CSCI 870

WEBCON

Know your future job

Sonam Dhadiwal
Department of Computer Science
New York Institute of Technology
New York, U.S.A
sdhadiwa@nyit.edu
NYIT ID: 112397

Ajayraj Hulikere
Department of Computer Science
New York Institute of Technology
New York, U.S.A
ahuliker@nyit.edu
NYIT ID: 1177123

Under the guidance of

Dr. Wenjia Li,

Department of Computer Science

NYIT New York Institute of Technology

Abstract —

In recent years we have witnessed a rapid and continuous use of job portals. However it is difficult to find a relevant job that matches with the profile of the user and suggest the most relevant job. Therefore having a good web application which provides such a functionality has become the requirement.
To the best of our knowledge this paper presents systematic study of our job portal recommending a job and the vectors and statistical part being used in the application. The various mathematical calculations required to match the document along with natural language processing are being used in the development of this application.
This application requires you to enter the job title you want to search for along with city and state. It allows you to upload your curriculum vitae (CV) so that it can match with the job descriptions. WEBCON does text mining and gives you the output with the top job matches from all the jobs present on job portal as per user's search criteria.

Keywords —
Text Mining, tfidf Vectorizer, cosine similarity, natural language processing.

I. INTRODUCTION

Online recruitment has revolutionized the recruitment industry that benefits both employer and job seekers and ultimately accelerates the recruitment process. Thus online recruitment is an era of quick hiring where employers can get the potential talent quickly. Online recruitment today is indispensable for recruiters and employers who use job portals as the primary source for headhunting, although this is a traditional hiring method but still it works, so job portals and career sites are the two edges of the recruitment equalizer. The penetration of the internet in our society has crossed the geographical boundaries, thus mobilize the recruiting process. The quest for hunting top talent is a quest to boost regional economy in reality.
WebCon allows a candidate to search a company/organization that matches his/her job requirements. This match is made by comparing and matching the resume keywords with job description keywords.
Correct recommendations are much more useful for a candidate as it reduces the time and hassle one has while finding a job. The application is developed using Python.

II CONTENT BASED FILTERING

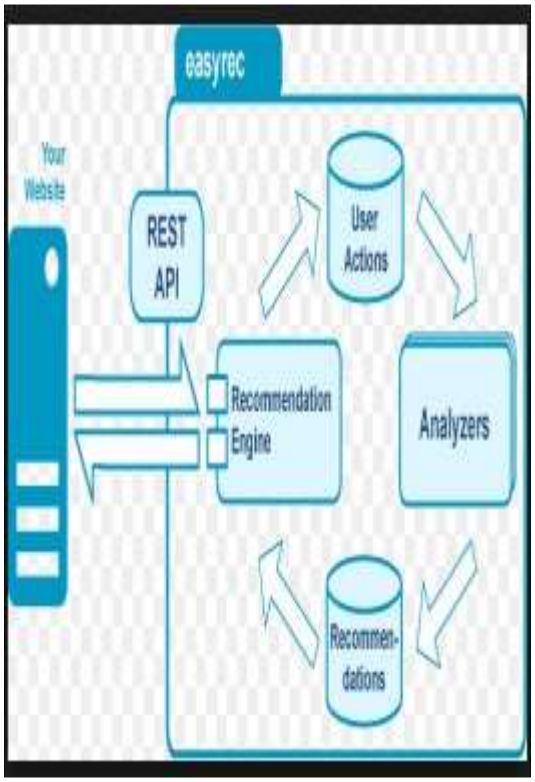


Fig 2.1: Content Based Recommendation System [1]

Content based recommendation system is applied in context with recommendation and prediction preference. This paper uses this system to match the resume with the job description. Content Based Filtering algorithms try to recommend the results which are similar to the previous search results.

The figure above shows how the content based recommendation system works. This system works with the data provided by user. User inputs the data which can be any file. Then based on the given data user profile is generated. Using this generated profile, suggestions for the jobs are made. As the user takes some actions the system becomes more and more accurate.

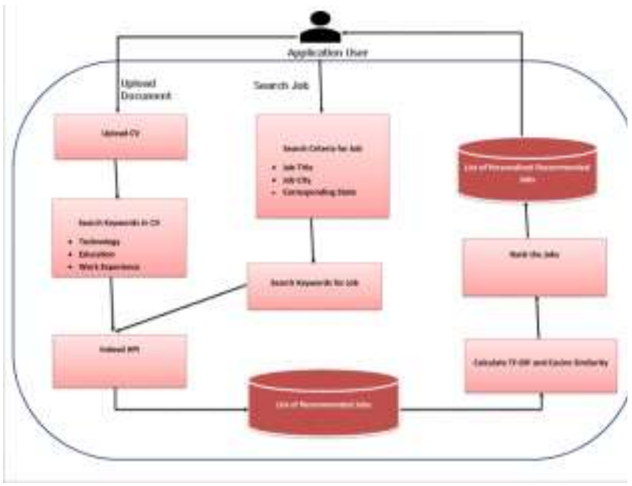


Fig. 2.2: Schematic Diagram

The above diagram explains the schema of the project.

III TF-IDF VECTORIZER

Term frequency (**TF**) is the frequency of a word in a document and inverse document frequency (**IDF**) is used to find inverse frequency of a word in a document. Term frequency counts the number of words in the document and stores the count.

For instance "The sun rises in the east." It stores the count of each word.{the:2, sun:1, rises:1, in:1, east:1}. In this example though the count of word *the* is more than *sun*, its importance is less compared to the word *sun* and so in order to provide proper weightage to each word we need to find inverse document frequency.

In order to dampen the effect of high frequency words we use log calculation in IDF.

Suppose a document has 100 words where the word *cat* appears 3 times.

TF(cat) = 3/100 = 0.03

Now suppose we have such 10 million documents (which are very large) and the word cat is appearing in 1000 documents.

Idf(cat) = log(10,000,000 / 1,000) = 4.

Tf-idf(cat) = tf*idf

= 0.03 * 4

= 0.12

The basic idea of the algorithm is to represent each document d as a vector in vector space so that document with similar content have similar vectors. Each dimension of the vector space represents a word selected by feature selection process.

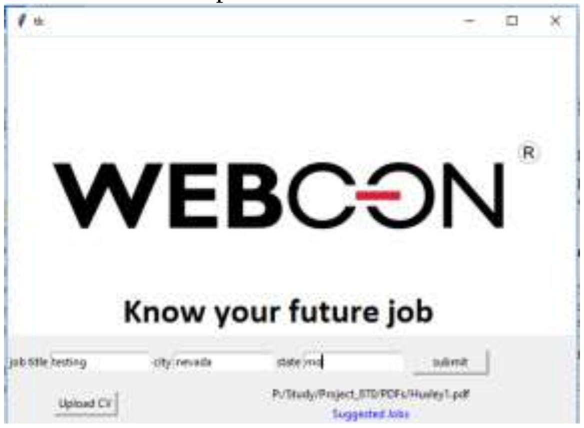
The values of the vector elements $d^{(i)}$ for a document d is calculated as acombination of statistics TF(w,d) and DF(w). The term frequency TF(w,d) is the number of times word 'w' occurs in document d. The document frequency DF(w) is the number of documents in which the word –w occurs atleast once. The inverse document frequency IDF(w) can be calculated from the document frequency.

$$\text{IDF}(w) = \log\left(\frac{|D|}{DF(w)}\right)$$

|D| is the total number of documents . The inverse document frequency of a word is low if it occurs in many documents and is highest if the word occurs in only one.[4]

$d^{(i)} = \text{TF}(w_i,d) * \text{IDF}(w_i)$

Consider an example:



job title testing
city nevada
state mo
P:/Study/Project_STD/PDFs/Huxley1.pdf

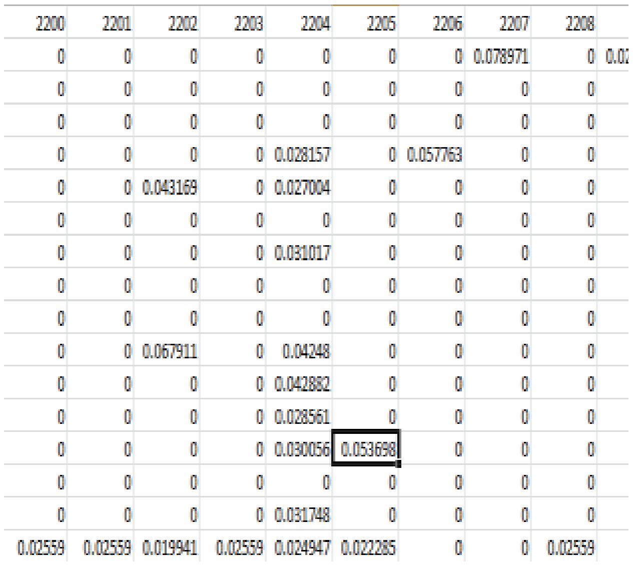
Fig:3.1 Application Window

Job title: testing

City: Nevada

State : mo

Result:



2200	2201	2202	2203	2204	2205	2206	2207	2208	
0	0	0	0	0	0	0	0.078971	0	0.02
0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	
0	0	0	0	0.028157	0	0.057763	0	0	
0	0	0.043169	0	0.027004	0	0	0	0	
0	0	0	0	0	0	0	0	0	
0	0	0	0	0.031017	0	0	0	0	
0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	
0	0	0.067911	0	0.04248	0	0	0	0	
0	0	0	0	0.042882	0	0	0	0	
0	0	0	0	0.028561	0	0	0	0	
0	0	0	0	0.030056	0.053698	0	0	0	
0	0	0	0	0	0	0	0	0	
0	0	0	0	0.031748	0	0	0	0	
0.02559	0.02559	0.019941	0.02559	0.024947	0.022285	0	0	0.02559	

Fig: 3.2 tfidf / sparse matrix

The highlighted part shows the tfidf of the word *testing*.
Using vector, sparse matrix was generated as shown in figure above. Row indicates file name and column indicates the feature of the tfidf matrix. Corresponding to the feature (here testing) we calculate term frequency as:

$$\text{TF} = \frac{(\text{total count of word testing})/}{(\text{total number of words in that document})} = \frac{10}{55}$$

Here testing appears 10 times collectively in all the documents.
As we are calculating the value with respect to document 12.txt, the total number of words in 12.txt is 55.

Now, let's calculate inverse document frequency:

$$\text{IDF} = \log_e\left(\frac{total\ number\ of\ testing\ files}{1+no.\ of\ files\ containing\ word\ testing}\right)$$
$$= \log_e\left(\frac{15}{11}\right)$$
$$= 0.310$$

Therefore,
$$\text{TF*IDF} = (10/55) * (0.310)$$
$$= 0.056$$

The above value is approximate to the tfidf value shown in figure.

Cosine similarity

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The cos of 0° is 1, and it is less than 1 for any other angle.
The Cosine similarity measure is mostly used in document similarity [28,33] and is defined as $Cosine(x,y) = \frac{\sum_{i=1}^{n} x_i y_i}{\|x\|_2 \|y\|_2}$, where ||y||2 is the Euclidean norm of vector y = (y1, y2, …, yn) defined as $\|y\|_2 = \sqrt{y_1^2 + y_2^2 + \ldots + y_n^2}$. The Cosine measure is invariant to rotation but is variant to linear transformations. It is also independent of vector length.[5]
Cosine similarity in sklearn computes the similarity between two documents or two data sets.
The parameters of this function is ndarray or sparse matrix. In this project cosine similarity is used to match two files one with resume and another gathered from job portal.

Cosine similarity calculates the distance using Euclidean distance.

The cosine distance is equivalent to the half the squared Euclidean distance if each sample is normalized to unit norm.[2]

In mathematics perspective, Cosine similarity is perfect. However, if we check it in text mining perspective, it may not always be reasonable[5].

Given two dimension vectors , the cosine similarity between them is calculated as follows:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Considering the above example from tfidf and comparing the documents

Cos(0.056) = 0.9984≈ 1

This shows that document number 12 is more similar to the uploaded resume.

Putting the value

IV Flow of the Application

The following steps describe the flow of the application.

1. Start.
2. When user launches the application he/she needs to enter the search criteria like job title, city/ zip code, state, upload the resume.
3. Click the submit button.
4. Application calculates tf-idf and then cosine similarity. Based on cosine similarity, ranking is done and job results are displayed to the user.
5. Stop.

Below is the pictorial representation of the flow.

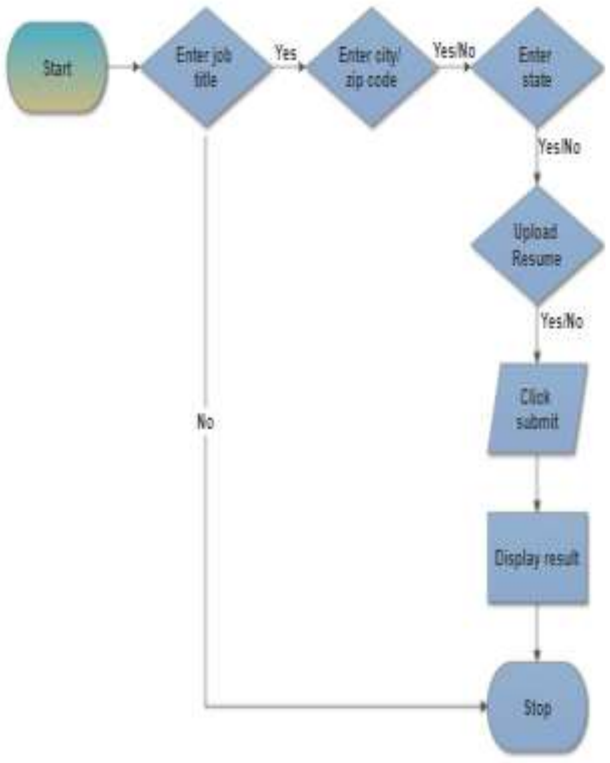


Fig. 4.1 Flowchart

V APPLICATION

Launch the application.

Following window appears.

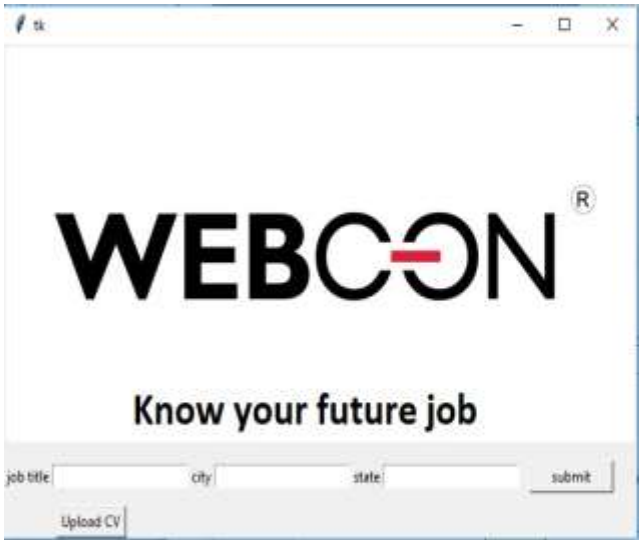
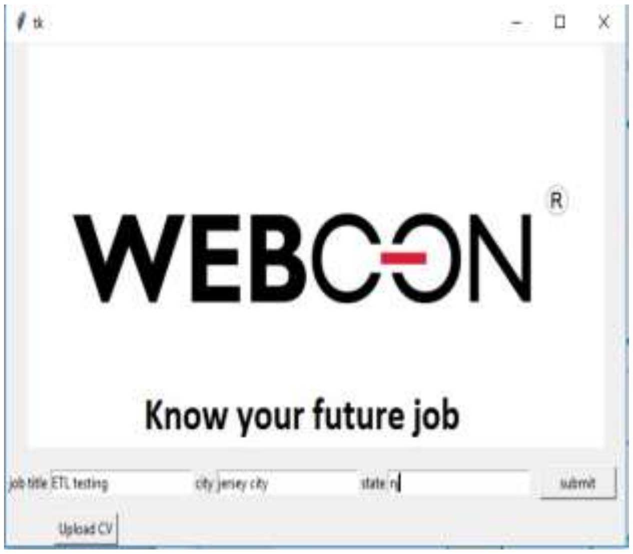


Fig 5.1.Application Window

This is the starting point of application where user can enter job title, city, state etc.

2. Enter the job title, city, state

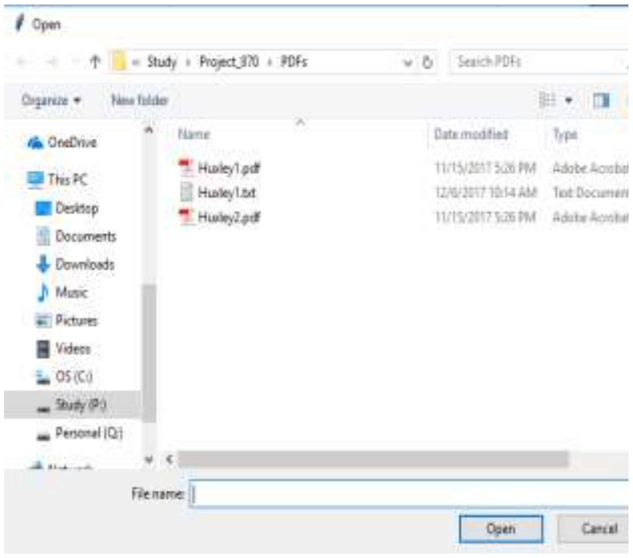


job title ETL testing
city jersey city
state nj

Fig. 5.2 Search criteria window

In the above GUI you can enter any job title. Here for example I am searching for ETL testing job. You can choose any city and state. Remember city should match the state it belongs to or it will not display any results. Instead of city you can also enter the zip code if you want.

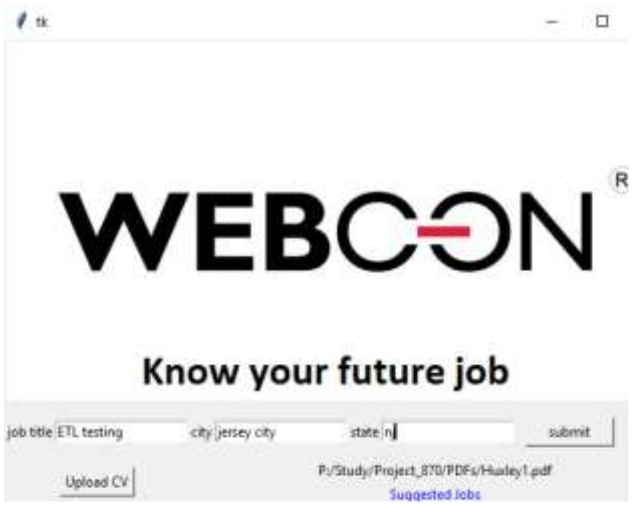
4. Click on upload CV button.



Name	Date modified	Type
Huxley1.pdf	11/15/2017 5:26 PM	Adobe Acrobat
Huxley1.txt	12/6/2017 10:14 AM	Text Document
Huxley2.pdf	11/15/2017 5:26 PM	Adobe Acrobat

Fig. 5.3 Upload resume window

It opens a location from where you can select the file. You can choose any path and select the file wherever it is stored. Remember to select your resume in pdf format (Only pdf format supported in Webcon).

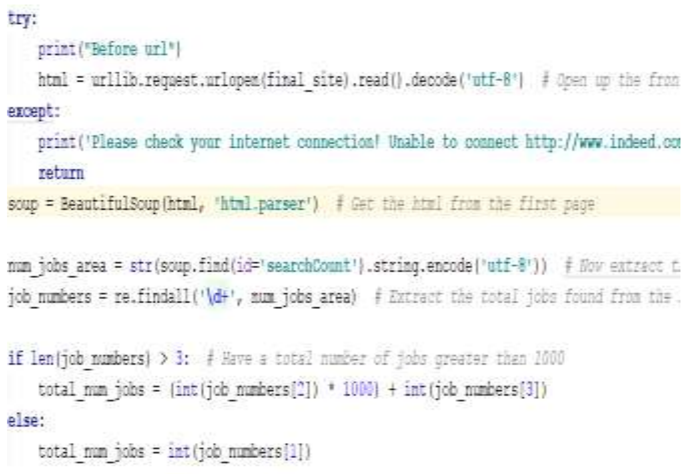


job title ETL testing   city jersey city   state nj

P:/Study/Project_870/PDFs/Huxley1.pdf

Fig. 5.4 Application window with results link

5. Click on Submit button



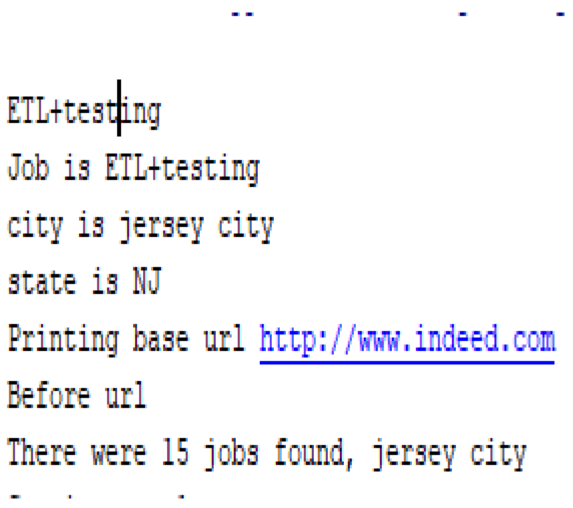
```
try:
    print("Before url")
    html = urllib.request.urlopen(final_site).read().decode('utf-8')  # Open up the fron
except:
    print('Please check your internet connection! Unable to connect http://www.indeed.co
    return
soup = BeautifulSoup(html, 'html.parser')  # Get the html from the first page

num_jobs_area = str(soup.find(id='searchCount').string.encode('utf-8'))  # Now extract t
job_numbers = re.findall('\d+', num_jobs_area)  # Extract the total jobs found from the .

if len(job_numbers) > 3:  # Have a total number of jobs greater than 1000
    total_num_jobs = (int(job_numbers[2]) * 1000) + int(job_numbers[3])
else:
    total_num_jobs = int(job_numbers[1])
```

Fig 5.5 Code Snippet

The above code finds number of jobs



```
ETL+testing
Job is ETL+testing
city is jersey city
state is NJ
Printing base url http://www.indeed.com
Before url
There were 15 jobs found, jersey city
```

Fig 5.6 Snippet of output

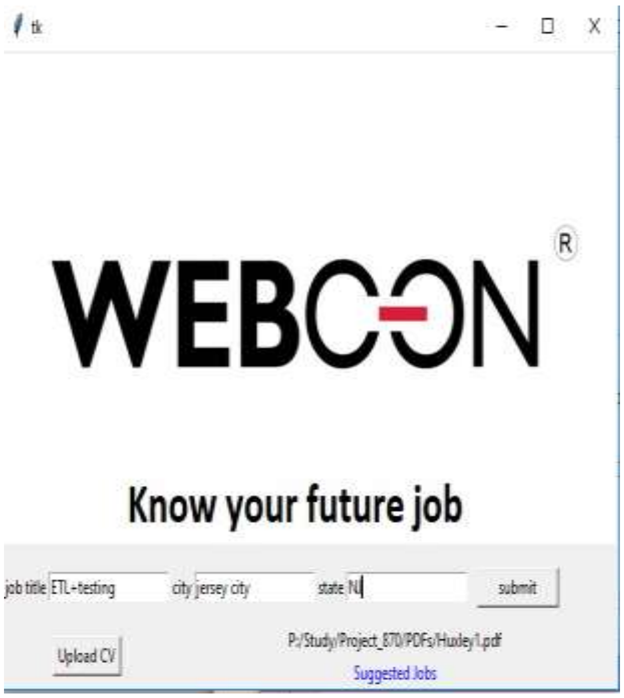
Output shows 15 jobs with the above given combination.

User can use any combination and see the number of jobs in that particular area with particular job title.

Entering the text field data (city and state) is optional.

6. Following figure shows the window that appears when the processing is finished. It shows "*Suggested Jobs*" hyperlink. Clicking on

this hyperlink shows the top recommended jobs by the system, opening separate browser tab for each recommended job.

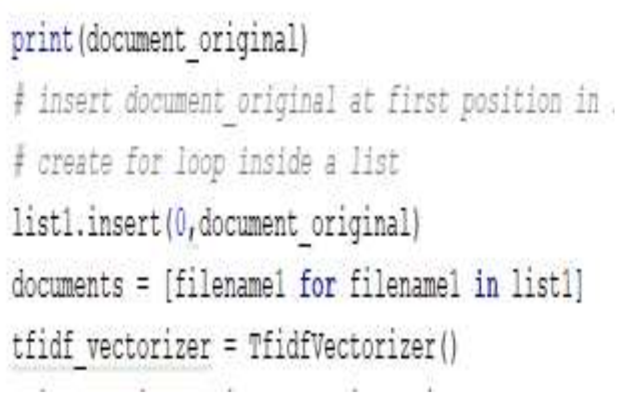


job title ETL+testing city jersey city state NJ

P:/Study/Project_870/PDFs/Huxley1.pdf

Fig 5.7 Result Window

7. Code snippet for finding top recommended jobs.



```
print(document_original)
# insert document_original at first position in .
# create for loop inside a list
list1.insert(0,document_original)
documents = [filename1 for filename1 in list1]
tfidf_vectorizer = TfidfVectorizer()
```

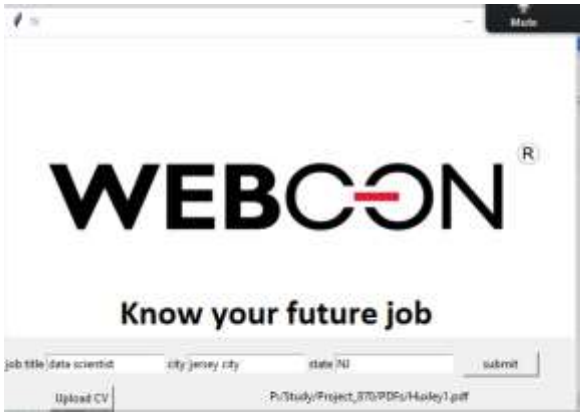
Fig. 5.8 Code Snippet

VI RESULTS

We executed the project multiple times and collected few results by specifying different search criteria. The output shows top 10 jobs, each opening in new browser tab. Here we have only specified the hyperlinks to those jobs.

Result 1:

Search criteria:



job title data scientist city jersey city state NJ

P:/Study/Project_870/PDFs/Huxley1.pdf

Fig. 6.1 Application Window

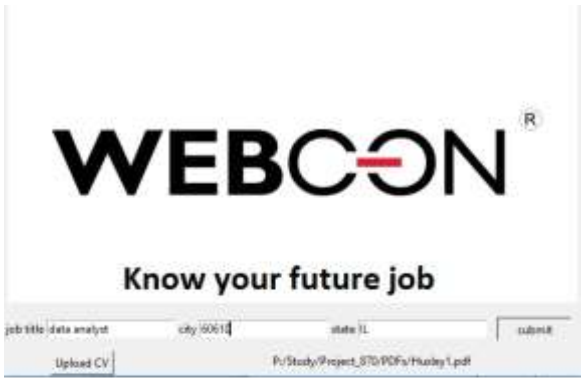
Job title: data scientist
City: jersey city
State: NJ

Keyword	Job heading	hyperlink	Cosine similarity
Data scientist	MId Level Data Scientist	Hyperlink1	0.0695
Jersey city	Data Scientist / Big Data Engineer (Ph.D. / Python / R / Lua / Scala / C++)-172098	Hyperlink2	0.0612
Nj	Data Scientist Executive - Commercial Banking Group (New York or San Francisco)	Hyperlink3	0.0612
"Uploaded cv"	Project Manager	Hyperlink4	0.0612
	Natural Language Processing Engineer (NLP)	Hyperlink5	0.0612
	Lead Data Scientist	Hyperlink6	0.0612
	Data Scientist - Japanese Speaking	Hyperlink7	0.0549
	Senior Data Scientist / Senior Machine Learning Engineer	Hyperlink8	0.0547
	Sr. Director, Technical Product Management, Enterprise Customer Intelligence	Hyperlink9	0.0473
	Product Designer	Hyperlink10	0.0461

Table 1: Top 10 recommendations

Result2:
Search Criteria:

Job title: data analyst
City: 60610
State: IL



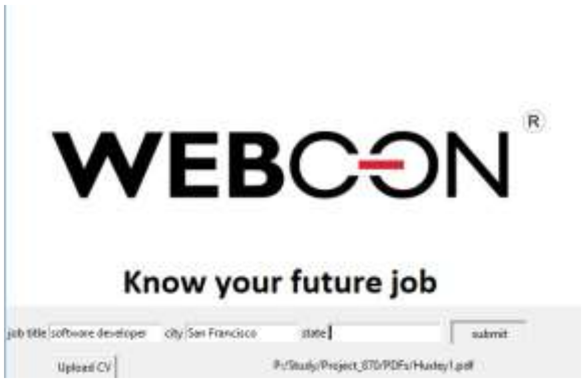
job title: data analyst
city: 60610
state: IL
P:/Study/Project_570/PDFs/Huxley1.pdf

Fig.6.2 Application window

Keyword	Job heading	hyperlink	Cosine similarity
Data Analyst	Talent Acquisition Partner	Hyperlink 1	0.0622
60610	AUTOMOTIVE SERVICE ADVISOR	Hyperlink 2	0.044
IL	Sr. Software Development Engineer in Test	Hyperlink 3	0.0437
"Uploaded cv"	AMAZON ALEXA IoT- Software Development Engineer - Alexa Machine Learning Team - Boston, MA	Hyperlink 4	0.0437
	AMAZON ALEXA IoT- Software Development Engineer - alexabostonjobs@amazon.com)	Hyperlink 5	0.0437
	Nurse Practitioner	Hyperlink 6	0.0437
	AUTOMOTIVE SALES ASSOCIATES	Hyperlink 7	0.0436
	Home Health Aide - Lamar, MO	Hyperlink 8	0.0436
	Software Development Engineer in Test (SDET)	Hyperlink 9	0.0429
	Registered Nurse-Per Diem - (179386)	Hyperlink 10	0.0408

Table 2: Top 10 recommendation

Result 3:


WEBCON®

Know your future job

job title software developer city San Francisco state submit

Upload CV P:/Study/Project_870/PDFs/Huxley1.pdf

Fig. 6.3 Application window

Search criteria:
Job title: software developer
City: San Francisco

Keyword	Job heading	hyperlink	Cosine similarity
Software developer	Staff QA Engineer (REF2679M) - Digital and Mobile Product Development (DMPD)	Hyperlink1	0.057
San Francisco	Sr. Prin / Architect - Software Dev - Chatbot Platform - AI / Machine Learning / NLP-17000YMO	Hyperlink2	0.0539
	Quality Engineer	Hyperlink3	0.0418
“Uploaded cv”	**C++ Software Developer**	Hyperlink4	0.0416
	Senior Software Engineer - Sales Cloud	Hyperlink5	0.0411
	Quality Assurance Lead	Hyperlink6	0.0409
	Senior Technical Program Manager	Hyperlink7	0.0409
	Sr Systems C++ Software Developer	Hyperlink8	0.0409
	Sr Systems C++ Software Developer	Hyperlink9	0.0409
	Enterprise Sales Engineer)	Hyperlink10	0.0409

Table 3: Top 10 recommendations

Note: The results are based on indeed API and availability of job at that time. These results may vary if the job requirements are fulfilled or company has removed it.

CONCLUSION

We developed a system which recommends jobs based on resume similarity with the job description gathered from job portal.

FUTURE SCOPE

1. We can display the list of all jobs instead top 10 jobs.
2. We can develop our own API.

REFERENCES

[1]https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/

[2] Alexandre Gramfort <alexandre.gramfort@inria.fr>,Mathieu Blondel <mathieu@mblondel.org>,Robert Layton <robertlayton@gmail.com>,Andreas Mueller <amueller@ais.uni-bonn.de>,Philippe Gervais <philippe.gervais@inria.fr>,Lars Buitinck,Joel Nothman joel.nothman@gmail.com on Github: https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/metrics/pairwise.py

[3]http://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[4]Thorsten Joachims, School of computer science, Carnegie Mellon University, Pittsburgh, PA 15213 March 1996. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization.

http://www.dtic.mil/get-tr-doc/pdf?AD=ADA307731

[5 Ali Seyed Shirkhorshidi , Saeed Aghabozorgi,Teh Ying Wah.] A Comparison Study on Similarity and Dissimilarity Measures in Clustering Continuous Data. Published : December 11,2015 http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0144059