text = lematize_sentence(clean_text)
  clean_text = remove_stop_words(clean_text)
  tokens = tokens + list(tokenize(clean_text))
  return tokens
```

Figure 7.8: Sample code for tokenizing text

Word Frequency Dictionary from log

The word frequency dictionary consists of resume-name as keys and arrays as values. The value holds two arrays. Of the arrays, first one represents unique words list from a given resume and second arrays is mapped with word occurrence number respectively. here is a sample.

```
%%writefile utils.py --append

def get_document_tokens(doc_list):
  doc_tokens = []
  for idx,doc in enumerate(doc_list):
    tokens = get_tokens(doc)
    doc_tokens.append(tokens)
  return doc_tokens
```

```
Appending to utils.py
```

```
[ ] %%writefile utils.py --append

def get_document_frequency_dictionary(doc_dict):
  dictionary = {}

  for key in doc_dict.keys():
    doc_text = doc_dict.get(key,'')
    doc_tokens = get_tokens(doc_text)
    token,counts = np.unique(doc_tokens,return_counts=True)
    #dictionary[index] = dict(zip(token,counts))
    dictionary[key] = token,counts
  return dictionary
```

Figure 7.9: Sample code for building word frequency dictionary

Bag of words

1. Using Panda library, a 2D matrix is initialized with zero where index column from vocabulary list and remaining columns from document name.
2. The location(words,resume no.) of the array is updated with word count through iteration on frequency dictionary
3. There is also a binary bag of words prepared to check if a particular word exits or not on a resume.