

# **Job Recommendation System Using Machine Learning And Natural Language Processing**



**JEEVANKRISHNA**

Dublin Business School

Dissertation submitted in partial fulfilment of the requirements for the degree  
of  
*MSc in Data Analytics*

May 2020

## **Declaration**

I Jeevankrishna, declare that this dissertation that I have submitted to Dublin Business School for the award of MSc in Data Analytics is the result of my own investigations, except where otherwise stated, where it is clearly acknowledged by references. Furthermore, this work has not been submitted for any other degree.

JEEVANKRISHNA

10510202

May 2020

## **Acknowledgements**

I want to express my deep and sincere gratitude to my project supervisor, Dr.Shahram Azizi, for his guidance, encouragement, and support throughout my academic year in Dublin business school. His advice on the web scraping project in the first semester made me improvise my ideas and finally led to my dissertation topic. I'm grateful for all the staff who have taught and refined us with better knowledge. I want to extend sincere thanks to my best friends in Dublin Smita Sharma, Nikil Nair, Aditi Patil, Tushar Nayak, and Abhinov Bardan. I want to thank Mr.Chiranjeevi B Totad for his recommendation to study in Ireland. Also, I would like to express my gratitude for the whole stack overflow community. Their yearly survey data has been one of the best work given to the community of developers and students. Last but not least, I would like to thank my parents Sathyanarayana Rao and Sharada S Rao, who gave me an opportunity to pursue my master's dream, my brother Anoopkrishna, and all my cousins who supported and encouraged me at every step.

## Abstract

The rise of digital communication and the spread of the internet has made an enormous impact in every industry. One such domain is the Hiring process, where a job seeker applies to a job by creating a profile on a job portal by providing all his/her work preferences. These work preferences of each user can be collected from each user and provide job recommendations based on their preference. There had been work done in this field, where researchers have implemented Recsys using the Hybrid filtering method as user data had previous interaction with item (Rafter *et al.*, 2000). In this dissertation, we have approached the problem with the three-tier approach design. Data acquired for our study has no previous interaction between the user data and Job listing data. With such a dataset, we have addressed the issue of cold start from both User and Job perspective. Also, recommend the top-n job to the user by analyzing and measuring similarity between the user preference and explicit features of job listing using Content-based filtering, which is devised in support of natural language processing and cosine similarity. The Recommender System is then evaluated using precision, recall, and F1 score (Barrón-Cedeno *et al.*, 2009). The top-n recommendation made to the user is presented in the third tier of the design, a web app deployed in the local server. The presentation layer web-app is developed using Plotly's dash web framework.

**Keywords:** *Recommender system, Job domain, Content-based filtering, Natural language processing, cosine similarity.*

# Table of contents

<b>List of figures</b>	<b>vii</b>
<b>List of tables</b>	<b>ix</b>
<b>List of source codes</b>	<b>x</b>
<b>List of equations</b>	<b>x</b>
<b>Nomenclature</b>	<b>xii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 What is Recommender system? . . . . .	2
1.2 Why do we use Recommender system ? . . . . .	2
1.3 How to implement Recsys in the Hiring process? . . . . .	3
1.4 Problem Statement . . . . .	3
1.5 Research Aims and Objectives . . . . .	4
1.6 Road map for the Dissertation . . . . .	4
<b>2 Literature Review</b>	<b>6</b>
2.1 Introduction . . . . .	6
2.2 RecSys . . . . .	6
2.3 NLP . . . . .	10
2.4 Inferences . . . . .	13

---

<b>3</b>	<b>Methodology</b>	<b>14</b>
3.1	Web scraping - rvest . . . . .	14
3.2	Natural Language Processing- spaCy . . . . .	16
3.2.1	Word2Vec . . . . .	17
3.3	Content Based Filtering . . . . .	19
<b>4</b>	<b>Design And Implementation</b>	<b>22</b>
4.1	CRISP-DM with RecSys . . . . .	23
4.2	Data Source . . . . .	27
4.2.1	User profile Data . . . . .	28
4.2.2	Job profile Data . . . . .	30
4.3	Data Preprocessing . . . . .	31
4.3.1	User Data Preprocessing . . . . .	33
4.3.2	Job Data Preprocessing . . . . .	34
4.4	Modeling of Recommender System . . . . .	35
4.5	Evaluation . . . . .	37
4.6	Deployment . . . . .	41
<b>5</b>	<b>Results</b>	<b>43</b>
<b>6</b>	<b>Future work and Discussion</b>	<b>48</b>
6.1	Future work . . . . .	48
6.2	Discussion . . . . .	49
<b>7</b>	<b>Conclusion</b>	<b>50</b>
	<b>References</b>	<b>51</b>

# List of figures

3.1	Web Scrapping overview . . . . .	15
3.2	word2vec-banana . . . . .	17
3.3	word2vec models . . . . .	18
3.4	Assumption of Content-Based Recsys . . . . .	19
3.5	Content-Based Recsys . . . . .	20
4.1	CRISP-DM . . . . .	24
4.2	Modified CRISP-DM . . . . .	28
4.3	Design spec . . . . .	29
4.4	User profile data's structure . . . . .	29
4.5	Columns of Interest in User data . . . . .	30
4.6	Job data's structure . . . . .	31
4.7	Columns of Job data . . . . .	31
4.8	Process flow - Data Pre processing . . . . .	32
4.9	Process flow - User Data Preprocessing . . . . .	33
4.10	Processed user data . . . . .	34
4.11	Process flow - Job Data Preprocessing . . . . .	34
4.12	Process flow - Recommender system . . . . .	35
4.13	Screenshot-Webapp Initial page . . . . .	42
4.14	Screenshot-Webapp Recommendation page . . . . .	42

---

5.1	Screenshot-Webapp initiation . . . . .	46
5.2	Screenshot-Webapp homepage . . . . .	46
5.3	Screenshot-Webapp result page . . . . .	47



# List of tables

- 4.1 F1 score - User 7 . . . . . 39
- 4.2 F1 score - User 25 . . . . . 39
- 4.3 Average F1 score from sample set of users . . . . . 40
- 4.4 F1 score at threshold . . . . . 40
  
- 5.1 Average Coverage and F1 score . . . . . 45

# List of source codes

1	Install and Load library <b>rvest</b> . . . . .	22
2	Install packages required in python for NLP task . . . . .	23
3	Install packages required in python for Dashboard . . . . .	23
4	Load User data and display columns . . . . .	30
5	Load Job data and display columns . . . . .	31
6	Install packages required for Web-app Deployment . . . . .	41

# List of equations

2.1 Jaccard Index . . . . .	12
2.2 Jaccard Index in notation . . . . .	12
2.3 Cosine Similarity . . . . .	12
2.4 Cosine distance . . . . .	12
4.1 Job Score . . . . .	36
4.2 Cosine distance measure . . . . .	36
4.3 Coverage of item . . . . .	38
4.4 Threshold score . . . . .	38
4.5 $F_\beta$ measure . . . . .	39
4.6 $F_1$ measure . . . . .	39

# Nomenclature

## Acronyms / Abbreviations

ACF Automated Collaborative Filtering

CBF Content-Based filtering

CF Collaborative filtering

CV curriculum vitae

Cython Superset of the Python programming language, designed to give C-like performance

en\_vectors\_web\_lg English vector trained on large web content of words

HTML Hypertext Markup Language

NLP Natural Language Processing

NLTK Natural Language Toolkit

PCR Personalized Case Retrieval

RBF Rule-Based filtering

RecSys Recommender System

word2vec Word to vector

# Chapter 1

## Introduction

When the whole world is coming back on its feet, those businesses affected by this pandemic disease slowly tries to gain back the momentum it lost. Now is the time when the companies or businesses seek to invest in human resources, which would help them to gain the momentum it lost during this period. When the governments across the world ask businesses to halt the operation in the effort of controlling the pandemic, many companies asked their employees to work remotely. In contrast, many other companies started to reduce their operational cost by terminating employees who were in permanent and contract roles. Individuals who lost their job to the consequence of shutdown are awaiting for their next opportunity. Naturally, we human tries to strive through all difficulties to serve the purpose of our life. A daily job provides a sense of purpose to an individual(stillman, 2019), and he tries to get better at it, which results in leaving current employment and looking for a new one; this is a constant cycle of the hiring process.

To serve the constant cycle of the hiring process in the job applicant's perspective, many job companies have come up with solutions for providing the job board. Here a seeker looks up for the job he would find relevant to him and apply for it. As there are many job boards, applicants tend to use the tool that provides better services to them, services such as writing a CV, creating a job profile, and recommending new jobs to a job seeker.

Job applicants have become more persistent and proactive in searching for new opportunities that fit their skills. However, companies that are targeting these job seekers are finding it challenging to identify the job seeker's skill and provide personalized job recommendations

## **1.1 What is Recommender system?**

These are the systems that help us to select out similar things whenever we select something online. The concept of understanding a user's preference by their online behaviour, previous purchases, or history in the system is called a recommender system. The need for a recommender system has grown from time to time. At First, Entertainment industries exploited the benefits of these systems. Then recommender systems were implemented in e-shopping businesses, online news, but very few companies have tried implementing it in the hiring process.

## **1.2 Why do we use Recommender system ?**

Industries try finding ways to increase their revenue. In a classic business model, the up-selling is the term used when a sales advisor tries to sell an item to a customer based on things that he is planning to purchase. As business started to integrate the technology to increase the user interaction with business, customers gave out their preferences by interacting or purchasing the product the business is selling to them. The business has utilized the collected big data to make a better-personalized recommendation to its customer base. Netflix is a streaming service that generates its revenue from customer subscriptions to its content. According to reports from business insider Australia, Netflix estimates 20% of their video watches are derived from the search. Whereas, 80% comes from its recommendation system (McAlone, 2016).

## 1.3 How to implement Recsys in the Hiring process?

As we know the cycle of the hiring process, we have the opportunity to implement a recommender system to it. We can implement a recommender system from the enterprise perspective and the job seeker perspective. From the perspective of job seeker, we can implement a recommender system that could collect the user preference that is user skills, Location. We are concentrating on job-related to the Information and technology domain. These IT domain jobs require skills such as programming languages, database skills, Framework skills, and user preference on different platforms. The author's motivation to implement a recommender system for a job seeker came from personal strive when I was looking for the job. All the job boards would provide us with the jobs that the user searched for based on keywords that we entered in that search box. Is it not better to have a system in place recommend jobs based user skills and preferences. In this dissertation, I will be implementing a recommender system based using filtering techniques and Natural language processing to recommend top-n jobs based on the user profile.

## 1.4 Problem Statement

The dataset used for this research are sourced from Stack overflow survey data which is modelled as the user data for this research. Another dataset was created by web scrapping the Job board Using R programming language to fulfill the road map of this dissertation. The research question proposed by this research is "**Can an efficient recommender system be modeled for the Job seekers which recommend Jobs with the user's skill set and job domain and also addresses the issue of cold start?**". To answer the research question, below are the objectives that need to be satisfied with going forward.

## 1.5 Research Aims and Objectives

1. To scrape data from Job board to create offline Job dataset.
2. To develop a user profile based on stack overflow survey data
3. To construct a recommender model that can address cold start issue
4. To devise recommender model which recommends Job to the job seeker based on skills.

## 1.6 Road map for the Dissertation

This dissertation segmented into several chapters, where we describe each chapter as below,

**Chapter one** This chapter includes the introduction and background of the topic as well as the motivation of the author behind this dissertation. Also we briefly discussed on research question that need to be answer and objective for this dissertation.

**Chapter Two** This chapter includes a literature review on the Recommender system, Natural language processing and some of the similarity measures that have been utilized to research in the field of the recommender system

**Chapter Three** In this Chapter, researcher elaborate on the methods, tools and technology used in accomplishing the dissertation.

**Chapter Four** In this chapter, we discuss the implementation part of the dissertation. It includes data set description, programming languages and tools used, data pre-processing, modelling and evaluation method used.

**Chapter Five** This chapter will discuss the interpretation of the results, answering the research questions, and discussing the inferences of the findings.



**Chapter Six** This Chapter discuss about the future work that could be conducted using this research as a baseline to further study.

**Chapter Seven** This Chapter provides the conclusion to the dissertation by discussing results that we acquired from the research conducted on recommender system.

# Chapter 2

## Literature Review

### 2.1 Introduction

The recommender system is becoming part of every business. The business tries to increase its revenue by raising the user's interaction by recommending new items based on user preferences. We have witnessed the rise of Netflix in the entertainment domain, using their strategies to implement a recommender system into their existing ecosystem. But there has been a minimal study in the hiring field from the perspective of a job seeker. To start any research, it is quintessential to review relevant work in the domain and technology.

### 2.2 Recommender Systems

As discussed previously, RecSys are the system that analyses user preference history and caters them with different options of services related to the requirement. Recommender systems emerged as an independent research area in the mid-1990s(Ricci *et al.*, 2011). In recent years, the interest in recommender systems has dramatically increased. In the Recommendation algorithm, it classifies into four types: Content-based filtering, Collaborative filtering, Rule-based, and Hybrid approaches (Mobasher, 2007; Al-Otaibi and Ykhlef, 2012).

Collaborative Filtering (CF): Collaborative Filtering is a technique is based on the human ratings that are given to an item by a user and find similarity between different users who have given similar ratings to an items(Hu and Pu, 2011). The essential operation used here is the memory-based nearest neighbor approach to group users who have a similar interest. As the volume of data grows gradually, there will be high latency in generating recommendations Mobasher (2007); Herlocker *et al.* (1999). Collaborative filtering has an advantage over content-based filtering techniques, but due to the nature of the hiring process, a job cannot be rated by the user and will not be possible to create a similarity matrix.

Content-based filtering (CBF): These are the most subjective and descriptive based filtering. Content-based filtering can also be called as attribute-based recommender as it uses the explicitly defined property of an item. It is an approach to an information retrieval or machine learning problem. The assumption made in content-based filtering is that user prefers item with similar properties. Content-based filtering recommends items to the user whose properties are similar to the item which the user has previously shown interest. Mobasher (2007) express that drawback of this filtering technique is their tendency to over-specialize in suggesting the item to a user profile as user profiles are relayed on an attribute of the previous item opted by the user. Nevertheless, in the job domain, the job listed in the job board be available only for few days; due to the nature of the domain, the tendency to over-specialize in recommending the same item would not be any problem in the job domain recommender system. In domains like entertainment, user preference are tends to change depending on various factors, but In Job domain, the user tends to look for the job where he can use his previous skills. New recommendation of jobs can be made when there is a change in user preference, i.e. if a user thinks to change his/her job domain by updating his new skills and the job domain if he/she wishes. Another scenario of new recommendation is when new jobs are listed in the database; system would identify the properties of the job listed, such as

job domain and skills required for the job and matches with the users with a high similarity score.

**Rule-based Filtering (RBF):** These filtering techniques depend upon decision rules such as an automatic or manual decision rule that are manipulated to obtain a recommendation for the user profile. Currently, the E-commerce industry uses a rule-based filtering technique to recommend an item based on the demographic region of a user, purchase history, and other attributes that can be used to profile an user. A drawback in rule-based filtering is user feeds the information to the system. These inputs are utilized as a description of a user profile or can be considered as a preference of a user, defined by the user. Thus the data acquired is prone to bias. With the age of the user's profile, recommendation tends to hit the saturation and become static Mobasher (2007).

**Hybrid filtering (HF):** As the title describe, its incorporation of multiple techniques to improve the performance of recommendation. The previously discussed recommendation technique has its weakness and strengths. In order to get a better recommendation and overcome the challenges posed by earlier techniques, this technique is sought after. All of the learning/model-based techniques suffer from cold-start in one or other form. It is a problem related to handling a new user or new item. These and other shortcomings of the CF, CBF, and RBF could be resolved by using hybrid filtering techniques Burke (2007); Jain and Kakkar (2019); Dhameliya and Desai (2019).

The surveys conducted by Burke (2002) and Dhameliya and Desai (2019) have identified different types of hybrid filtering techniques that could be used by integrating CF, CBF, and RBF.

1. **Weighted:** The similarity score obtained from different recommendation components are coupled numerically to get one better recommendation.
2. **Mixed:** Recommendations obtained from different recommending techniques are put together and presented as one recommendation.

3. Switching: choosing one among the recommendation components based on the scenarios where it suits best.
4. Feature Combination: Attributes derived from diverse knowledge origins are fused and supplied to a recommendation algorithm.
5. Feature Augmentation: One recommendation technique is used to compute a set of attributes of user or item, which is then part of the input to the next recommendation technique. Two or more recommendation techniques are serialised to get on recommendation.
6. Cascade: Recommending systems are given strict priority, with the lower priority ones breaking ties in the scoring of the higher ones. Here one Recsys technique refines recommendation of another.

There had been attempts to develop a recommendation system by several researchers. One such implementation was done by Rafter *et al.* (2000). They had devised a hybrid Recsys CASPER for Job finding search engine. They had implemented an automated collaborative filtering module and personalized case retrieval module in their job recommendation system. ACF module utilized user behavior information such as read time and activity on the page during his time on the system to profile the user. Similarity measure such as the Jaccard index and other clustering algorithms was used for similar grouping user against target user. Their other module PCR finds the similarity between the user's query and jobs in the system. The module computes similarity with a target user's query and jobs from the job case base using different similarity measures. This system has faced sparsity and scalability problems.

## 2.3 Natural language processing

These are the times that can be considered as an era of data. Every keystroke hit on twitter, online news, or in a research paper is recorded somewhere on the internet. All these generated data are available for the analysis through many means. In this abundance of data, Text data holds the majority of the share. Most of these text data are in an unstructured form. To put the abundance of text data into a perspective, a trillion-plus query per year is being handled by Google, and Whatsapp handles 30+ billion messages per day. That being said, how do we extract information from the unstructured text data or how can we make machine understand what the text is about? To answer all the questions, Text analysis is a most sought after technique to extract useful information from the text data. Text analysis can be performed by utilizing techniques such as Natural language processing. Natural language processing is a process of information retrieval from unstructured data. It refers to the utilization of computers to process natural language(Brants, 2003). The advancement in the personal assistant, text summarizing, and methods to caption a subject is due to the successful research in the field of NLP. Search engines like google and other industry leaders utilize NLP to its full extent. The gap between industry and academia in the field of NLP is very minimal as there is an advancement in the NLP; the business has tried implementing and has brought closer to everyone's life.

In Recsys for the hiring domain, the data we handle here is nothing other than text data. A user profile describes the details about user experience and skills he/she familiar with. On the other hand, the job listed has information as job title, skills required to fulfill the role. All these information is filled with text data. In this scenario, we utilize the Natural Language Processing to measure the similarity between Jobs by checking the similarity between the job title and job description of the listed job. Determining the text-similarity is an essential task in several industrial application such as query search, text summarizing and video tagging(Kenter and De Rijke, 2015). In earlier studies, researchers have used

different approaches to identify similarity between the text by using edit distance algorithm which is discussed by Mihalcea *et al.* (2006), lexical overlapping technique (Jijkoun *et al.*, 2005) as this might work in most cases but can't rely on these technique because of its frail nature(Kenter and De Rijke, 2015). In such cases, we rely on technique called word embedding. This is huge development in the field of distributional semantics. As this requires only a large amount of unlabelled word data. These words are represented in semantic space as a vector. That is, words that are semantically similar will stay close in the semantic space. In order to retrieve terms that based on the similarity between two terms, we can utilize most well know method called word2vec a vector space model then we can use cosine similarity to measure the similarity between them (Shrestha, 2011; Barrón-Cedeno *et al.*, 2009). This model can also be used to determine similarity between the sentences(Barzilay and Elhadad, 2003). It's a group related model which is used to produce word embedding and these are set of language modelling and feature learning techniques of NLP where words are mapped to real values in the vector. Typically word2vec takes large set of words which is called corpus as a input and produces vector space with dimensions being in hundreds(Mikolov *et al.*, 2013). Once vector space model is generated we can use similarity measuring method to determine the distance or how similar is the word with which we are comparing. To find similarity in vector space we can use similarity measures like Cosine similarity and Jaccard similarity.

1. **Jaccard Coefficient:** Jaccard Coefficient is a method to compare elements of two sets to identify which elements are shared between two sets and which are distinct. It's similarity measure for two sets of data with result ranging from 0% to 100%. Two sets can be said similar, when result is close to 100% . Formula for Jaccard Index is as shown below(Sternitzke and Bergmann, 2009),

$$Jaccard\ Index = \frac{Number\ of\ Elements\ common\ in\ two\ sets}{Number\ of\ Elements\ in\ two\ sets} \quad (2.1)$$

The above formula can be put into notation as below,

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} \quad (2.2)$$

2. **Cosine similarity:** Cosine similarity is also a measure to find similarity between two sets of non zero vector. It is a weighted vector space model utilized in the process of information retrieval. The similarity is measured by using euclidean cosine rule,i.e., by taking inner product space of two non zero vector that measures the cosine of the angle between the two vectors. If the angle between two vectors is 0deg , then the cosine of 0 is 1; Meaning that the two non zero vectors are similar to each other.In order to weight the words we have used the well-known word2vec vector space model(Rong, 2014; Herremans and Chuan, 2017).

$$\text{Cosine Similarity}(A,B) = \cos(\theta) = \frac{\sum_{i=1}^n A_i * B_i}{\sqrt{\sum_{i=1}^n A_i^2} * \sqrt{\sum_{i=1}^n B_i^2}} \quad (2.3)$$

We can also compute cosine distance by using below equation,

$$\text{Cosine Distance} = 1 - (\text{Cosine Similarity}) \quad (2.4)$$



## 2.4 Inferences

Based on all the research methodologies and techniques reviewed in this chapter, the CF technique cannot be considered as it does not satisfy the aims of the research. As the dataset of the user does not hold the information of rating against a particular job, we will not be able to create a rating matrix that requires for CF technique. Instead, I have chosen to implement content-based filtering. I used multiple attributes in the user data to create a user profile and recommend the job to those profiles which have a high similarity score received from cosine similarity. Also, i have given higher weights to job skills when compared to the job domain of the user while computing similarity scores between user profile and job.

# Chapter 3

## Methodology

Text data is one of the principal kinds of data. As we are extracting data from the job board, We need tools to scrape the data from the website. These text data will be in an unstructured format, so we need to pre-process the data before going to the modeling stages. Once the extracted data is processed and transformed in to feature, we can further go ahead utilize those features as input to the content-based filtering. Then we feed the user and item feature matrix to a similarity measure algorithm to identify the similarity between the job and user. Once the score is generated to every job against the user profile, we will be recommending the job that has high scores, and display top-n scored jobs to the users' dashboard, which is built on top of the Plotly's Dash web framework. The feature creation and algorithms used in this project are explained below in details:

### 3.1 Web scraping - rvest

For any study to be performed, the main concern is to collect the data that is useful to the study. If the data is available on the internet in an HTML format on a particular website, then a traditional way to collect the data would be hard, as it would be time-consuming. The data that are available on the internet will be in forms tables, comments, articles, job listing,

which are embedded in different HTML tags. Gathering such data would not be an easy task considering the volume of it. So we rely on the method such as web scraping. This technique came into existence right around the year, where the internet was introduced to the world.

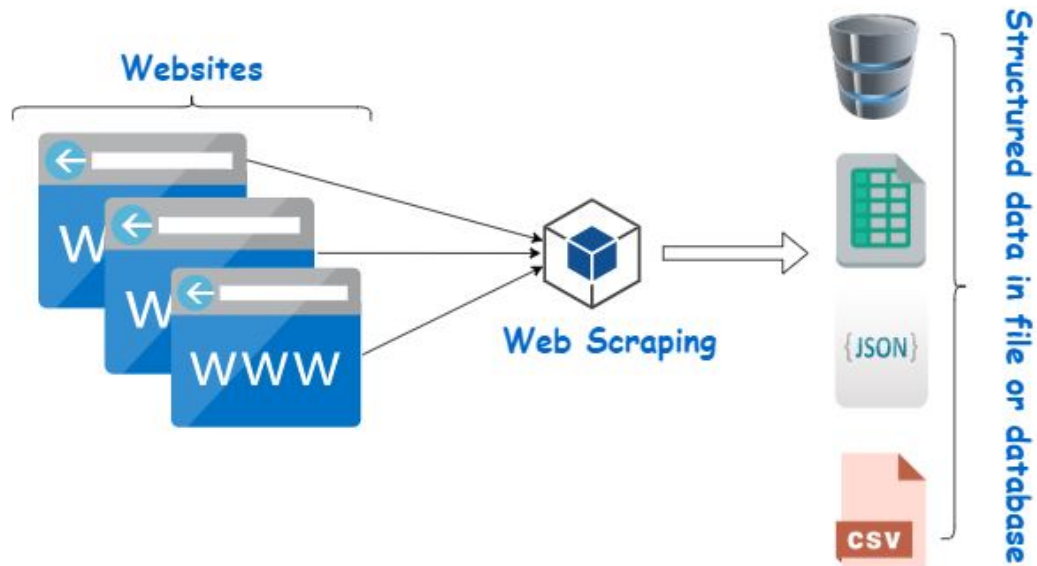


Fig. 3.1 Web Scrapping overview (SysNucleus, 2019)

Web scraping is the technique used to collect or extract information from the web sources and store it locally for the further analysis of the user's choice. Web scraping is also termed as screen scraping, web data extraction, or even web harvesting. Data in HTML language can be viewed through a web browser (Slamet *et al.*, 2018). Every website has its own structure, so the method of web scraping is hard to generalize for every website. So we rely on automating or creating a web crawler using python or R programming language. Python has a package called BeautifulSoup, which is used to request the HTML data and parse it using HTML nodes available in the file (Richardson, 2004). As websites try to put a restriction on the client system to avoid web scraping, The code which was intended to automate the extraction of the data would get blocked by a site. R programming also supports developers with web scraping packages such as rcurl and rvest. rcurl is the base web scraping package provided by the R, whereas rvest is also a package for web scraping developed by tidyverse

inspired by beautiful Soup. Both packages load HTML to an object, but the difference with rvest is it is not just a web parser, but it can connect to a web page, scrape and parse HTML in a single package. So to extract the data from the source web site, i have written a R code using rvest package.

## 3.2 Natural Language Processing- spaCy

In this study, Natural Language processing plays a vital role. The data collected by web scraping is descriptive values, and also the user data collected from stack overflow contains descriptive fields. As both data has no history of a previous interaction between them, the study was continued with the approach of analyzing the explicit property of the content. The data set is file full of text data, which is unstructured, and on the other hand, user data that was collected from stack overflow is structured. The current study needed a method to analyze text data to categorize all the job listing into a different category, also to find the similarity between the vector of words from the user data and the vector of words from the job listing data. In the current study spaCy is the Natural language processing package used instead of NLTK. Both NLTK and spaCy are popular NLP tools available in python. While both tools can accomplish required NLP task, each one excels in particular scenarios. spaCy is developed in perspective of developers and real time production environment. While NLTK provides all the access to algorithms to complete the task,spaCy provides accurate syntactical analysis of all NLP libraries available in NLTK with better performance and It also provides access to largest word vector corpus called en\_vectors\_web\_lg. While NLTK is not it's not known for robustness and fast results because of it's approach in handling inputs and output. On the other hand , spaCy is completely built on Cython from ground up which contributes in it's fast execution time and better performance(Kakarla, 2019).In this current study, spaCy's is used in extracting Named entity from the job description , also used to check similarity between the user profile. The similarity function in the spaCy package allows us to compare

two vector of words provide how similar two words are from one to another. This similarity function utilises word vector which is represented in multidimensional representation of the word. These word vectors are generated using word embedding algorithm. Below is the figure 3.2 which vector output for the word "BANANA" taken from spaCy.

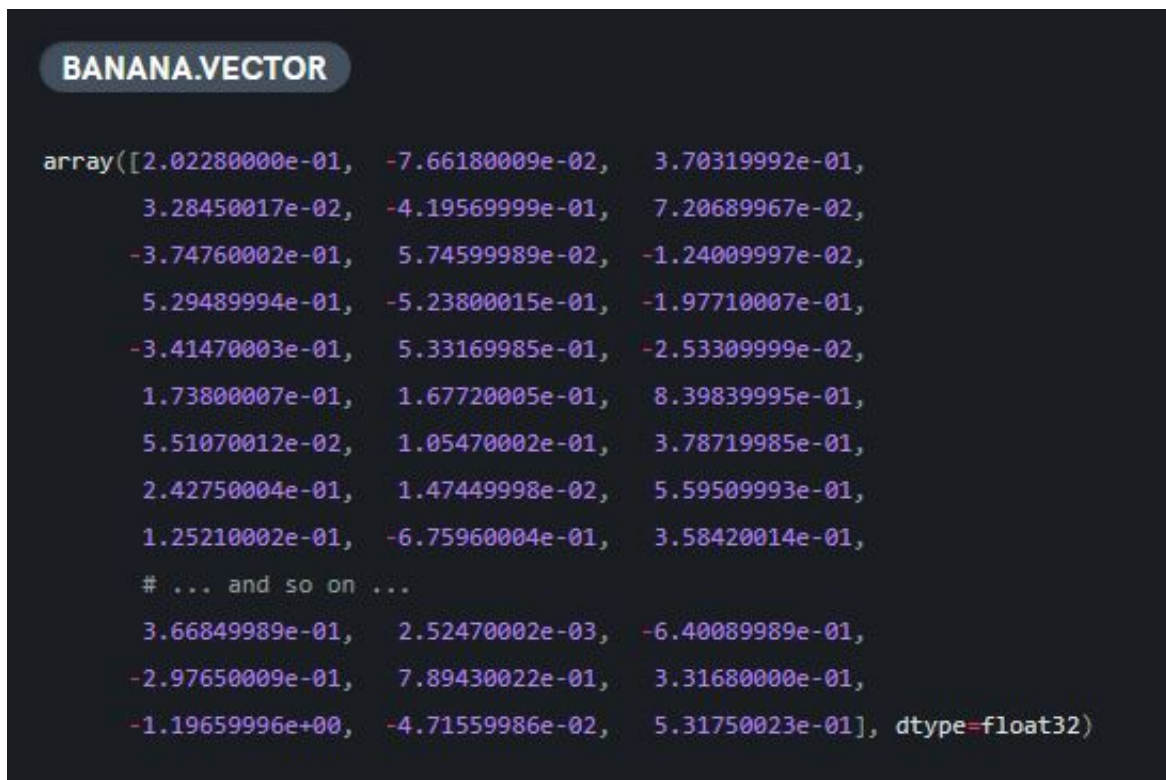


Fig. 3.2 Vector for the word "BANANA"  
source:<https://spacy.io/usage/vectors-similarity>

### 3.2.1 Word2Vec

Word embedding is the method to translate the words or phrases from the corpus to vectors of real number as shown in figure 3.2 . It's used to language modeling and feature learning methods as-well. The corpus here is taken from the spaCy's one of the largest word model i.e., en\_vectors\_web\_lg. Word2vec word embedding algorithm takes large corpus en\_vectors\_web\_lg of text as input and produces a word vector space, which is in several

hundred dimensions. These models learn based on the two layered or shallow neural network method to perform the required task. Word vectors that are in same vector space are bound to be similar and share same context to one another. This allows us to use this word embedding method in our study to find similarity between two word vectors. There are two different learning models available in word2vec algorithm that are Continuous bag-of-words or CBOW model and Continuous skip-gram model(Brownlee, 2017).

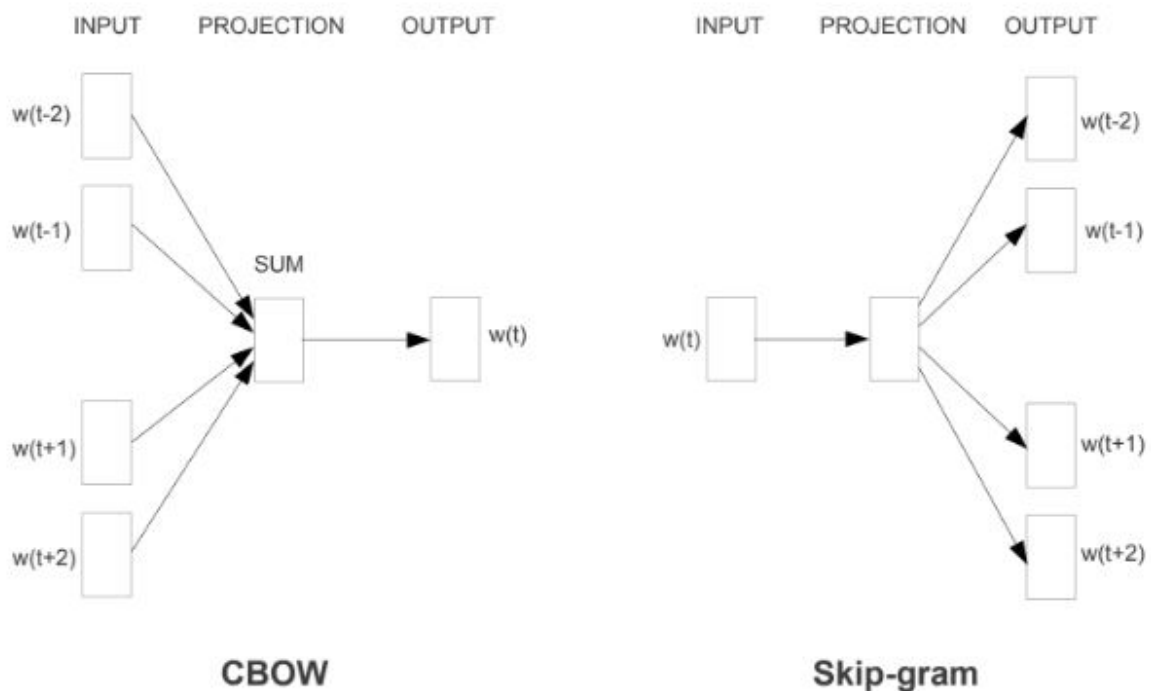


Fig. 3.3 Representation of Word2vec training models  
source:<https://arxiv.org/pdf/1301.3781.pdf>

The CBOW model tries to learn the word embedding by predicting the current based on surrounding words or context. Whereas, Continuous skip-gram models learns by predicting the context or the surrounding words based on the current word. The advantage of using word2vec word embedding method is that it can learn efficiently with a high quality word embedding's which in-turn allows us to learn from big corpus like en\_vectors\_web\_lg of spaCy(Brownlee, 2017).

### 3.3 Content Based Filtering

As discussed earlier, the collaborative filtering method uses the history of the user's interaction with an item as the input for building the recommenders. In contrast, Content or attribute-based recommenders use specific properties of an item or the user along with the previous user interaction with the item to recommend the user with the relevant item. As we have collected two data set, where one is a user data set another is dataset of job extracted from web, there is no previous interaction between the user and item. So during the implementation, we will be generating user profile and the item profile based on the dataset considering the explicit attributes of user and item. Content-based recommenders operate with the assumption that items having related attributes have the similar interest at the user level. Content-based recommendation engines keep suggesting an item to a user similar to

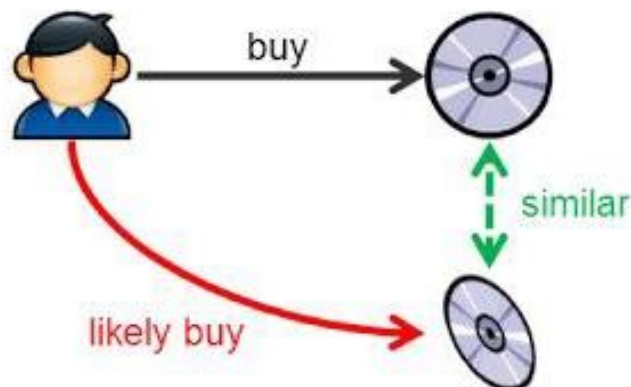


Fig. 3.4 Assumptions made in Content-Based Recsys (Luk, 2019)

the items rated highly by the same user. The user will most likely get recommendations about jobs with the same skill or Job domain as the user preferred in the past. In most scenarios, this behaviour of the recommending system is considered as it is saturated. However, in the job domain, user preference remains the same in the system. As the user rarely changes his preference to change his job domain or work on different skill sets and also the previously

recommended job would not be in the system as job listings will expire when the role is fulfilled.

There are two approaches when it comes to content based recommendation,

1. Analysing explicit property of the Content.
2. Building User Profile and Item Profile from User Rated Content

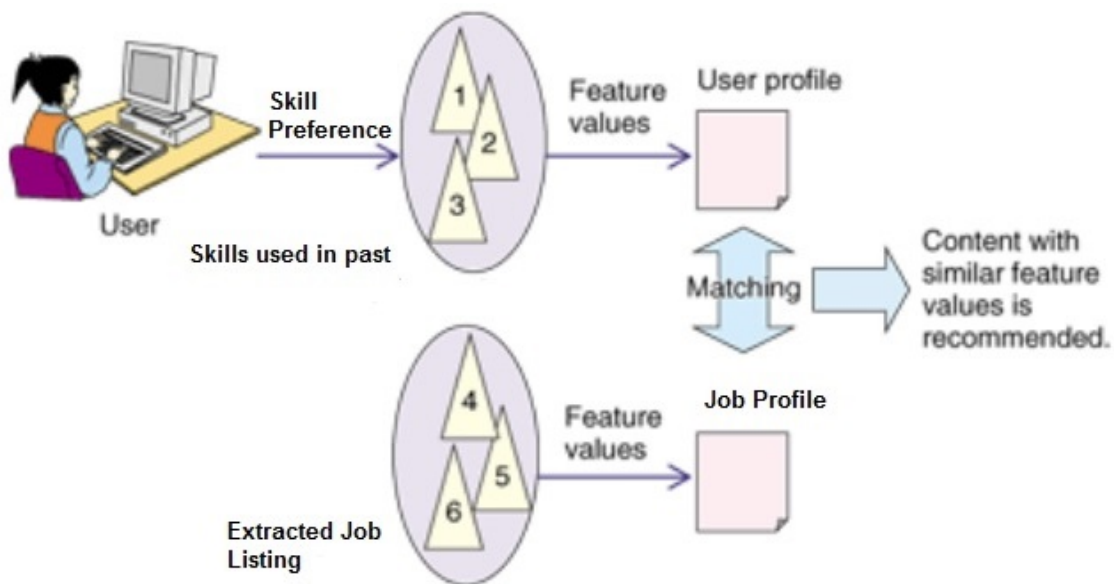


Fig. 3.5 Overview of Content-Based Recommender System (Recsys, 2012)

In this study, the approach followed is Analysing explicit property of the content i.e., Job title and Job description will be analysed using Natural language processing technique offered by a python software package called sapCy. All the words in the Job description, are loaded vector space of Word2Vec where the corpus is provided by sapCy is utilised in this study. The corpus used in this study is en\_vectors\_web\_lg which includes over 1 million unique vectors. Then a similarity algorithm is used along with Word2Vec in the similarity function of the sapCy to measure the similarity between the Job category and the Job title , same process is used to get similarity between Job category and job description. Unlike other recommender system, content-based filtering has works on the content of the item and user, this can avoid



problems of new item or new user. Depending on the requirement, content-based filtering allows us to use approaches like text processing or semantic information to analyze the data and also the recommender system will be transparent as the recommendation made to the user can be explainable based on the content in the item.

# Chapter 4

## Design And Implementation

In Previous Chapter, We discussed all the methodology that need to be used in the study to implement the recommender system for a hiring process in perspective of job seeker. This chapter will discuss on how implementation was carried out using CRISP-DM framework and also discuss each step on detail and what led to model a recommender system for hiring process.

Before advancing to implementation stage, we shall discuss on the tools and technology used in implementing the recommender system. The data extraction was done using R-programming code, Rx643.5.3 version was used in collaboration with R studio. The R library used to extract the data from website was "rvest", Below was the code to install and load the package in the environment.

```
>install.packages("rvest")  
>library(rvest)
```

Listing 1 Install and Load library **rvest**

To implement recommender system, the rest of the development was done in the python environment. The python 3.7.3 environment was used within anaconda framework with

spyder IDE. Listing 2 and 3 shows the packages/modules that were installed in python to achieve a working recommender system.

```
>>>pip install spacy
>>>pip install nltk
>>>python -m spacy download en_core_web_lg
```

Listing 2 Install packages required in python for NLP task

```
>>>pip install Dash
>>>pip install plotly
```

Listing 3 Install packages required in python for Dashboard

## 4.1 CRISP-DM with RecSys

CRISP-DM (Cross-industry standard process for data mining) is a widely used documented, open-source process available to use without any restriction. We can state CRISP-DM as an open-source model openly available to use and experiment. It is a process formed after consulting various data practitioners with their interest aligned with the industry of data mining and information retrieval. This Framework has been in use and widely accepted by the organization as it is a model which is neutral to all industry or application in use as it obtains the better result from data mining(Shearer, 2000).

The generic CRISP-DM model provides us with an overview of the tasks and phases involved in the data mining project. Generic CRISP-DM model has six phases in it, and all task of the data mining projects falls within these six phases. Fig 4.1 shows the six phases of the CRISP-DM model.

Once a phase is completed, depending on the results obtained from that phase, it will be decided to go further into the next phase or go back to any stage of the phase to get the desired result. So, these phases do not have a strict sequence. The outcome of these phases

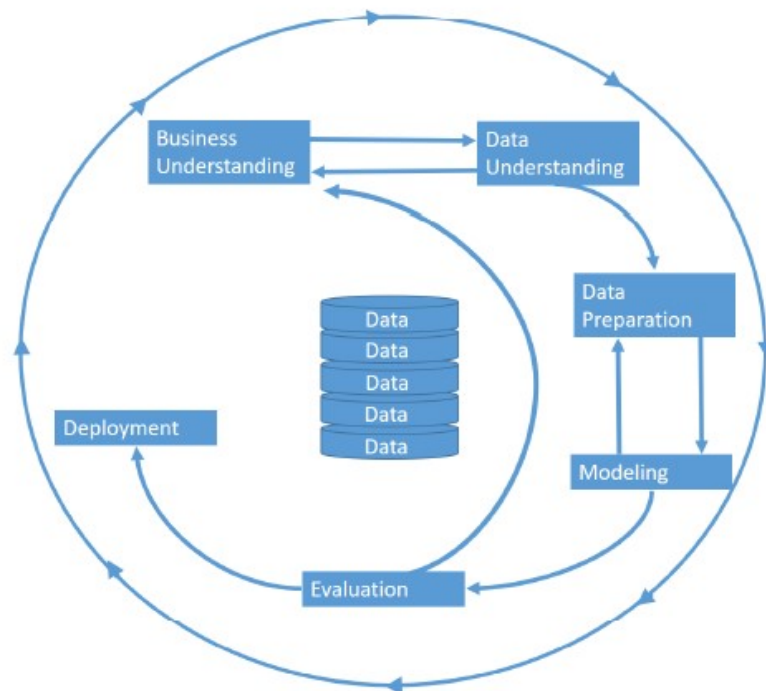


Fig. 4.1 Six phases of generic CRISP-DM reference model

Source:<http://www.cs.unibo.it/~danilo.montesi/CBD/Beatriz/10.1.1.198.5133.pdf>

will be one of the factors to decide whether to go further into the next phase or re-run the same phase. The outer circle of the Fig 4.1 shows the complete process after deployment, where the process sustains to provide better results and performance.

Lets discuss each stage in detail by defining task that are to be performed under that each phase of data mining process.

**Business understanding:** That is the initial CRISP-DM process. In this step a comprehension of the project's aim Furthermore, criteria should be established from the business perspective. This understanding would then allow us to identify the challenges of data mining to establish a strategy to achieve the objectives. This information can be obtained by pursuing the Tasks below.

1. Assess the situation
2. Determine the data mining goals

3. Producing project plan

**Data Understanding:** Data understanding is a method in which we state data facts by performing the tasks below to obtain better information about the data used in the data mining project. We also form hypotheses in this process for hidden knowledge about the target of the data mining project, based on experience and professional assumption. The process of Business understanding and data understanding are closely related. There must be some understanding of available data is required for the execution of the data mining project, and its plan.

1. Collect the initial data
2. Describe the data
3. Explore the data
4. Verify the data

**Data preparation:** This is the stage in which an analyst collects the data related to the data mining project and prepares it for further phases. The data preparation stage includes all activities from the initial stage of raw data to create the final data set. Data preparation tasks can be performed multiple times and can be performed in any order. Below is the task to be performed during the preparation of the data.

1. Select data
2. Clean data
3. Construct data
4. Integrate data
5. Format data

**Modelling:** In this phase, we will use one or more modeling techniques and also try to optimize by the model by varying the hyper-parameter of it. Among available modeling techniques, choose the appropriate for the task. The chosen modeling technique can be conservative, it expects the data to be in a particular format. In such a situation, the process might have to roll back to the data preparation model before working on the modeling phase again.

1. Select the modelling technique
2. Generate test design
3. Construct data
4. Creation of model
5. Assess the model

**Evaluation:** We would have a working model to analyze the data. However, before going into deployment, we need to check our model against actual production system datasets and assess the data mining findings with a business objective. Identifying whether the model has met the customer's requirements is essential in this phase. Below are the steps to be taken under the assessment phase

1. Evaluate result
2. Process review
3. Determine further steps

**Deployment:** Once the model meets the requirements of the customer in the evaluation phase, the model needs to be deployed to a production environment at a customer site. This involves both all the stakeholders of the projects, mainly the customer and the analysts. A team of analysts needs to provide knowledge transfer sessions to the customer on how they

can utilize from the deployed model. Below are the tasks that are meant to be performed in this phase

1. Plan deployment
2. Plan monitoring and maintenance
3. Generation of final report
4. Project review

This framework is designed for problems that require an iterative approach to understand the data in a better way before applying it to a model. To answer the research problem of this project, the implementation of this project will be done based on the CRISP-DM framework. CRISP-DM framework is modified to the level where it suits the business requirement of designing a recommender system for a job seeker. Fig 4.2 shows the Modified CRISP-DM framework for recommender system.

The design proposed for the study of recommender system for job seekers is a three tier design. The Data layer is combination of two sets of data i.e., user data from stack overflow and Job listing data which was extracted using R-Tools. The processing layer consist of Python/Spyder platform which is used for analyze, model and recommendation. Also research uses third layer as a presentation layer which consist of python/dash web framework which is used to recommend user top-n jobs for the skills and user's job domain. Fig 4.3 shows the three tier approach used for the current study.

## 4.2 Data Source

The Dataset used for this research is collected from two different source. First, the User profile data set- It is acquired from stack overflow survey 2019 (overflow, 2019) from the URL - <https://insights.stackoverflow.com/survey/>.

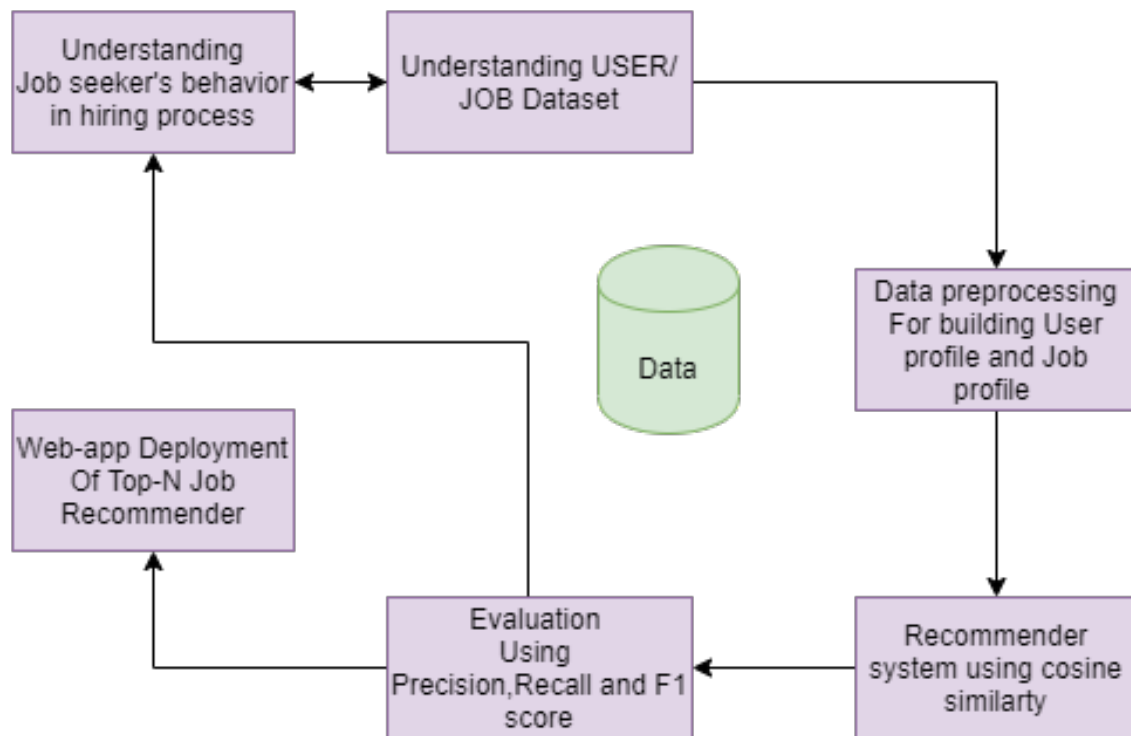


Fig. 4.2 Modified CRISP-DM for recommender system model

The Job listing dataset was created using web scraping method using R/Rstudio using a rvest package. The targeted Job listing site was "Seen" a new job listing website from indeed with URL- <https://www.beseen.com/tech-jobs/index.html>.

### 4.2.1 User profile Data

As the data is collected by stack overflow by conducting a survey, there are 88.9k observation in the data set, With 87 columns where just 1 column is integer datatype and 1 column with Boolean data type and rest 85 columns are string data type. Fig.4.4 shows the structure of the data set.

As this Data is acquired by conducting a survey there are columns that are not in interest of this research. The columns that are considered for the research are 'Respondent', 'DevType', 'LanguageWorkedWith', 'LanguageDesireNextYear', 'DatabaseWorkedWith',



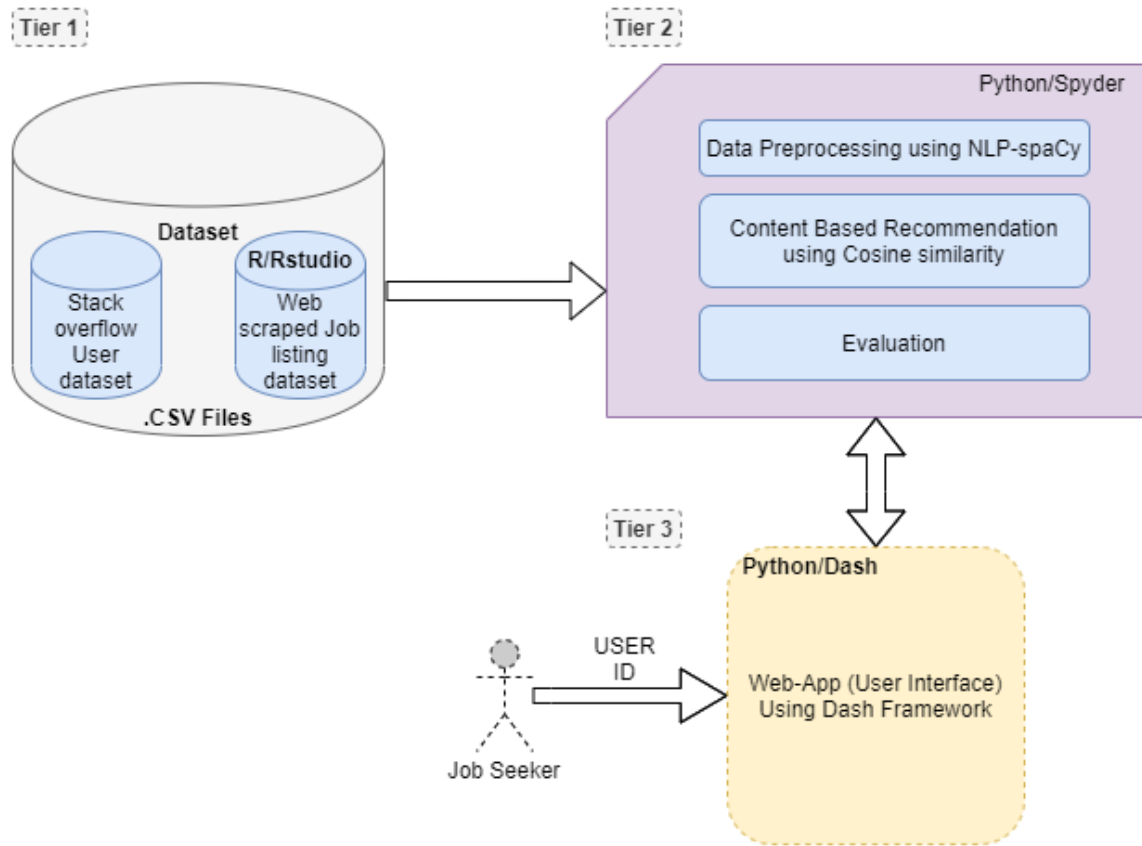


Fig. 4.3 Three Tier Design for recommender system

survey\_results\_public.csv

**Summary**

87 columns	
String	85
Integer	1
Boolean	1

Fig. 4.4 User profile data structure

'DatabaseDesireNextYear', 'PlatformWorkedWith', 'PlatformDesireNextYear', 'WebFrameWorkedWith', 'WebFrameDesireNextYear', 'OpSys'. Fig.4.5 shows the columns that are of interest to the research.

```
import pandas as pd
df = pd.read_csv('../input/stack-overflow-developer...
-survey-results-2019/survey_results_public.csv')
#subsetting dataframe
df1=df[['Respondent', 'DevType', 'LanguageWorkedWith',
'LanguageDesireNextYear', 'DatabaseWorkedWith',
'DatabaseDesireNextYear', 'PlatformWorkedWith',
'PlatformDesireNextYear', 'WebFrameWorkedWith',
'WebFrameDesireNextYear', 'OpSys']]
df1.columns
```

Listing 4 Load User data and display columns

```
df1=df[['Respondent', 'DevType', 'LanguageWorkedWith', 'LanguageDesireNextYear', 'DatabaseWorkedWith',
'DatabaseDesireNextYear', 'PlatformWorkedWith', 'PlatformDesireNextYear', 'WebFrameWorkedWith',
'WebFrameDesireNextYear', 'OpSys']]
df1.columns

Index(['Respondent', 'DevType', 'LanguageWorkedWith', 'LanguageDesireNextYear',
'DatabaseWorkedWith', 'DatabaseDesireNextYear', 'PlatformWorkedWith',
'PlatformDesireNextYear', 'WebFrameWorkedWith',
'WebFrameDesireNextYear', 'OpSys'],
dtype='object')
```

Fig. 4.5 Columns of Interest in User data

## 4.2.2 Job profile Data

As Job listing data was web scraped from seen job board, the extracted data was then saved to .csv file. There are 615 observations in the file, each observation represents a job listing from the website. There are totally 7 columns in the data set, With one being Id type and 6 of the column are String data type. Fig.4.6 shows the Job dataset's structure.

As web scraping was performed on the seen jobs website to collect information about 615 job listing, the data was stored to comma separated files with columns being 'Job\_Id',



Column Name	Count
String	6
Id	1

Fig. 4.6 Job data's structure

'jobtitle', 'company\_name', 'job\_role', 'jobdescription', 'job\_updatation', 'skills'. Fig.4.7 shows the columns available in the job dataset.

```
import pandas as pd
df = pd.read_csv('../input/job-data/seenJobs_new10_.csv', ...
encoding='unicode_escape')
print(type(df))
df.columns
```

Listing 5 Load Job data and display columns

```
#df = pd.read_csv('../input/job-data/seenJobs_new10_.csv', encoding='unicode_escape')
print(type(df))
df.columns
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Index(['Job_Id', 'jobtitle', 'company_name', 'job_role', 'jobdescription',
       'job_updatation', 'skills'],
      dtype='object')
```

Fig. 4.7 Columns available in Job data

## 4.3 Data Preprocessing

In this study, dataset acquired for the study has attributes filled string column with symbols and stop words. Especially in the column where skills details are present in both

dataset. Data preprocessing is our first process in the second tier of our study's three tier architecture. Below is Fig4.8 that shows data preprocessing process flow.

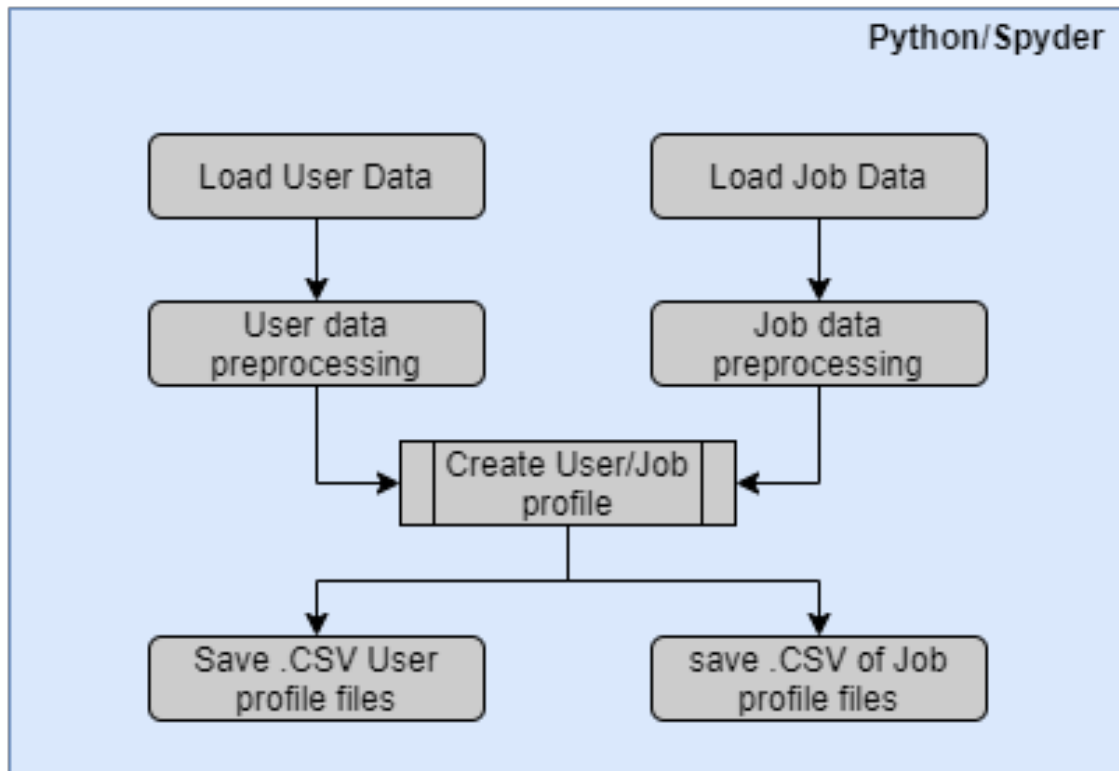


Fig. 4.8 Process flow - Data Pre processing

All the user and Job data is in a single comma separated file. In data preprocessing, our goal is to create a user preference matrix for each column. i.e., we will be creating two-dimensional matrix with each rows giving detail of user preferred language skill or database skill. The values of one particular skill column is striped from it's default form and transformed to be a columns name against user in each row. We will discuss the data preprocessing of each dataset in detail further sections.

### 4.3.1 User Data Preprocessing

In this section, let us look into the process of how the data have been transformed into a two-dimensional matrix for each column in the user data set. Fig4.9 shows the user data's data preprocessing steps in detail.

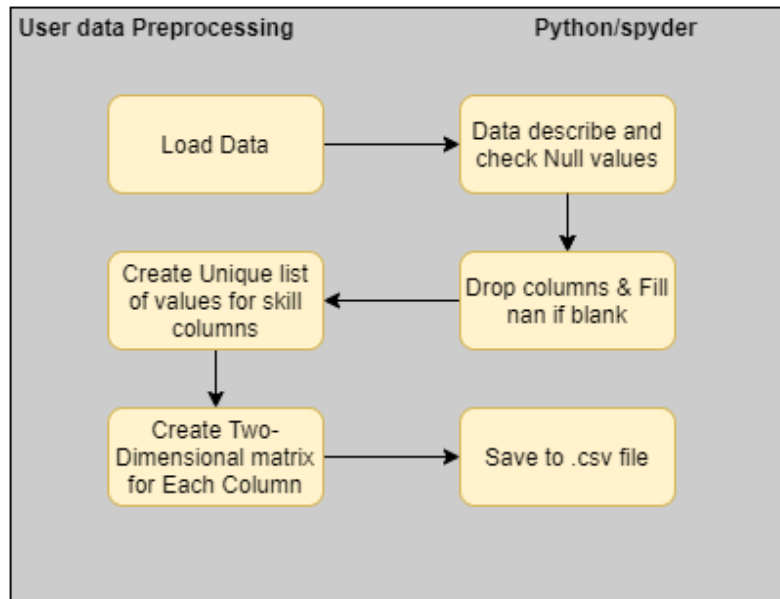


Fig. 4.9 Process flow -User Data Preprocessing

As a first step, we will load the data. Describe the loaded dataframe and check for null values. In next step we drop the columns that are of no interest to the study and replace all the blanks with numpy's 'nan'. Then we create list of unique values from targeted skill column(for ex:WebFrameDesireNextYear). Using these unique value skill list for the column 'WebFrameDesireNextYear', we will create two-dimensional matrix where each observation depicts users preference on skills from the column 'WebFrameDesireNextYear'. Fig4.10 shows the processed data for all user for the column 'WebFrameDesireNextYear'. Similarly we have to perform same activity on rest of the columns. At end of this process, we will have user preference on each skill stored in a 9 different dataframe.

```
WebFrameDesireNextYear.head()
```

	Respondent	jQuery	Other(s):	React.js	Angular/Angular.js	Vue.js	Drupal	Spring	Express	Laravel	Django	Ruby on Rails	Flask
0	1	1											1
1	2										1		
2	3		1										
3	4												
4	5	1		1				1					1

Fig. 4.10 Processed user data for column 'WebFrameDesireNextYear'

### 4.3.2 Job Data Preprocessing

While processing Job data, we will be using NLP to remove stop words, find similarity between word embedding while categorising job the based on the title and description. Then the skills in Job data is passed to clean skills function to remove symbols and store it as list. Then the list of skills of these jobs are passed to extract skill function to segregate skills into a language skill, database skill or a web framework skill. Using this data frame, we create two dimensional matrix for all type of skill, with each row depicting skill required for that job. As show in Fig 4.11.

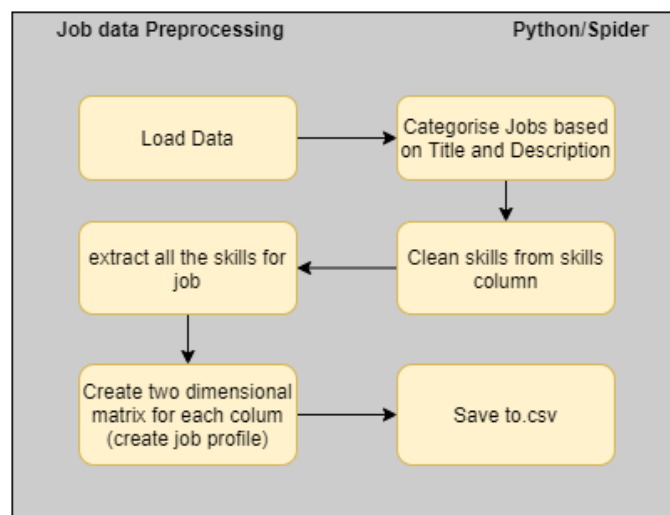


Fig. 4.11 Process flow - Job Data Preprocessing

## 4.4 Modeling of Recommender System

In modeling stage of the study, based on our literature study we had decided to proceed and implement recommender system using content based recommender system. Basic assumption of the content based recommender is that user would like to choose things that he used in the past. Adopting this assumption to a hiring domain, A job seeker would seek for a new job which needs skills that are relevant to his job skills and title. In content based filtering, all information related to user and item are supposed to be stored in a user profile vector and item profile vector. Then the similarity measures are like cosine similarity measure is used to determine the score of the job which is relevant to user's skills. Fig4.12 shows the process flow of the content-based recommender system. From the Fig.4.12, The process

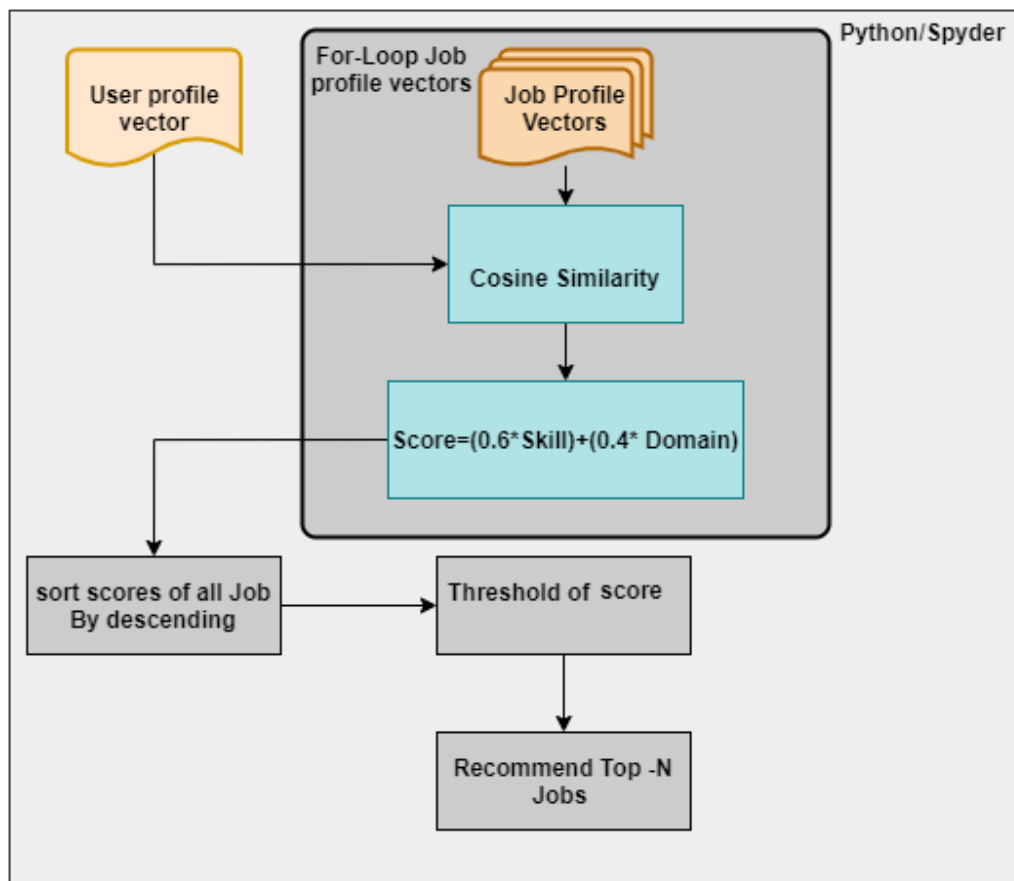


Fig. 4.12 Process flow - Recommender system

flow shows that there are five different process in the recommender system of this study. When user inputs his User\_ID, User's profile vector for that particular User\_ID is loaded. Then enter for-loop to perform the computation of cosine similarity measure for each job profile vector against the selected user profile vector. At end of each loop , the computation of similarity score is performed and recorded in the list type. once a vector of user skill and job skill is computed using cosine similarity, result will be used to compute the score by adding in all the skill similarity value with weight 0.6 which is then added with the domain similarity with weights of 0.4. In the study we have assumed that skill plays a major role in making decision on a job when compared to domain. So, additional weights of 0.6 to the skill similarity value and 0.4 to domain skill similarity value has been added to the equation as shown in the below equation 4.1

$$Job\ Score(U, J) = (0.6 * \sum_{i=1}^n Sim\ Dist_{skill}(U, J_i)) + (0.4 * \sum_{i=1}^n Sim\ Dist_{domain}(U, J_i)) \quad (4.1)$$

Where,

n = Total number of Jobs

U = Profile Vector of User

J = Profile Vector of Job

*Sim Dist*(U, J<sub>i</sub>) = Cosine distance equation as shown in equation 4.2

$$Similarity\ distance(A, B) = 1 - \cos(\theta) = 1 - \frac{\sum_{i=1}^n A_i * B_i}{\sqrt{\sum_{i=1}^n A_i^2} * \sqrt{\sum_{i=1}^n B_i^2}} \quad (4.2)$$



For- Loop ends with storing all the job similarity score for a user\_id in the list which is then sorted by similarity value in descending order. Once the Job similarity score is acquired, we can recommend the user with jobs using one of the two approaches as below.

1. Top-N approach
2. Rating Scale approach

In this study, The sorted list of job similarity value is used to filter the job by setting the threshold to 40% of the highest job similarity score. Then used the Top-N approach to recommend job to user. This whole process is implement using the python environment with the help of spyder IDE. Output of this modeling stage is handed off to further stages such as deployment and evaluation.

## 4.5 Evaluation

In Evaluation phase of this study, recommender system designed in support of cosine similarity need to be evaluated. As we use a threshold based approach to recommend the jobs from the retrieved matches based on user preference, We will evaluate the model by iterating over several user. The selection of user for the evaluation this evaluation is random. We use the random user details to generate several recommendation, compute the values coverage, precision, recall and F1 score for that particular recommendation. With the same user details, We try to compute the precision, recall and F measure for different value of threshold. Coverage is the value that represents the percentage of item that a recommender system was able to recommend out of I items. To calculating the value coverage for items recommended to the user u in the list of U can be put be put into notation as below equation.

$$Coverage = \frac{n}{N} * 100 \quad (4.3)$$

where,

n=number of items recommended

N=Number of items in the list

For this study we are considering to set the threshold value ranging from 0.5 to 0.2 which is step by 0.1. Threshold is calculated based on the maximum score in the recommended list generated from the recsys module in second tier of the design. Below is the equation for calculating threshold value of the score for list of recommendation.

$$\text{Threshold score} = \max(\text{Job Score}) * C \quad (4.4)$$

where,

Job score = List of Job matches along with similarity score as in the equation 4.1

C = Values ranging from 0.5 to 0.2

This threshold score generated is used to subset the jobs that are has the job score greater than the value of the threshold score for the iteration. The number of items in that subset after the filter is considered as True positive and subset of matches that were removed from the list of recommendation is considered as False positive. Number of matches with score which is equal to zero is considered as true negative. These are the jobs that are not similar to the user's preference. As there is no opportunity for the user to select the jobs that are not recommended by the model the value for the false negative will be constantly zero. As it is trivial to achieve recall of 100%, when a query returns all documents. hence we cannot just rely on recall. So we need to calculate precision as well. It is always desired to have high

precision and high recall value but both matrices are inversely related, so it's good evaluate system using F -measure. Following is the equation for the  $F_\beta$  measure.

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} \quad (4.5)$$

To calculate the harmonic mean between the precision and recall, we need to compute  $F_\beta$  measure with  $\beta = 1$ . Below is the equation used to compute  $F_1$ - measure,

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (4.6)$$

As discussed earlier in this evaluation phase, we have taken a random user for the evaluation purpose and recorded the precision and F1 score of the model at four different threshold value. Below is the table depicting the result for User ID 7.

Table 4.1 F1 score - User 7

Threshold	User 7	
	Precision	F1-score
0.2	0.93	0.96
0.3	0.49	0.65
0.4	0.49	0.65
0.5	0.35	0.51

Table 4.2 F1 score - User 25

Threshold	User 25	
	Precision	F1-score
0.2	0.66	0.79
0.3	0.37	0.54
0.4	0.24	0.4
0.5	0.12	0.21

Similarly we can compute the score for another user such as User 25, table 4.2 gives information about user 25. Same procedure is conducted for multiple user which were selected at random and queried for the recommendation for that particular user id with varying threshold. With collected result of precision and F1 score for multiple user, We measured the average precision and Average F1 score for the model. The result of average precision and F1 score is given in table 4.3.

Table 4.3 Average F1 score from sample set of users

Users	Average	
	Precision	F1-score
5	0.505	0.638
6	0.463	0.598
7	0.565	0.693
1	0.493	0.590
10	0.385	0.528
25	0.348	0.485
Avg from Users	0.46	0.59

Table 4.4 F1 score at threshold

Threshold	Average	
	Precision	F1-score
0.2	0.82	0.89
<b>0.3</b>	<b>0.50</b>	<b>0.66</b>
0.4	0.32	0.48
0.5	0.19	0.32

Based on the results recorded in the above tables, the best threshold value for recommending is **0.3**. The average precision recorded from all the user is 0.46. Similarly, the average F1-score recorded from all the user is 0.59. The recommendation needs to be diverse in nature, selecting 0.5 as a threshold would reduce the diversity of the recommendation to user and selecting 0.2 would include almost everything which is more diverse. So the balanced threshold value for the model would be 0.3.

## 4.6 Deployment

In this last phase of the CRISP-DM, we interact with third tier of the three tier design of the recommender system. Third tier of the design is our presentation layer where sends request for the recommendation and receives the top 10 recommendation for that particular request. This study makes use of the three tier design, where first tier is our data layer where all the data related to user profile and item profile is stored. When user sends a request for the recommendation, presentation layer interacts with the second tier to process the request.

First and second Tier of the design work in Python environment using Spyder IDE. whereas, Third tier is written in python environment which is integrated with the plotly's dash web framework for python. It is written on top of Flask, Plotly.js, and React.js, So this can be used develop custom user interface in python. Dash app render itself in clients web browser. So the dash web-app can be deployed onto a server and share URL with users of the product to access it. As dash apps open in web browser, the product developed using this will be a platform independent. Below is code snippet to install and import dash components to python environment.

```
>>>pip install dash==1.12.0
>>> import dash_core_components
>>> print(dash_core_components.__version__)
```

Listing 6 Install packages required for Web-app Deployment

When Web-app is initialised in local host server, We arrive at the below shown page (start page /landing page), Where user need to enter the 'USER\_ID NUMBER'.Figure 4.13 shows the screenshot of the initial page or landing page of the Job recommender web application. Further, When we enter a User ID Number(for example 656), The value is captured from html's input tag and passed to the recommendation package. After loading all user file and Job details file, We will check for similarity between user's (Job domain , languages and tools he has worked with) and each and every Job listings in the dataset (Job domain, languages

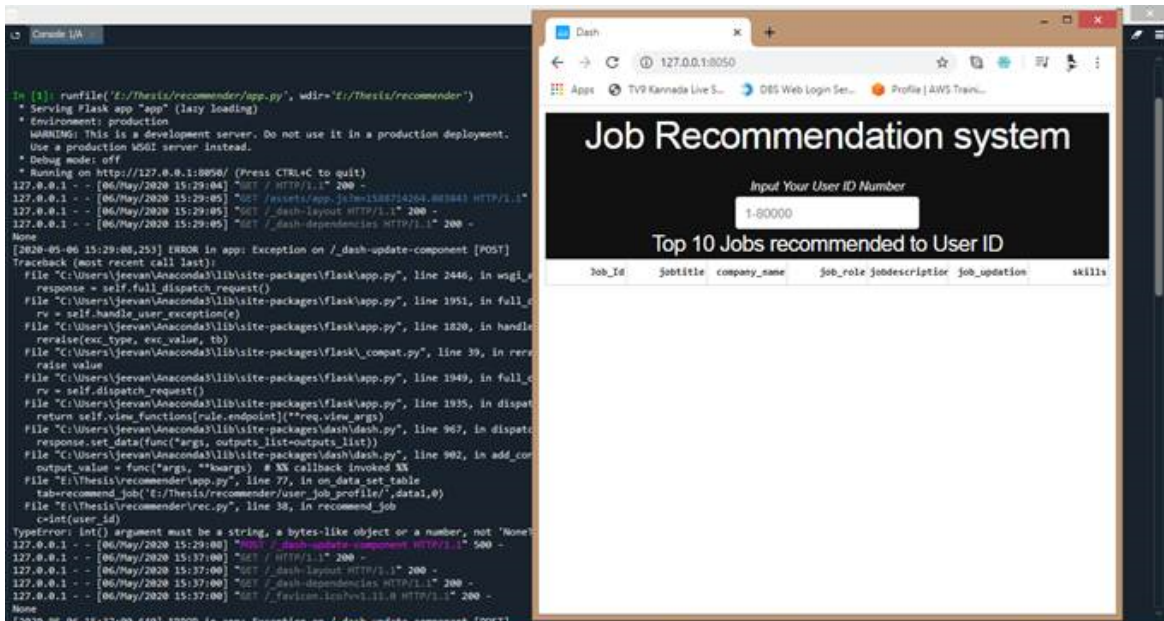


Fig. 4.13 Initial page of webapp

that the job require) using the modeling techniques that was discussed earlier. Then take similarity score between user and jobs . Then recommend Top N Jobs that are similar to user’s profile. The screenshot of the recommend jobs to the user 656 is shown in figure 4.14

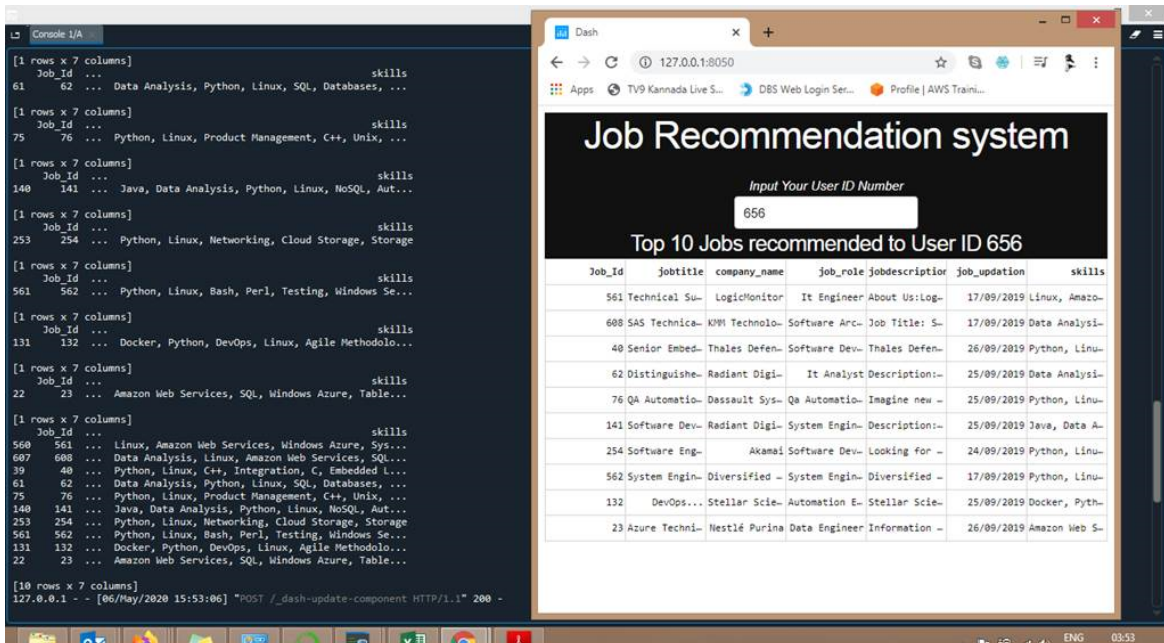


Fig. 4.14 Recommendation page of webapp

# Chapter 5

## Results

In this chapter, the results of job recommender system which are run in the python environment are presented. Recommender system is provided with the profile vector of user and item to generate a set of item as a recommendation to user in question. Profile vector of a user holds the information regarding the preferences of user on IT skill and domain specific information. Where as, profile vector of item holds the information regarding the skills and job domain that is required for that job. Also as defined in section 1.4, the purpose of this research was not only model a job recommender system using user's skill set and Job domain but also to address issue of cold start. To complete the research, we had set ourselves a objectives which ease our way in completion of the research.

As to conduct the research, the need for two different dataset, which will be used create a user vector and item vector was required. For the purpose of user data we had choose the data from stack overflow survey; Without the Job data set, we wouldn't be able to continue this research. So, In this study, the task of web scraping was performed on several techniques and finally opted to continue the study by scraping data using R programming in R-studio environment. This satisfied our first objective of our study. Further in this project, before starting with implementation, we had to perform the data preprocessing on the both user dataset from stack overflow and the job listing dataset , scraped from the job board. The

main objective was to create matrix of user and item which describes the user and item about their implicit attribute. Using state of the art technology like Natural language processing to analyze the implicit properties of the item and check similarity between the job domain by using word embedding and cosine similarity techniques offered by the spaCy package in python satisfies our second objective of our study.

In the third objective of the study, we were supposed to address the cold start issue of the recommender system. Due to the fact the both the data sets are from acquired from different source. There could be no interaction between the user and job data set chosen for the study, With the minimal interaction with between user and job data made it hard to collaborative filtering as it requires previous interaction. When the user's and Jobs doesn't have any predefined relation it also mean they are new to the system. So we decided to perform data preprocessing on data to transform it into user and job profile vector which describes attributes of user and job. These profile vector and use of content based filtering in this Recsys allowed us to avoid the issue of cold start from both new users and new jobs perspective.

The final objective of the study was to use the results of previous objectives and implement a recommender system to recommend jobs from the list of job dataset for a particular user based on the user profile vector; Which includes the details such as what language user would like to work on, what frameworks he has worked on, what was his role or domain of his work. This information is utilised to check similarity between the job profile vector. This led to generation of score against each job. The score is filtered using the rating scale approach, where we set a particular threshold value and subset the recommendation list by considering jobs with score greater than threshold value. To select the threshold, we performed the evaluation by taking random user for analysis of best threshold value for the recommendation. Based on the evaluation as discussed in the section 4.5, The suited threshold value for the recommender system is 0.3. The table 4.4 provides insight on the evaluation conducted



on different users that led to choosing of threshold value. Also another metric we used to determine the how good is recommender model was by using the metric called coverage. Coverage provides an information on what percentage of items were chosen to recommend items to a single user. This method is used on multiple user by randomly selecting the user and computed the value of coverage in each instance to calculate the average coverage of item in this recommender system. Below is the table 5.1 that provides insight on coverage and F1 on each instance and the average coverage value for the completion of the process.

Table 5.1 Average Coverage and F1 score

Users	Average	
	Coverage	F1-score
5	0.64	0.64
6	0.64	0.60
7	0.22	0.69
1	0.49	0.59
10	0.69	0.53
25	0.49	0.49
Avg	0.53	0.59

Based on above table, the recommender system is providing the coverage of 0.53 in average on every request of Job recommendation with F1- score being 0.59 in average for the system based on the evaluation conducted by this study. Further, the list of recommendation is pushed to the front-end of the user interface , where the request was generated for the job recommendation. Following is the result of deployment done in the local host using the plotly's dash framework which is written on top of flask a micro web framework.

The python file for the web-app initialisation is run from the console, to start the flask server where the job recommendation system's user interface is deployed. At start up of the app, we will receive message, as shown in figure 5.1

As the job recommendation web application is deployed in the local host URL of the system, we have to access the local host using the URL <http://127.0.0.1:8050/> through a web

```

Python 3.7.4 (default, Aug 9 2019, 18:34:13) [MSC v.1915 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.8.0 -- An enhanced Interactive Python.

In [1]: runfile('E:/Thesis/recommender/app.py', wdir='E:/Thesis/
recommender')
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a
  production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:8050/ (Press CTRL+C to quit)

```

Fig. 5.1 Web application initiation

browser. when a the URL of the web-app is loaded, we land in the home page of the web app as below figure 5.2, Home page has input field at the top, where user need to enter the

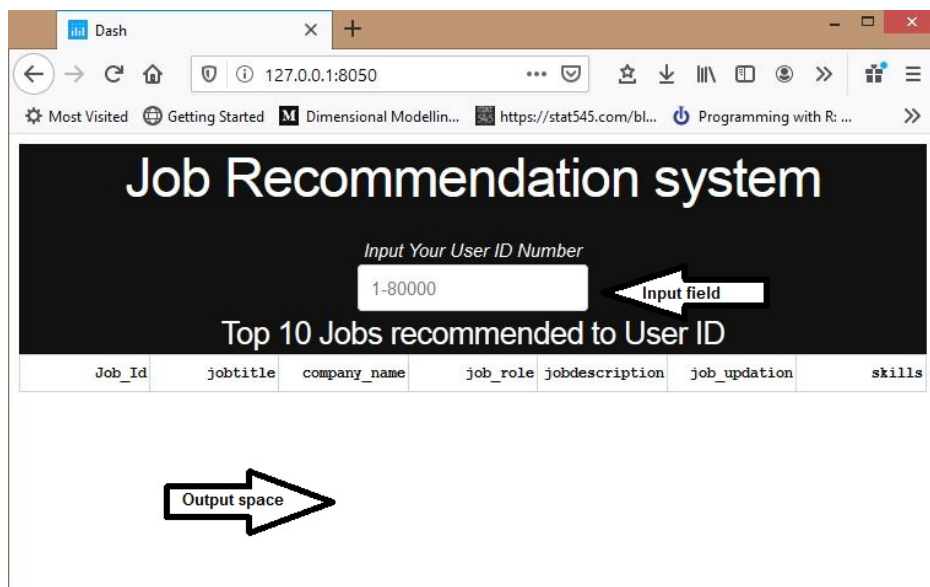
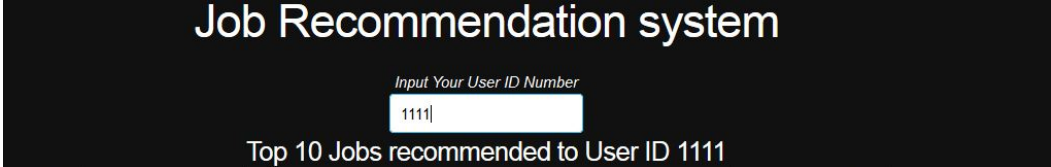


Fig. 5.2 Web application home page

user id number , which represents the user's profile vector in the data layer. The bottom

space in the home page is output space where user will see the list of recommendation as shown in the figure 5.2. Once the user input user Id and hit enter, Home page will connects to python function in back-end to pass the values, and retrieve the list of recommendation for that particular user as shown in image 5.3



Job_Id	jobtitle	company_name	job_role	jobdescription	job_updateion	skills
1	System Administrator	IP Pathways	Systems Administrator	Title Systems Admini...	26/09/2019	SQL, Networking, Vir...
2	Project Manager / Bu...	Headfarmer	Project Manager	Responsibilities:Fac...	26/09/2019	Agile Methodologies,...
3	SENIOR SOFTWARE ARCH...	Quicken Loans	Software Architect	Quicken Loans is loo...	26/09/2019	Javascript, Java, Do...
4	Senior Mulesoft Deve...	Healthfirst	Software Developer	Duties & Responsibil...	26/09/2019	Java, SQL, Web Servi...
5	Full-Stack Developer	Xanadu	Software Developer	Full Time   Toronto,...	26/09/2019	Javascript, HTML, Do...
6	Sr DevOps Engineer	Healthfirst	Devops	The Sr. DevOps, Appl...	26/09/2019	Javascript, Docker, ...
7	Salesforce Developer	Healthfirst	Software Developer	The Salesforce Devel...	26/09/2019	Salesforce Sales Clo...
8	Software Engineer	Rite Hire	Software Developer	Core Java Developmen...	26/09/2019	Java, Python, Scala
9	Senior Software Engi...	Roblox	Back End Developer	WHY ROBLOX?Roblox is...	26/09/2019	Java, Python, .NET, ...
10	Assemble & Repair Te...	Radiant Digital	Hardware Engineer	Description:Under st...	26/09/2019	Analysis, Manufactur...

Fig. 5.3 Web application Result page

If a user makes a changes in a profile, it would affect the profile vector and recommends the job that are similar to profile vector. The new recommendations are made to the user, when new jobs are added top the data layer, the job vector gains high similarity score. Above discussion gave a insights on the presentation layer of the job recommendation system.

# Chapter 6

## Future work and Discussion

### 6.1 Future work

Based on the current study, the recommendation system works on the content-based filtering using word embedding of word2vec and similarity measure of cosine similarity. As the corpus provides general information about the word and similar words around it, It is possible to create a better recommendation by creating a corpus related to the IT skills, terminology, Job domain and jargon of the industry. By using such corpus specific to the hiring domain, the recommendation could be better when analyzing implicit text data in the job description. It can be categorized in a better way. As this Recsys is currently working on data that has no interaction, a study needs to be conducted on the data that has previous interaction in the hiring domain. This would allow us to dynamically keep recommending new jobs based on user's change in preferences.

There is a recommender system in the hiring domain from LinkedIn but not in the perspective of a job seeker but from the perspective of a recruiter. Similarly, we could conduct a study based on LinkedIn data to recommend jobs using content-based filtering. As conditions change from domain to domain, it is not a good idea to recommend a job because a user liked it; instead, the recommendation has to be considered, if the profile of a

user matches the requirement. So, conducting more study based on content-based filtering ensemble with other filtering technique in hiring domain in the perspective of a job seeker can be considered as a part of future work.

## 6.2 Discussion

This study on recommender system in the field of the hiring domain concentrates on analysing the skills required for the job, to which domain user fall into; using this as a parameter to compute the similarity between available position and user. The available recommender system in the domain of news or the field of entertainment relies on user interaction. Interaction such as ratings provided by the user on a particular item, to make an item recommendation to a user but this concept of ratings and predicting the likelihood of a user to choose an item would be incorrect when it is viewed in the perspective of job domain or recruiter.

Implementation of a recommender system that is based on ratings, number of views and popularity of an item in the job domain would allow the user to apply most of the job that he sees online. However, this would hamper the process of hiring due to the clogging of profile at the recruiter end. In this study, we recommend the job that is similar to the user profile by analysing the user preference of the user using content-based filtering. Using this process of recommendation would efficiently make the user apply only to the jobs that he might be suited to, instead of applying to most of the jobs that are available in the system. This recommender system would ease the burden of a recruiter by reducing the number of irrelevant applicants.

# Chapter 7

## Conclusion

Therefore, We conclude that job recommendation system with analysis of job description to recommend a job based on user's skills and preferences presents itself as worthy Recsys model in recommending open position to the job seekers when looking for a new positions. Thus, among the different threshold and filtering techniques, we chose to model the recommender system using content-based filtering which is achieving F1-score of 66% with the threshold of 0.3 with average coverage of 53%.

# References

- Al-Otaibi, S.T. and Ykhlef, M. (2012) Job recommendation systems for enhancing e-recruitment process in: *Proceedings of the International Conference on Information and Knowledge Engineering (IKE)* p. 1 The Steering Committee of The World Congress in Computer Science, Computer ...
- Barrón-Cedeno, A., Eiselt, A. and Rosso, P. (2009) Monolingual text similarity measures: A comparison of models over wikipedia articles revisions *ICON 2009*, pp. 29–38
- Barzilay, R. and Elhadad, N. (2003) Sentence alignment for monolingual comparable corpora in: *Proceedings of the 2003 conference on Empirical methods in natural language processing* pp. 25–32 Association for Computational Linguistics
- Brants, T. (2003) Natural language processing in information retrieval. in: *CLIN* Citeseer
- Brownlee, J. (2017) What are word embeddings for text?
- Burke, R. (2002) Hybrid recommender systems: Survey and experiments *User modeling and user-adapted interaction* **12**(4), pp. 331–370
- Burke, R. (2007) Hybrid web recommender systems in: *The adaptive web* pp. 377–408 Springer
- Dhameliya, J. and Desai, N. (2019) Job recommender systems: A survey in: *2019 Innovations in Power and Advanced Computing Technologies (i-PACT)* vol. 1 pp. 1–5 IEEE
- Herlocker, J.L., Konstan, J.A., Borchers, A. and Riedl, J. (1999) An algorithmic framework for performing collaborative filtering in: *22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1999* pp. 230–237 Association for Computing Machinery, Inc
- Herremans, D. and Chuan, C.H. (2017) Modeling musical context with word2vec *arXiv preprint arXiv:1706.09088*
- Hu, R. and Pu, P. (2011) Enhancing collaborative filtering systems with personality information in: *Proceedings of the fifth ACM conference on Recommender systems* pp. 197–204
- Jain, H. and Kakkar, M. (2019) Job recommendation system based on machine learning and data mining techniques using restful api and android ide in: *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* pp. 416–421 IEEE

- Jijkoun, V., de Rijke, M. *et al.* (2005) Recognizing textual entailment using lexical similarity in: *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment* pp. 73–76 Citeseer
- Kakarla, S. (2019) Natural language processing: Nltk vs spacy
- Kenter, T. and De Rijke, M. (2015) Short text similarity with word embeddings in: *Proceedings of the 24th ACM international on conference on information and knowledge management* pp. 1411–1420
- Luk, K. (2019) Introduction to two approaches of content-based recommendation system
- McAlone, N. (2016) Why netflix thinks its personalised recommendation engine is worth \$1 billion per year
- Mihalcea, R., Corley, C., Strapparava, C. *et al.* (2006) Corpus-based and knowledge-based measures of text semantic similarity in: *Aaai* vol. 6 pp. 775–780
- Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013) Efficient estimation of word representations in vector space *arXiv preprint arXiv:1301.3781*
- Mobasher, B. (2007) Data mining for web personalization in: *The adaptive web* pp. 90–135 Springer
- overflow, s. (2019) Stack overflow insights - developer hiring, marketing, and user research
- Rafter, R., Bradley, K. and Smyth, B. (2000) Personalised retrieval for online recruitment services in: *The BCS/IRSG 22nd Annual Colloquium on Information Retrieval (IRSG 2000), Cambridge, UK, 5-7 April, 2000*
- Recsys (2012) Recommender systems-how they works and their impacts: Content-based filtering
- Ricci, F., Rokach, L. and Shapira, B. (2011) Introduction to recommender systems handbook in: *Recommender systems handbook* pp. 1–35 Springer
- Richardson, L. (2004)
- Rong, X. (2014) word2vec parameter learning explained *arXiv preprint arXiv:1411.2738*
- Shearer, C. (2000) The crisp-dm model: the new blueprint for data mining *Journal of data warehousing* **5**(4), pp. 13–22
- Shrestha, P. (2011) Corpus-based methods for short text similarity
- Slamet, C., Andrian, R., Maylawati, D.S., Darmalaksana, W., Ramdhani, M. *et al.* (2018) Web scraping and naïve bayes classification for job search engine in: *IOP Conference Series: Materials Science and Engineering* vol. 288 p. 012038 IOP Publishing
- Sternitzke, C. and Bergmann, I. (2009) Similarity measures for document mapping: A comparative study on the level of an individual scientist *Scientometrics* **78**(1), pp. 113–130



stillman, J. (2019) New harvard research: To be successful, chase your purpose, not your passion | inc.com [online] <https://www.inc.com/jessica-stillman/new-harvard-research-to-be-successful-chase-your-purpose-not-your-passion.html>

SysNucleus (2019) Webharvy web scraper