

**18XW86 - INFORMATION RETRIEVAL LAB  
PACKAGE DOCUMENTATION**

**GOMATHI K (19PW10)  
KEERTHANAH M K (19PW11)**

---

**PROJECT TITLE: JOB SEARCH**

**ABSTRACT:**

Job recommendation systems in information retrieval are designed to assist job seekers in identifying suitable employment opportunities based on their skills, qualifications, and experience. These systems use machine learning algorithms and data mining techniques to analyze job postings and resumes to identify matches between job requirements and candidate profiles.

Along with the recommendation of jobs, we have developed a user interface that showcases all the results in a tabular manner using Django, HTML, CSS and Bootstrap.

**METHODS:**

The recommendation system typically consists of three components: data collection, data preprocessing, and recommendation algorithms. Data collection involves gathering job postings and resumes from various sources such as job boards, social media, and company websites. Data preprocessing involves cleaning and transforming the data into a format that can be analyzed by the recommendation algorithms. The recommendation algorithms use various techniques such as content-based filtering, and collaborative filtering to identify potential job matches.

In content-based filtering, the system identifies relevant job postings based on the skills and qualifications mentioned in the candidate's resume. Collaborative filtering involves recommending jobs based on the candidate's behavior, such as job search history or previous job applications.

We have also performed web scraping using Selenium as an application for referral system.

The steps involved in building job recommendation systems by considering the content-based approach:

1. Spacy, NLTK and scikit-learn libraries are used
2. Dataset Link:  
<https://www.kaggle.com/datasets/kandij/job-recommendation-datasets>
  - a. Job\_Views:
    - Contains details about the job the applicants are seeking
  - b. Experience
    - Contains the experience from the applicant
  - c. Positions\_Of\_Interest
    - Contains the interest the user previously has manifested
  - d. Combined\_Jobs\_Final
    - Contains main jobs data(title, description, company, etc.)
3. Cleaning and building datasets: This step involves collecting data from various sources, removing any irrelevant or erroneous data points, and structuring the data in a way that can be easily processed by the recommendation algorithm (merge data to have 1 dataset for jobs, and 1 dataset for users)
4. Numerical feature extraction: In this step, we extract relevant features from the data, such as the job title, job description, job location, required skills, etc. These features will be used as input to the recommendation algorithm
5. As a part of preprocessing, we perform -
  - Imputation: This step involves filling in missing values in the data with estimates based on the available data. In the context of job descriptions, missing values might arise when certain fields are not provided by the job posting website or when some job postings are incomplete.

- Stop word removal: Stop words are words that occur frequently in a language, such as "the", "and", "of", etc. These words are often irrelevant to the meaning of a sentence or document and can be removed to reduce the dimensionality of the text data and improve the accuracy of the analysis.
- Non-alphanumeric character removal: Removing non-alphanumeric characters, such as punctuation marks and symbols, is another common preprocessing step that helps to simplify the text data and reduce noise.
- Lemmatization: Lemmatization is the process of reducing words to their base or dictionary form. For example, the word "ran" would be lemmatized to "run". This step helps to normalize the text data and reduce the number of unique words in the corpus.
- Corpus creation: Finally, the preprocessed text data is combined into a single corpus or document for each job posting. This allows the data to be analyzed as a whole and enables similarity measures, such as cosine similarity, to be calculated between the job postings and user preferences.

6. As this use case has more textual data and there are no numerical data like ratings available for any job, we are not matrix decomposition methods or correlation coefficient-based methods.

Hence we use content based filtering which will recommend jobs to people based on the attributes of the jobs only

We build 4 recommenders systems:

- A. Content based Recomender with Tf idf
- B. Content Based Recomender with CountVectorizer
- C. Content Based Recomender with Spacy
- D. Content Based Recomender with KNN

- Content based Recomender with Tf idf:

This recommender system uses the term frequency-inverse document frequency (tf idf) method to extract features from the job descriptions. It then recommends jobs to users based on the similarity between the job description features and the user's job preferences

- Content Based Recomender with CountVectorizer:

This recommender system uses the CountVectorizer method to extract features from the job descriptions. It then recommends jobs to users based on the similarity between the job description features and the user's job preferences

- Content Based Recomender with Spacy:

This recommender system uses Spacy, a natural language processing library, to extract features from the job descriptions. It then recommends jobs to users based on the similarity between the job description features and the user's job preferences.

In IR, SpaCy can be used to preprocess and clean text data, extract relevant information, and perform various analyses on the processed data. For example, it can be used to tokenize and lemmatize text documents, identify named entities such as people, organizations, and locations, and extract features such as part-of-speech tags and syntactic dependencies. These features can then be used to train machine learning models for tasks such as document classification and entity recognition.

- Content Based Recomender with KNN:

This recommender system uses the K-Nearest Neighbor algorithm to find similar jobs based on the attributes of the job descriptions. It then recommends jobs to users based on the similarity between the job description features and the user's job preferences

- *Similarity function*: A similarity function measures the similarity between two sets of data points. In this case, we use cosine similarity to measure the similarity between the user's previous job history or expressed job interests and the available jobs. Cosine similarity is a commonly used similarity function that calculates the cosine of the angle between two vectors, which represents their similarity
- *Top recommendation*: Once the similarity scores between the user's previous job history or expressed job interests and the available jobs are calculated, the jobs are sorted based on their similarity score and present the top recommendations to the user.

#### 7. UI:

- We have used Django, HTML, CSS and Bootstrap to visualize the recommender system
- We have specifically designed the UI to visualize the recommendations using Tf idf, CountVectorizer and KNN
- The home page of the UI looks like the image given below



## Job Search

Tf-Idf
Count Vectorizer
KNN

## 8. Content based Recomender with Tf idf:

### a. Output in Google Colab:

The top recommendations using TF-IDF

```
[ ] top = sorted(range(len(output2)), key=lambda i: output2[i], reverse=True)[:10]
list_scores = [output2[i][0][0] for i in top]
get_recommendation(top,df_all, list_scores)
```

	ApplicantID	JobID	title	score
0	326	303112	Java Developer @ TransHire	0.749478
1	326	294684	Java Developer @ Kavaliro	0.740886
2	326	269922	Entry Level Java Developer / Jr. Java Develop...	0.737007
3	326	141831	Lead Java/J2EE Developer - Contract to Hire @ ...	0.671667
4	326	270171	Senior Java Developer - Contract to Hire - Gre...	0.645037
5	326	305264	Sr. Java Developer @ Paladin Consulting Inc	0.625532
6	326	309945	Java Software Engineer @ iTech Solutions, Inc.	0.592291
7	326	245753	Java Administrator @ ConsultNet	0.530231
8	326	146640	Jr. Java Developer @ Paladin Consulting Inc	0.510534
9	326	150882	Java Consultant - Mobile Apps Development @ Co...	0.486789

### b. Output in UI:

Recommendation			
ApplicantID	JobID	Title	Score
326	303112	Java Developer @ TransHire	0.7494775251900346
326	294684	Java Developer @ Kavaliro	0.7408859264648847
326	269922	Entry Level Java Developer / Jr. Java Developer - Contract to Hire @ iTech Solutions	0.7370067616428033
326	141831	Lead Java/J2EE Developer - Contract to Hire @ iTech Solutions, Inc.	0.6716673937124861
326	270171	Senior Java Developer - Contract to Hire - Great Salary @ iTech Solutions, Inc.	0.6450373772174958
326	305264	Sr. Java Developer @ Paladin Consulting Inc	0.6255315736496132
326	309945	Java Software Engineer @ iTech Solutions, Inc.	0.5922913263911141
326	245753	Java Administrator @ ConsultNet	0.5302305793452344
326	146640	Jr. Java Developer @ Paladin Consulting Inc	0.5105340708444382
326	150882	Java Consultant - Mobile Apps Development @ Consultis	0.4867887337500865

## 9. Content Based Recomender with CountVectorizer:

### a. Output in Google Colab:

▼ The top recommendations using CountVectorizer

```
top = sorted(range(len(output2)), key=lambda i: output2[i], reverse=True)[:10]
list_scores = [output2[i][0][0] for i in top]
get_recommendation(top, df_all, list_scores)
```

	ApplicantID	JobID	title	score
0	326	303112	Java Developer @ TransHire	0.635001
1	326	294684	Java Developer @ Kavaliro	0.600245
2	326	269922	Entry Level Java Developer / Jr. Java Develop...	0.571726
3	326	141831	Lead Java/J2EE Developer - Contract to Hire @ ...	0.496907
4	326	270171	Senior Java Developer - Contract to Hire - Gre...	0.481757
5	326	309945	Java Software Engineer @ iTech Solutions, Inc.	0.454673
6	326	305264	Sr. Java Developer @ Paladin Consulting Inc	0.406017
7	326	245753	Java Administrator @ ConsultNet	0.378968
8	326	150882	Java Consultant - Mobile Apps Development @ Co...	0.363216
9	326	146640	Jr. Java Developer @ Paladin Consulting Inc	0.323381

### b. Output in UI:

Recommendation			
ApplicantID	JobID	Title	Score
326	303112	Java Developer @ TransHire	0.6350006350009525
326	294684	Java Developer @ Kavaliro	0.6002450479987809
326	269922	Entry Level Java Developer / Jr. Java Developer - Contract to Hire @ iTech Solutions	0.571726113011571
326	141831	Lead Java/J2EE Developer - Contract to Hire @ iTech Solutions, Inc.	0.49690718681906354
326	270171	Senior Java Developer - Contract to Hire - Great Salary @ iTech Solutions, Inc.	0.4817571941126949
326	309945	Java Software Engineer @ iTech Solutions, Inc.	0.4546733209087348
326	305264	Sr. Java Developer @ Paladin Consulting Inc	0.4060174087890591
326	245753	Java Administrator @ ConsultNet	0.37896836447993354
326	150882	Java Consultant - Mobile Apps Development @ Consultis	0.36321635614607395
326	146640	Jr. Java Developer @ Paladin Consulting Inc	0.32338083338177726

## 10. Content Based Recomender with Spacy:

The Top recommendations using Spacy

```
get_recommendation(index_spacy, df_all, list_scores)
```

	ApplicantID	JobID	title	score
0	326	316105	Bookkeeper @ Accountemps	0.71202
1	326	142577	Auditor @ Accountemps	0.640437
2	326	308009	Employment Specialist (PT) - Deaf & HoH @ AHEDD	0.613965
3	326	272697	AR/Collections @ Accountemps	0.60072
4	326	136432	Collections Specialist @ Accountemps	0.590401
5	326	286758	A/P Clerk @ Accountemps	0.588502
6	326	316108	Staff Accountant @ Accountemps	0.577872
7	326	273217	Staff Accountant @ Accountemps	0.577872
8	326	293877	Financial Analyst @ Accountemps	0.570692
9	326	294234	Intern @ CBRE	0.568222

## 11. Content Based Recomender with KNN:

### a. Output in Google Colab:

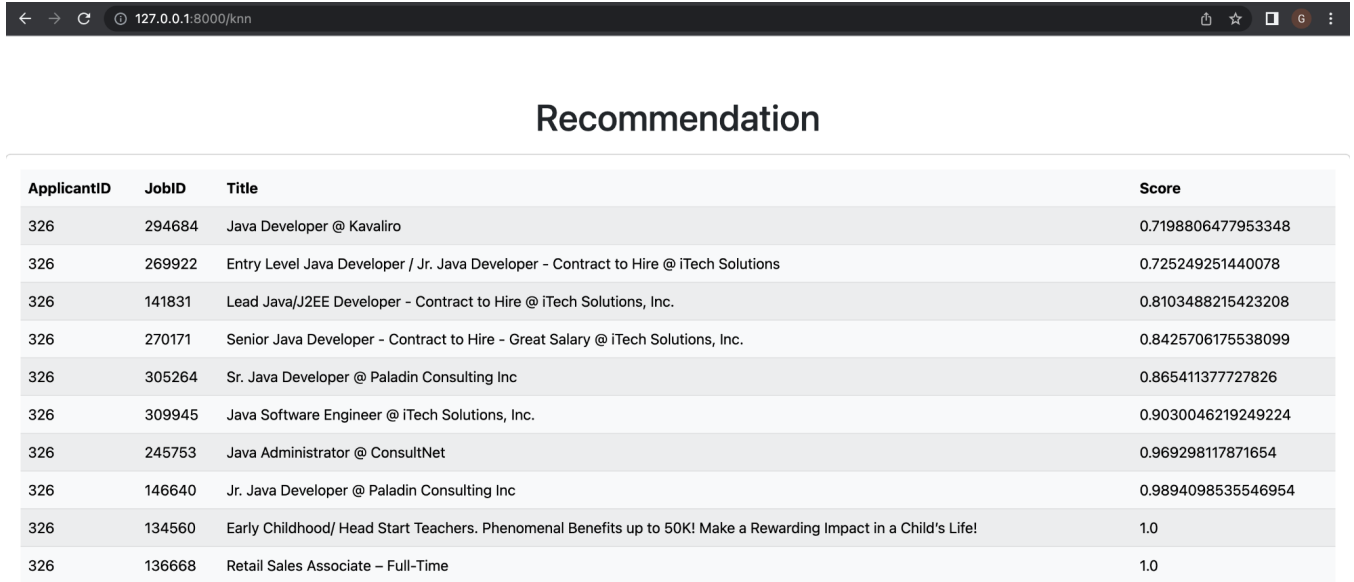
The top recommendations using KNN

```
[ ] top = NNs[1][0][1:]
index_score = NNs[0][0][1:]
get_recommendation(top, df_all, index_score)
```

	ApplicantID	JobID	title	score
0	326	294684	Java Developer @ Kavaliro	0.719881
1	326	269922	Entry Level Java Developer / Jr. Java Develop...	0.725249
2	326	141831	Lead Java/J2EE Developer - Contract to Hire @ ...	0.810349
3	326	270171	Senior Java Developer - Contract to Hire - Gre...	0.842571
4	326	305264	Sr. Java Developer @ Paladin Consulting Inc	0.865411
5	326	309945	Java Software Engineer @ iTech Solutions, Inc.	0.903005
6	326	245753	Java Administrator @ ConsultNet	0.969298
7	326	146640	Jr. Java Developer @ Paladin Consulting Inc	0.98941
8	326	134470	Bookkeeper (Part-time)	1.0
9	326	136666	T-Mobile Sales Representative	1.0



## b. Output in UI:



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000/knn'. The main content area features a table titled 'Recommendation'. The table has four columns: 'ApplicantID', 'JobID', 'Title', and 'Score'. It contains 11 rows of data, each representing a job recommendation for a specific applicant.

ApplicantID	JobID	Title	Score
326	294684	Java Developer @ Kavaliro	0.7198806477953348
326	269922	Entry Level Java Developer / Jr. Java Developer - Contract to Hire @ iTech Solutions	0.725249251440078
326	141831	Lead Java/J2EE Developer - Contract to Hire @ iTech Solutions, Inc.	0.8103488215423208
326	270171	Senior Java Developer - Contract to Hire - Great Salary @ iTech Solutions, Inc.	0.8425706175538099
326	305264	Sr. Java Developer @ Paladin Consulting Inc	0.86541137727826
326	309945	Java Software Engineer @ iTech Solutions, Inc.	0.9030046219249224
326	245753	Java Administrator @ ConsultNet	0.969298117871654
326	146640	Jr. Java Developer @ Paladin Consulting Inc	0.9894098535546954
326	134560	Early Childhood/ Head Start Teachers. Phenomenal Benefits up to 50K! Make a Rewarding Impact in a Child's Life!	1.0
326	136668	Retail Sales Associate - Full-Time	1.0

## 12. Collaborative filtering

- We have used job ratings for job recommendation based on collaborative filtering
- We ourselves have created a sample dataset to perform the same
- Collaborative filtering is a technique used in recommendation systems to predict a user's interests based on their past behaviors or the behavior of similar users. The idea behind collaborative filtering is that people who have similar preferences in the past are likely to have similar preferences in the future.
- There are two types of collaborative filtering: user-based and item-based. In user-based collaborative filtering, recommendations are made based on the preferences of similar users.
- In item-based collaborative filtering, recommendations are made based on the similarities between items.

- Collaborative filtering can be effective in recommending products, movies, music, and other types of items. However, it does have limitations, such as the cold-start problem (when a new user or item has no data), sparsity (when there is not enough data on a user or item), and the popularity bias (when popular items dominate recommendations)

### 13. Web Scraping

- In order to facilitate easier referral system, we have scrapped users from a particular position and company from LinkedIn using Selenium
- We obtain a CSV file which contains all such users
- Example: We have considered extracting all users who are Python Developers in Delhi. Finally, a CSV file containing all such users will be generated

### 14. Evaluation

- As there is no predefined testing matrix available for generating the accuracy score we need to check our relevance of our recommendations manually
- Inference: We find that the outputs of Tf idf and CountVectorizer are very similar

### **MODEL USED:**

- 1) Vector space model:
  - Tf-Idf
  - Count Vectorizer
- 2) NLP Library:
  - Spacy
- 3) ML model:
  - KNN