computers

MDPI

Article

Learning-Based Matched Representation System for Job Recommendation

Suleiman Ali Alsaif, Minyar Sassi Hidri *, Hassan Ahmed Eleraky, Imen Ferjani and Rimah Amami

Computer Department, Deanship of Preparatory Year and Supporting Studies, Imam Abdulrahman Bin Faisal University, Dammam 31441, Saudi Arabia
* Correspondence: mmsassi@iau.edu.sa

Abstract: Job recommender systems (JRS) are a subclass of information filtering systems that aims to help job seekers identify what might match their skills and experiences and prevent them from being lost in the vast amount of information available on job boards that aggregates postings from many sources such as LinkedIn or Indeed. A variety of strategies used as part of JRS have been implemented, most of them failed to recommend job vacancies that fit properly to the job seekers profiles when dealing with more than one job offer. They consider skills as passive entities associated with the job description, which need to be matched for finding the best job recommendation. This paper provides a recommender system to assist job seekers in finding suitable jobs based on their resumes. The proposed system recommends the top-n jobs to the job seekers by analyzing and measuring similarity between the job seeker's skills and explicit features of job listing using content-based filtering. First-hand information was gathered by scraping jobs description from Indeed from major cities in Saudi Arabia (Dammam, Jeddah, and Riyadh). Then, the top skills required in job offers were analyzed and job recommendation was made by matching skills from resumes to posted jobs. To quantify recommendation success and error rates, we sought to compare the results of our system to reality using decision support measures.

Keywords: job recommender system; content-based filtering; web scrapping; job matching; machine learning; indeed; natural language processing

Citation: Alsaif, S.A.; Sassi Hidri, M.; Eleraky, H.A.; Ferjani, I.; Amami, R. Learning-Based Matched Representation System for Job Recommendation. *Computers* **2022**, *11*, 161. https://doi.org/10.3390/computers11110161

Academic Editors: Phivos Mylonas, Katia Lida Kermanidis and Manolis Maragoudakis

Received: 20 October 2022
Accepted: 7 November 2022
Published: 14 November 2022

1. Introduction

Online recruiting websites or job boards, such as Indeed or LinkedIn become one of the main channels for people to find jobs. These web platforms have provided their services for more than ten years and have saved a lot of time and money for both job seekers and organizations who want to hire people.

To serve the constant cycle of the hiring process from the job seeker's perspective, many job companies have come up with solutions for providing the job board. Here, a seeker searches for the job they would find relevant to them 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 resume, creating a job profile, and recommending new jobs to a job seeker.

Since the number of results returned to job seekers may be huge, they are required to spend a significant amount of time reading and reviewing their options. That is why traditional information retrieval (IR) techniques may not be appropriate for users.

To improve traditional IR techniques, several published works consider skills as keywords associated with the job description, which needs to be matched for finding the best job recommendation (e.g., [1,2]). They consider user–job dyadic data for modeling recommendations using both content-based and collaborative filtering-based methods [1]. For these models, skill is not an active entity, they are used indirectly as keywords in the job profile and user profile.

Job recommendation has also been performed by building generative models using sequences of the job history of many users [2]. Such a model only uses job history data ignoring skills associated with a given job.

Some works focused on the influence of feature selection (FS) [3] and different forms of FS optimization [4–10] and on job clustering for an e-recruitment recommender system.

In [11], the authors described the job recommendation framework of the integration of migrants' matcher service. They capture their features including expectations, job history, and skills in an ontology to facilitate matching. In [12], they addressed the dynamic nature of the job market leads to cold start and scalability issues and they proposed a deep semantic structure algorithm that overcomes the issue of the existing system.

To track suitable job opportunities, the authors in [11] proposed to automate the tracking process to eliminate this problem. They aggregated and recommended appropriate jobs to job seekers, especially in the engineering domain.

The majority of the proposed works consider skills as passive entities associated with the job description, which need to be matched for finding the best job recommendation.

The proposed question 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?"*

To answer this question, we need to devise a learning model which recommends a job to the job seeker based on resume content.

Through this work, we tried to help job seekers identify suitable job offers based on the skills in their resume and their relation to the required skills in the job offer.

The main contributions of this article can be summarized as:

- Analyzing the skills required for a job, into which the user's domain falls; using this as a parameter to compute the similarity between available job offers on Indeed and job seekers;
- A web-based application enabling recommendation job postings that match job seekers skills reflected in their resumes;
- The recommender system uses NLP (Natural language processing) to match skills between resumes and job descriptions.

The proposed system is evaluated using a job offers dataset scrapped from three cities in Saudi Arabia (Dammam, Jeddah, and Riyadh) from the job board platform Indeed. The performance of generated recommendations is evaluated using machine learning (ML) decision support measures such as accuracy, precision, recall, and F1-score.

The remainder of this paper consists of seven sections. Methods in JRS are highlighted in Section 2, followed by evaluation methods in JRS in Section 3. The methods used for building the recommender model are presented and subsequently described in Section 4. Results are presented in Section 5, followed by the proposed 3-Tier design for JRS in Section 6. A summary of the paper's contributions and future directions are presented in Section 7.

2. Recommender Systems Methods

Customized recommendation sites implemented a variety of types of recommender systems, such as Collaborative Filtering (CF), Content-Based Recommender Systems (CBR), and Hybrid Recommender Systems (HRS).

In CF, recommendations are based solely on behavioral data, typically stored in a user items rating matrix. In the e-recruitment setting, as in the e-commerce setting, this matrix is usually filled with click behavior (e.g., the rating matrix equals 1 if the job seeker (row) clicked on a vacancy (column) to see the vacancy details, 0 otherwise).

However, in case such interaction data are missing, also the sequence of previous job occupations can be used to fill the rating matrix [13]. The latter case does require jobs to be categorized, such that they can serve as discrete items in the rating matrix. For simplicity, we will refer to the entries in the rating matrix as ratings, irrespective of the exact type of behavioral feedback (e.g., [14,15]).

Figure 1 shows the collaborate filtering JR paradigm.

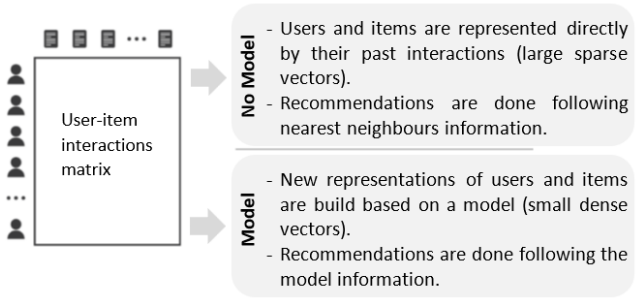


Figure 1. Collaborate filtering JR paradigm.

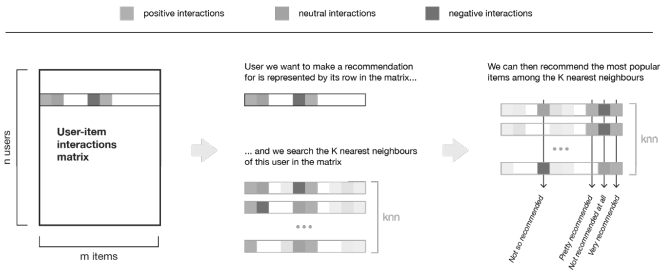
The literature commonly distinguishes between two types of CFs: memory-based and model-based CF [16,17].

In memory-based CF, recommendations are created using a K-nearest neighbor (KNN) type of approach. That is, for some user u, one tries to find either K users similar to u (user-based CF) (see Figure 2a, or K items similar to the items u has already ranked (item-based CF) (see Figure 2b). Here, the similarity is always based on the entries of the rating matrix.

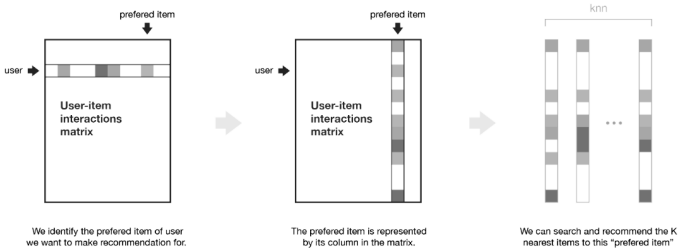
In [13,18–20], the authors used textbook CF methods. In [19,20], they applied CF methods to the RecSys 2016 dataset. In [21], they compared several auto-encoders to encode user interactions, based on which similar users are determined.

Model-based CF attempts to fill the missing values in the rating matrix using regression-based models, based solely on the rating matrix [22]. However, likely due to the large amount of textual data, which can ideally be considered as features in such regression models, we are not aware of any studies using model-based CF in job recommender systems. On the other hand, although we classified regression models using textual data as hybrid, it should be noted that these models have a strong CF flavor.

Even though textual features are used, their weights in the regression model are determined by behavioral feedback. Hence, even though not only behavioral data are used, still the resulting recommendation is largely determined by what vacancies similar job seekers have interacted with. Only the definition of *similar* has changed, as it now partly includes features based on content.



(**a**) User-based CF.



(**b**) Item-based CF.

Figure 2. Memory-based CF.

In e-recruitment, this strategy would amount to studying the preferences of a recruiter for a few candidates in order to offer them more candidates. These would be drawn from candidates retained by other recruiters, who share preferences with the *A* recruiter. The approach is difficult in our case, as the needs of recruiters are very disparate and unlikely to overlap, especially in a database of profiles of several tens of millions of candidates. Furthermore, the relevance of profiles changes over time, depending on experience and training.

CF rely more on the information given by the user (his profile for example) when it comes to offering them relevant elements [22,23]. This is the approach that we retain in our approach, insofar as we enrich the profile of the recruiter with his explicit need, expressed in the form of the job offer for which they are recruiting. In this case, it is possible for us to target relevant candidate profiles using exclusively the information provided by the recruiter without having, as in the previous strategy, to rely on candidate assessments made by other recruiters [24,25].

In most scenarios, this behavior of the recommending system is considered as it is saturated. However, in the job domain, user preference remains the same in the system [24,25]. This happens because the user rarely changes their preference to change their 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 [23]:

- Analyzing explicit property of the content;
- Building a user profile and an item profile from user-rated content.

Figure 3 shows an overview of content-based recommender system.

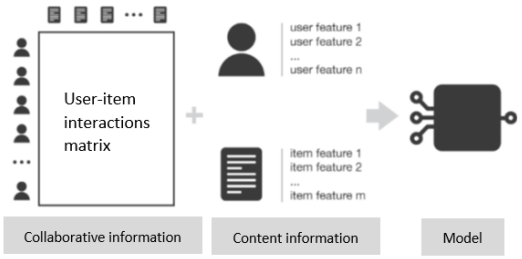


Figure 3. Content-based paradigm.

Several researches have adopted this recommendation approach in varied domains, for example, the approach presented in [26] which proposes a platform that allows building recommendation systems in IoT environment. The proposed system relies on vector representations of IoT resources using Doc2Vec neural model. Obtained results confirm the effectiveness of the approach in IoT. In the same field, the authors in [27] proposed a heuristic method to build a recommender engine in IoT environment exploiting swarm intelligence techniques.

As for hybrid recommender systems, they combine the two approaches: use of evaluations and information given by the user [28–32].

3. How to Evaluate a JRS

As for any ML algorithm, we need to be able to evaluate the performances of our JRS to decide which algorithm fits best our situation. Evaluation methods for RS can mainly be divided into two sets: an evaluation based on well-defined metrics and an evaluation mainly based on human judgment and satisfaction estimation.

It is often expensive to test algorithms on real job seekers and measure their effects. Additionally, some effects may not be measurable. The evaluation of a JRS, therefore, requires the use of intermediate measures. In the JRS literature, the evaluation of algorithms on publicly available data plays a major role. It is indeed common to calibrate and validate

the models on these data before placing the algorithms into production. This measure is also beneficial for testing and comparing different algorithms in a reproducible way.

The standard method for evaluating a system is the same as that used in machine learning, relying on a training set and a test set. Since the aim here is to predict future jobs, we artificially separate the set of jobs, revealing only the training set to adjust the parameters and we evaluate the trained systems on their ability to (re)discover the (hidden) jobs of the set of tests. If we have the opportunity to deploy the system in production, we can assess the quality of the recommendations by monitoring the effective return of job seekers. This method is commonly called A/B testing [33].

There are three families of measures to evaluate the recommendations. The most popular at the time of the Netflix challenge, in the 2000s, is based on a measure of the accuracy of the note, where we find the different scores associated with regression problems. However, these measures are less and less used because the important thing for a recommender system is rather the order—and, in particular, the order of the first recommendations—than the value of the predicted score. The most commonly used in the literature (and particularly in the field of research information) is the classification scores, called decision support measures, which quantify the number of relevant objects recommended.

Finally, finer measures, also coming from the field of information retrieval, in particular, used when recommending web pages, are rank measures. However, rank measures such as precision measures present the same problem: the recommendations judged to be false positives (FP) can correspond to true positives (TP), in the case where the note has not been observed. Thus, an excessive penalization of false positives can undervalue a recommender system.

3.1. Accuracy Measurement: Predicting the Score

Given the score matrix R, the first measure of natural reconstruction is the one established on the regression score, evaluating the distance between predicted scores and actual scores [34].

- Mean Absolute Error (MAE):
  it gives the average of the deviations between the predicted score and the actual score. It is given by the equation Equation (1).

$$MAE = \frac{1}{N}\sum_{i=1}^{N} \mid \hat{r}_{.,i} - r_{.,i} \mid \tag{1}$$

- Root Mean Square Error (RMSE), or L2 distance between predictions and actual values (used during the Netflix challenge) [35]. It is given by Equation (2).

$$RMSE = \sqrt{\frac{\sum_i (\hat{r}_{.,i} - r_{.,i})^2}{N}} \tag{2}$$

N represents the number of recommended objects. $r_{.,i}$ is the actual grade, and $\hat{r}_{.,i}$ is the predicted grade.

3.2. Decision Support Measure: Classification Score

Decision support metrics quantify recommendation success and error rates, considering that a recommender system generates a list of object recommendations for each user.

- Precision is the ratio between the number of true positives (TP) (i.e., the number of relevant recommended objects) and the number of produced recommendations (i.e., the TP plus the false positives (FP)) [36,37]. It estimates the probability that a recommended object is relevant. It is given by Equation (3).

$$Precision@k = \frac{TP}{TP + FP} \tag{3}$$

$Precision@k = |\{Recommended\ jobs\ among\ the\ first\ k, relevant\}|/|k|$.

- Recall is the ratio between the number of TP and the total number of positives, i.e., the fraction of relevant objects found in the recommendation list, i.e., the probability that a relevant object is recommended [37,38]. It is given by Equation (4).

$$Recall@k = \frac{TP}{TP + FN} \tag{4}$$

At a fixed threshold of k, the recall value makes it possible to compare two recommender systems.

- The $F1\text{-}score(F1)$ is a measure combining precision and recall [37]. It is given by Equation (5).

$$F1@k = 2 \times \frac{Precision@k \times Recall@k}{Precision@k + Recall@k} \tag{5}$$

- The ROC curve (receiver operating characteristic) visually compares the evolution of the relevance of recommendations according to the number of recommendations: on the x axis, the $FP \;\; Rate = FP/FP + TN$, in y axis, the $TP \;\; Rate = TP/TP + FP$ (i.e., the recall), starting from the point $(0, 0)$ and reaching $(1, 1)$ when all jobs have been recommended. A recommender system is all the better if it maximizes the area under the ROC curve [39]. A random system corresponds to the first bisector, and a perfect system corresponds to a curve passing through the point $(0, 1)$, thus recommending all relevant objects before irrelevant objects.

3.3. Rank Measure

As the quality of the first recommendations is more important than those of the last recommendations, in addition to considering the recall for the first values, we can use rank measures to assess the order of the first recommendations. These measures from the field of information retrieval can be adapted to the case of recommendation systems:

- Normalized Discounted Cumulative Gain (NDCG) [40] measures the quality of the order by essentially taking into account the rank assigned to the most relevant objects. This measure is essentially used in the case where there are several degrees of relevance, for example (not very relevant, somewhat relevant, very relevant. It thus measures the gain that each document according to its position in the list of results. It is given by Equation (6).

$$DCG_k = \sum_{i<k} \frac{2^{rel_i} - 1}{log(i + 1)} \tag{6}$$

where rel_i corresponds to the relevance degree of the i^{th} recommendation. NDCG is the normalization of DCG by its maximum value, thus giving a score in the range [0, 1];

$$NDCG_k = \frac{DCG_k}{maxDCG_k} \tag{7}$$

- Mean Reciprocal Rank (MRR) [41] is the inverse of the rank of the first relevant recommended job, averaged over all users. Formally, the rank of the first relevant job can be written according to Equation (8).

$$rr_u = min_{i \in \eth | R^{test}_{u,i} = 1} rang(i) \tag{8}$$

u designates a user. The set of users is noted as U. i designates a job, and the set of jobs is noted $\eth$. R is the collaborative matrix, of size $\mid U \mid \times \mid \eth \mid$.

The average overall users give:

$$MRR = \frac{1}{\mid U \mid} \sum_{u=1}^{|U|} \frac{1}{rr_u} \tag{9}$$

3.4. Other Measures

As mentioned by [42], these performance measures are not enough to fully characterize a recommender system. Other criteria are also defined:

- The *coverage* corresponds to the fraction of the data for which the recommender system can predict recommendations. For example, collaborative filtering systems do not cover the case of new users and new objects;
- *Complexity*, *calculation time*, and *response time* measure how quickly recommendations are generated. In some applications, especially for online recommendations, response time can be a critical parameter;
- *Novelty* or *Serendipity* measures how well the recommender system suggests new or unfamiliar items; this corresponds to the exploration factor of the recommendation.

4. Methodology

In this section, we will discuss our proposed representation learning model. Based on all the research methodologies and techniques reviewed in Section 2, 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 is required for the CF technique.

Instead, we have chosen to implement CBR. We 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 or Jaccard similarities. Additionally, we assigned higher weights to job skills when compared to the job domain of the user while computing similarity scores between user resumes and job offers.

Figure 4 shows the overview of the proposed content-based JRS. Three steps distinguish the proposed JRS:

1. Web scrapping: job offers are scrapped from *sa.indeed.com* (Three major cities were considered: Jeddah, Riyadh, and Dammam).
2. Data pre-processing: Tokenization and keywords extraction.
3. Matched job recommendations: similarity is used to rank recommended job offers for a job seeker's resume.

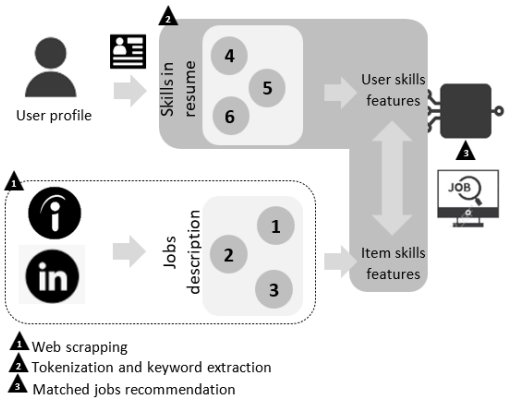


Figure 4. Overview of content-based JRS.

4.1. Data Source

For any study to be performed, the main concern is to collect data that is useful to the study. On a particular website, if the data available on the internet is in an HTML format, so the traditional way to collect the data will be complex and difficult, as it would be time-consuming.

Data in HTML language can be viewed through a web browser. Every website has its structure, so the web scrapping technique is complex to generalize for every website. Therefore, we rely on automating or creating a web crawler using the Python programming language [43].

Figure 5 shows an overview of the web scraping process.

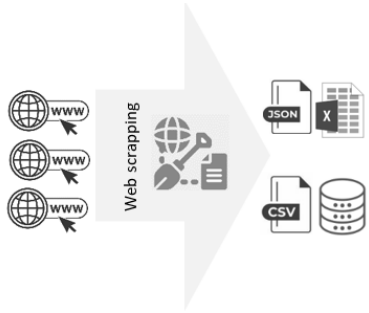


Figure 5. Web Scrapping overview.

In our study, we used *BeautifulSoup* ti scrap Web. The combination of *BeautifulSoup* and *Selenium* will accomplish the dynamic scraping [44].

We scraped *Data Scientist*, *Civil Engineer*, *Network Security Engineer*, *Sales*, and *Electrical Engineer* jobs from sa.indeed.com, accessed on 29 September 2022. We gathered jobs posted in the last 30 days (20 August–24 September 2022), in 3 major Saudi Arabia regions (Dammam, Jeddah, and Riyadh).

We used Selenium Webdriver to automate web scraping and save results in a local JSON file.

Figure 6 shows the different parts that interest us in our work from https://sa.indeed.com.

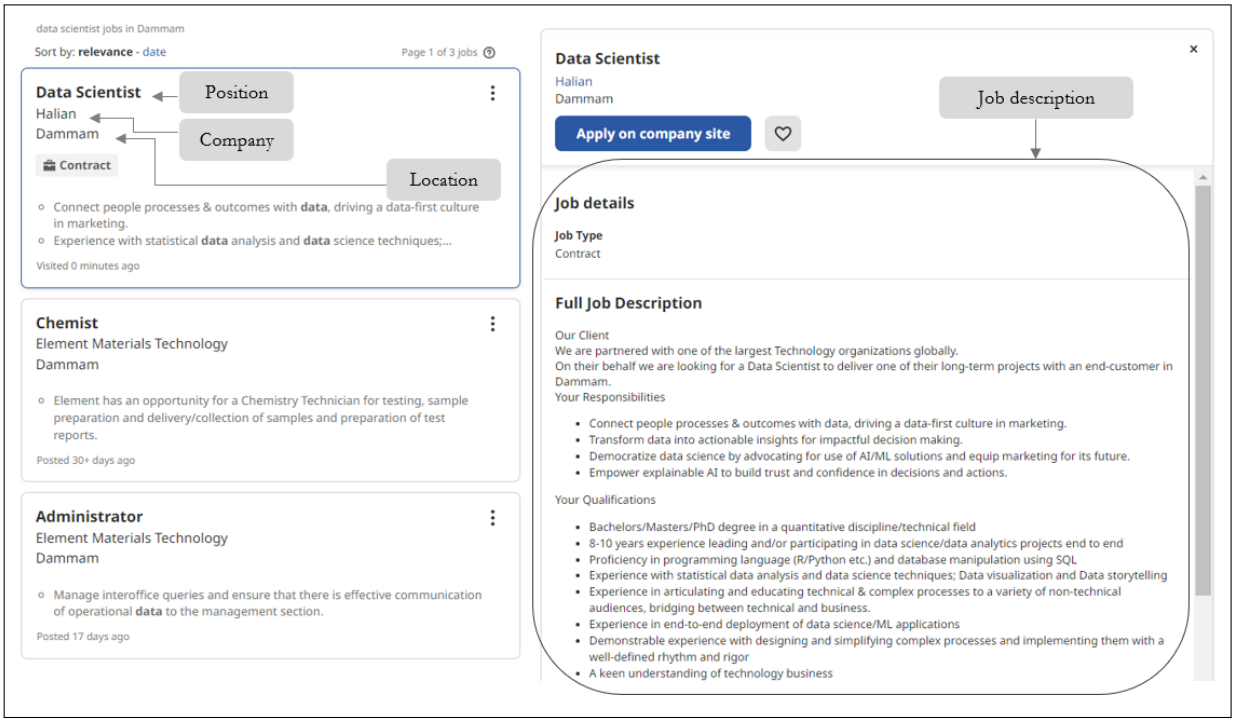


Figure 6. Data structure.

4.2. Data Pre-Processing: Tokenization and Keywords Extraction

Data pre-processing is the first process in the second tier of our study's three-tier architecture. In this study, the dataset acquired for the study has attributes filled with string columns with symbols and stop words. In particular, this is the case in the column where skills details are presented in both datasets (resumes and jobs). The usual pre-processing used methods are:

- Clean tags: remove special, unrecognized characters, and HTML tags.

- Tokenization: segmentation of data into meaningful units—terms, sometimes called *tokens*. Words are character strings separated by spaces and punctuation marks. A term is defined as a word or a series of consecutive words. More generally, n-gram designates a term formed of n consecutive words.
- Lemmatization: Normalization of terms to disregard variations in inflection (essentially related to conjugation, capital letters, and declension). We retain the masculine singular for an adjective and the infinitive for a conjugated verb.
- Stop words: filtering of terms containing no semantic information (e.g., "of", "and", "or", "but", "we", "for"); the list of these tool words is determined manually for each language. The pruning of the most frequent terms (terms present on all the texts and therefore without impact on their characterization) is also performed; the only frequency is a hyper-parameter of the approach.

4.3. Matched Jobs Recommendation

In JRS, the handled data are simply text data. A user profile describes the details of user experience and technical skills they are familiar with. On the other hand, the job listed has information such as job title, and skills required to fulfill the role. All this information is filled with text data.

In this scenario, we utilize Natural Language Processing (NLP) to measure the similarity between jobs by checking the similarity between the job title and the job description of the listed job.

Determining the text similarity is an essential task in several industrial applications such as query search, text summarizing, and video tagging [45]. In earlier studies, researchers have used different approaches to identify similarities between the text by using the edit distance algorithm which is discussed by [46], lexical overlapping technique [47] as this might work in most cases, but we cannot rely on this technique because of its frail nature [45]. In such cases, we rely on a technique called word embedding. This is a huge development in the field of distributional semantics, as it requires only a large amount of unlabelled word data. These words are represented in semantic space as a vector. That is, semantically similar words will stay close in the semantic space. To retrieve terms that are based on the similarity between two terms, we can utilize the most well known method called word2vec—a vector space model—then, we can use cosine similarity to measure the similarity between them.

This model can also be used to determine the similarity between the sentences [48]. It is a group-related model which is used to produce word embedding and these are a set of language modeling and feature learning techniques of NLP where words are mapped to real values in the vector. Typically, word2vec takes a large set of words, which is called a corpus, as input and produces vector space with dimensions being in the hundreds [49]. Once the vector space model has been generated, we can use the similarity measuring method to determine the distance or how similar the word with which we are comparing. To find similarity in vector space, we can use similarity measures such as Cosine similarity and Jaccard similarity.

- Jaccard Coefficient (JC): JC is a method to compare elements of two sets to identify which elements are shared between two sets and which are distinct. It is a similarity measure for two sets of data with results ranging from 0% to 100%. Two sets can be said to be similar when the result is close to 100%. JC is given as follows [50]:

$$JC = \frac{\#\ of\ \ elements\ \ common\ \ in\ \ two\ \ sets}{\#\ of\ \ elements\ \ in\ \ two\ \ sets} \tag{10}$$

  The above formula can be put into notation as below:

$$JC(X, Y) = \frac{|\ X \cap Y\ |}{|\ X \cup Y\ |} \tag{11}$$

- Cosine similarity (CS): CS is also a measure which finds the similarity between two sets of a non-zero vector. It is a weighted vector space model utilized in the process

of information retrieval. The similarity is measured by using the Euclidean cosine rule, i.e., by taking the inner product space of two non-zero vectors that measures the cosine of the angle between the two vectors. If the angle between two vectors is 0 deg, then the cosine of 0 is 1; this indicates that the two non-zero vectors are similar to each other. To weight the words, we used the well-known word2vec vector space model [51,52].

$$CS(A, B) = cos(\theta) = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} * \sqrt{\sum_{i=1}^{n} B_i^2}} \quad (12)$$

5. Results

In previous sections, we discussed the methodology which is fundamental to this study to implement the proposed JRS. In this section, we will discuss in more detail how the implementation and evaluation were carried out. We will also explain each step and how to conduct a JRS model.

5.1. Validating the Recommendation

The proposed JRS generates a list of job recommendations for each job seeker. To quantify recommendation success and error rates, we sought to compare the results of our system to reality. Consequently, we used the confusion matrix to highlight not only the correct and incorrect recommendations, but, also, to give us an indication of the type of errors made. Precision, recall, F1-score, and accuracy were also used to validate the recommendation.

Table 1 presents the job class codes used in the confusion matrix.

Table 1. Job domain class.

Code	Job Class
0	Data Scientist
1	Civil Engineer
2	Network Security Engineer
3	Sales
4	Electrical Engineering

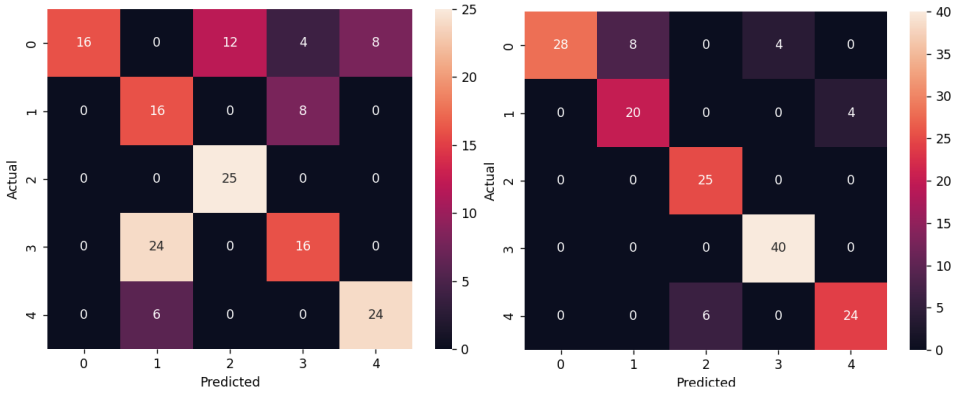
Figure 7a,b show respectively the confusion matrix using JC and CS for skills matching between job offers and job seeker resumes. We find the best recommendations on the diagonal of the confusion matrix using CS better than in that using JC. We can conclude that the use of CS for the calculation of similarity in the context of text mining to compare job descriptions and resumes is a relevant choice compared to the use of JC.

If the CS between a resume and a job offer is higher, then both the resume and job description have more skills in common. JC can be used as a dissimilarity or distance measure, whereas cosine similarity has no such constructs.

Considering the CS-based confusion matrix shown in Figure 7b, we have:

- Twenty-eight resumes classified as belonging to the Data Scientist job class out of a total of twenty-eight resumes, which is very satisfactory;
- For the Civil Engineer job class, 20 out of 28 were identified as belonging to this job class, which is quite satisfactory;
- For the Network Security Engine job class, 25 out of 31 resumes were identified, which is very satisfactory;
- For the Sales job class, 40 out of 44 resumes were classified as belonging to it, which is very satisfactory;
- Twenty-four resumes have been classified as belonging to the Electrical Engineering job class out of a total of twenty-eight, which is also quite satisfactory.

The number of TP is therefore 137 out of a total of 159 resumes, which is very satisfactory.



(**a**) JC. (**b**) CS.

Figure 7. Confusion Matrix.

We also calculated metrics necessary for the analysis of this matrix such as precision, recall, F1-score, and accuracy.

According to Table 2, using CS which gives an accuracy equal to 0.86 compared to JC (0.61), we can observe a precision of 1 for the *Data Scientist* job class. This is simply because all resumes were predicted to belong to this job class. This is in contrast to the results for the *Civil Engineer*, *Network Security Engineer*, *Sales*, and *Electrical Engineering* job classes where 20, 25, 40, and 24 were correctly assigned respectively out of 28, 31, 44, and 28 which were assigned and which gave us a precision of 0.71, 0.81, 0.91, and 0.88, respectively.

Table 2. Validating the recommendation using decision support metrics.

	Job Class	Precision	Recall	F1-Score	Support
JC	Data Scientist	1.0	0.40	0.57	40
	Civil Engineer	0.35	0.67	0.46	24
	Network Security Engineer	0.68	1.0	0.81	25
	Sales	0.57	0.40	0.47	40
	Electrical Engineering	0.75	0.80	0.77	30
	Macro AVG	0.67	0.65	0.62	159
	Weighted AVG	0.70	0.61	0.60	159
CS	Data Scientist	1.0	0.7	0.82	40
	Civil Engineer	0.71	0.83	0.77	24
	Network Security Engineer	0.81	1.0	0.89	25
	Sales	0.91	1.0	0.95	40
	Electrical Engineering	0.86	0.80	0.83	30
	Macro AVG	0.86	0.87	0.85	159
	Weighted AVG	0.88	0.86	0.86	159

Whatever the metric used, the Macro AVG presented in Table 2 performs an average following the calculation of the metric regardless of the job classes. On the contrary, the Weighted will take into account the contributions of each job class to calculate the average metric.

In our classification with several job classes, we have favored this approach since we suspect an imbalance between the job classes (number of job offers, importance, etc.)

5.2. Data Analysis

The Word Clouds for *Data Scientist* and *Network Security Engineer* shown in Figure 8a and Figure 8b, respectively, were generated from a job posting. They are graphical repre-

sentations of skills related to a specific job description from a sample of written content. They vary the size of words within the cloud based on their frequency in the source, which in this case is a job posting.

(**a**) Data Scientist. (**b**) Network Security Engineer.

Figure 8. Exploratory qualitative analysis using Word Cloud.

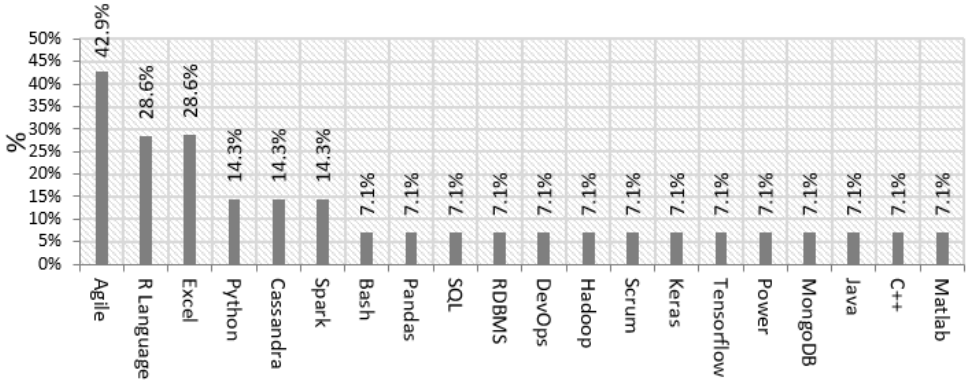
These representations are, in most cases, a reflection of the employer's internal job description for the role. In our work, when placing the job posting into a word cloud, we can quickly see what is most important to the employer. This also gives us a concrete basis for customizing the resumes to the job description to make sure a resume emphasizes the same wording.

As shown in Figure 8a,b, the obtained word clouds visualize skills and degrees as textual data. The given figures display the frequency of words and therefore, their importance. They are the best suited for exploratory qualitative analysis of skills in each job profile.

We need to know various skills related to a job description, the proposed JRS displays for a job description a ranked list of skills. Figure 9a,b show the percentage of required skills for *Data Scientist* and *Network Security Engineer*, respectively.

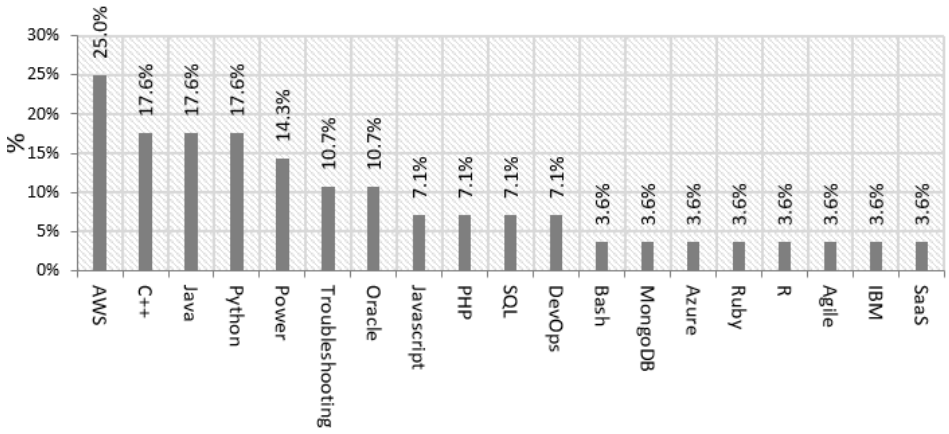
Agile (42.9%) and R programming language (28.6%) are strongly required skills for *Data Scientist* job. AWS (25%), C++ (17.6%), JAVA (17.6%), and Python (14.3%) are strongly required for *Network Security Engineer* job. Python and SQL are both required skills for *Data Scientist* (14.3% and 7.1% resp.) and *Network Security Engineer* (17.6% and 7.1%, respectively).

Troubleshooting (10.7%), SaaS (3.6%), Azure (3.6%), and Ruby (3.6%) are specific skills for *Network Security Engineer* job.



(**a**) Data Scientist.

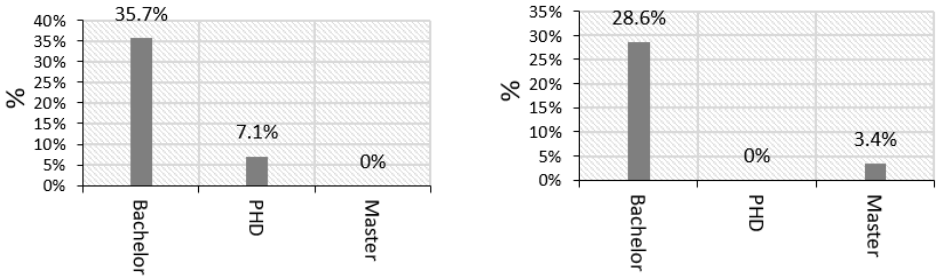
Figure 9. *Cont.*



(**b**) Network Security Engineer.

Figure 9. Percentage of required skills.

As shown in Figure 10a,b, the proposed JRS can show the percentage of required education degrees for each job.



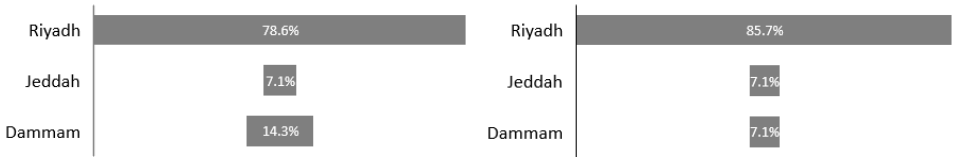
(**a**) Data Scientist. (**b**) Network Security Engineer.

Figure 10. Percentage of required education degree.

For Saudi *Data Scientist* jobs, companies strongly require a bachelor's (35.7%) more than a Ph.D. degree (7.1%). A master's degree is not required for this job. For a *Network Security Engineer* job, a bachelor's degree is strongly required (28.6%) more than a master's degree (3.4%). A Ph.D. degree is not required.

The proposed JRS can also display the percentage of each job class in major Saudi cities (see Figure 11). For example, in Riyadh, *Data Scientist* jobs are in high demand (78.6%) more than in Jeddah (7.1%) and Dammam (14.3%).

Network Security Engineer jobs are in higher demand in Riyadh (85.7%) than in Jeddah (7.1%) and Dammam (7.1%).



(**a**) Data Scientist. (**b**) Network Security Engineer.

Figure 11. Percentage of job profiles in Dammam, Jeddah, and Riyadh.

6. 3-Tier Design for RS

As shown in Figure 12, the design proposed for the study of RS for job seekers is a three-tier design. The data layer is composed of job listing data which were scrapped from http://sa.indeed.com. The processing layer is used for analysis, modeling, and recommendation.

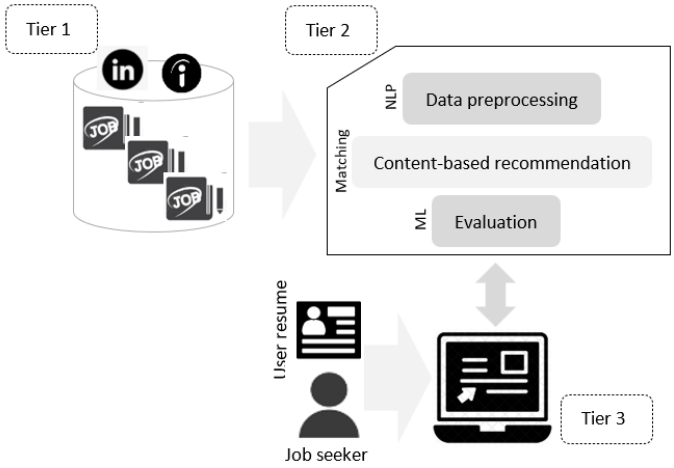


Figure 12. 3-Tier design for JRS.

A third layer is used as a presentation which consists of Python/dash web framework, which is used to recommend user top-n jobs for the skills and the user's job domain.

The first interface (see Figure 13) of the system offers the job seeker the possibility of uploading their resume, then offers a top-five recommended job.

Once the job seeker uploads their resume, the system displays at the bottom a list of recommended jobs ranked from the top satisfied matched job with their resume associated with a similarity, skills required, and a link to the full job description from *sa.indeed* (see Figure 14).

The JRS also proposes to analyze their job domain according to three criteria (see Figure 14:

- Percentage of required skills for the required job;
- Percentage of required education degree required for the required job;
- Percentage of the recommended job in major Saudi cities.

For each chart, a zoom option is provided to show in detail information related to recommended job domain (see Figure 15).

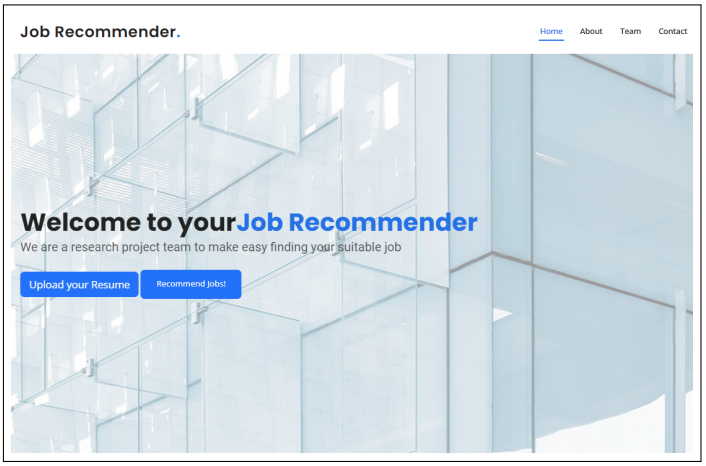
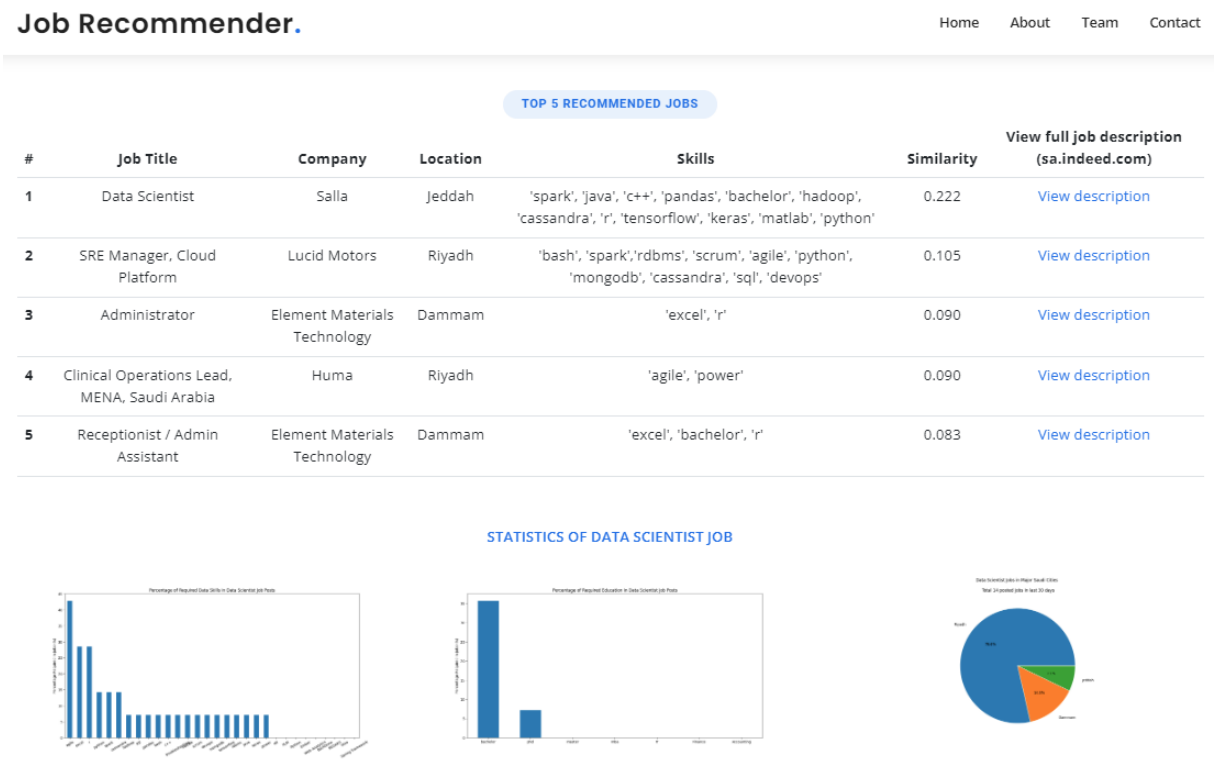


Figure 13. Main JRS interface.



#	Job Title	Company	Location	Skills	Similarity	View full job description (sa.indeed.com)
1	Data Scientist	Salla	Jeddah	'spark', 'java', 'c++', 'pandas', 'bachelor', 'hadoop', 'cassandra', 'r', 'tensorflow', 'keras', 'matlab', 'python'	0.222	View description
2	SRE Manager, Cloud Platform	Lucid Motors	Riyadh	'bash', 'spark','rdbms', 'scrum', 'agile', 'python', 'mongodb', 'cassandra', 'sql', 'devops'	0.105	View description
3	Administrator	Element Materials Technology	Dammam	'excel', 'r'	0.090	View description
4	Clinical Operations Lead, MENA, Saudi Arabia	Huma	Riyadh	'agile', 'power'	0.090	View description
5	Receptionist / Admin Assistant	Element Materials Technology	Dammam	'excel', 'bachelor', 'r'	0.083	View description

Figure 14. Top-5 recommended job for uploaded resume.

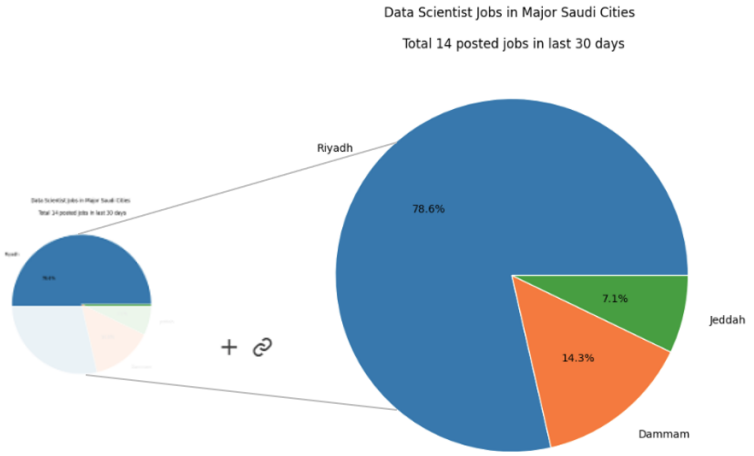


Figure 15. Zoom on charts.

7. Conclusions and Future Works

Researchers' contributions in the field of JRS still do not satisfy fully the users and do not deal with a dataset that is built from scraping online professional websites.

In this study, we aim to propose a learning system that responds to the emerging needs of any job seekers from any background, and it shows promising results. The proposed JRS is based on matching available job offers and job seekers' resumes. It aims to extract relevant data from posted jobs on the most popular websites for job seekers. As a result, a learning model is used to match skills and calculate similarity using the Jaccard coefficient and Cosine distance.

We first scanned available job descriptions from the Indeed jobs board and parsed the meaningful data to match them with the skills included in the job seekers' resumes. Then, we analyzed the skills required for the job, and to which domain the user fall; we used this as a parameter to compute the similarity between available job offers and job seekers.

Finally, we recommended a job offer that is similar to the users' skills on their resumes using CF.

Using this recommendation process allows the user to gain time and apply only to the jobs that they might be suited to, instead of applying to many jobs available in the world's largest online professional networks such as Indeed or LinkedIn.

Our recommender system helps to identify suitable job offers, based on the skills and experience in the job seekers' resumes and their relation to the required skills in the job offer.

As future work, we aim to build efficiently and evaluate the recommendation. Indeed, the classic framework of displaying recommendation results as a single sorted list is very restrictive and the highlighting of multiple criteria is desirable. Similarly, the usual evaluation indicators may be questioned, and others proposed to assess, for example, the diversity of the recommended content.

Author Contributions: Conceptualization, S.A.A., H.A.E., I.F. and R.A.; methodology, S.A.A. and H.A.E.; formal analysis, S.A.A., M.S.H., H.A.E. and I.F.; investigation, S.A.A., M.S.H., I.F. and R.A.; resources, S.A.A., H.A.E. and I.F.; data curation, S.A.A., H.A.E. and I.F.; writing—original draft preparation, S.A.A. and M.S.H.; writing—review and editing, H.A.E., I.F. and R.A.; visualization, S.A.A. and M.S.H.; supervision, M.S.H.; project administration, M.S.H.; funding acquisition, M.S.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Deanship of Scientific Research, Imam Abdulrahman Bin Faisal University, Dammam, Saudi Arabia Grant [2022-017-PYSS].

Data Availability Statement: Publicly available datasets were analyzed in this study (uploaded on 16 October 2022. This data can be found here: [https://github.com/imenFerjani/Job_Recomender_System.git].

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CBR	Content-Based Recommender systems
CF	Collaborative Filtering
CS	Cosine similarity
FP	False Positives
HRS	Hybrid Recommender Systems
IR	Information retrieval
JC	Jaccard Coefficient
JRS	Job Recommender Systems
KNN	K-Nearest Neighbor
MAE	Mean Absolute Error
ML	Machine Learning
NDCG	Normalized Discounted Cumulative Gain
NLP	Natural Language Processing
RMSE	Root Mean Square Error
ROC	Receiver Operating Characteristic
RS	Recommender Systems
TP	True Positives

References

1. Lu, Y.; El Helou, S.; Gillet, D. A Recommender System for Job Seeking and Recruiting Website. In Proceedings of the 22nd International Conference on WWW. Association for Computing Machinery, Rio de Janeiro, Brazil, 13–17 May 2013; pp. 963–966. [CrossRef]
2. Xu, Y.; Li, Z.; Gupta, A.; Bugdayci, A.; Bhasin, A. Modeling Professional Similarity by Mining Professional Career Trajectories. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 1945–1954. [CrossRef]

3. Junior, J.; Vilasbôas, F.; De Castro, L. The Influence of Feature Selection on Job Clustering for an E-recruitment Recommender System. In *International Conference on Artificial Intelligence and Soft Computing*; Springer: Cham, Switzerland, 2020; pp. 176–187. [CrossRef]
4. Mohmmadzadeh, H.; Gharehchopogh, F.S. An efficient binary chaotic symbiotic organisms search algorithm approaches for feature selection problems. *J. Supercomput.* **2021**, *77*, 9102–9144. [CrossRef]
5. Mohammadzadeh, H.; Gharehchopogh, F.S. Feature Selection with Binary Symbiotic Organisms Search Algorithm for Email Spam Detection. *Int. J. Inf. Technol. Decis. Mak.* **2021**, *20*, 469–515. [CrossRef]
6. Abdollahzadeh, B.; Gharehchopogh, F.S. A Multi-Objective Optimization Algorithm for Feature Selection Problems. *Eng. Comput.* **2022**, *38*, 1845–1863. [CrossRef]
7. Naseri, T.S.; Gharehchopogh, F.S. A Feature Selection Based on the Farmland Fertility Algorithm for Improved Intrusion Detection Systems. *J. Netw. Syst. Manag.* **2022**, *30*, 40. [CrossRef]
8. Goldanloo, M.J.; Gharehchopogh, F.S. A hybrid OBL-based firefly algorithm with symbiotic organisms search algorithm for solving continuous optimization problems. *J. Supercomput.* **2022**, *78*, 3998–4031. [CrossRef]
9. Abedi, M.; Gharehchopogh, F.S. An Improved Opposition Based Learning Firefly Algorithm with Dragonfly Algorithm for Solving Continuous Optimization Problems. *Intell. Data Anal.* **2020**, *24*, 309–338. [CrossRef]
10. Gharehchopogh, F.S.; Gholizadeh, H. A comprehensive survey: Whale Optimization Algorithm and its applications. *Swarm Evol. Comput.* **2019**, *48*, 1–24. [CrossRef]
11. Ntioudis, D.; Masa, P.; Karakostas, A.; Meditskos, G.; Vrochidis, S.; Kompatsiaris, I. Ontology-Based Personalized Job Recommendation Framework for Migrants and Refugees. *Big Data Cogn. Comput.* **2022**, *6*, 120. [CrossRef]
12. Mishra, R.; Rathi, S. Enhanced DSSM (Deep Semantic Structure Modelling) Technique for Job Recommendation. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 7790–7802. [CrossRef]
13. Lee, Y.; Lee, Y.C.; Hong, J.; Kim, S.W. Exploiting job transition patterns for effective job recommendation. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, Canada, 5–8 October 2017; pp. 2414–2419. [CrossRef]
14. Sun, N.; Chen, T.; Guo, W.; Ran, L. Enhanced Collaborative Filtering for Personalized E-Government Recommendation. *Appl. Sci.* **2021**, *11*, 12119. [CrossRef]
15. Hu, B.; Long, Z. Collaborative Filtering Recommendation Algorithm Based on User Explicit Preference. In Proceedings of the IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), Dalian, China, 28–30 June 2021; pp. 1041–1043. [CrossRef]
16. Wu, L. Collaborative Filtering Recommendation Algorithm for MOOC Resources Based on Deep Learning. *Complexity* **2021**, *2021*, 5555226. [CrossRef]
17. Liu, X.; Li, S. Collaborative Filtering Recommendation Algorithm Based on Similarity of Co-Rating Sequence. In Proceedings of the International Symposium on Electrical, Electronics and Information Engineering, Seoul, Korea, 19–21 February 2021; pp. 458–463. [CrossRef]
18. Reusens, M.; Lemahieu, W.; Baesens, B.; Sels, L. A Note on Explicit versus Implicit Information for Job Recommendation. *Decis. Support Syst.* **2017**, *98*, 26–35. [CrossRef]
19. Ahmed, S.; Hasan, M.; Hoq, M.N.; Adnan, M.A. User interaction analysis to recommend suitable jobs in career-oriented social networking sites. In Proceedings of the International Conference on Data and Software Engineering (ICoDSE), Denpasar, Indonesia, 26–27 October 2016; pp. 1–6. [CrossRef]
20. Liu, K.; Shi, X.; Kumar, A.; Zhu, L.; Natarajan, P. Temporal Learning and Sequence Modeling for a Job Recommender System. In Proceedings of the Recommender Systems Challenge; Association for Computing Machinery, New York, NY, USA, 15 September 2016; pp. 7:1–7:4. [CrossRef]
21. Lacic, E.; Reiter-Haas, M.; Kowald, D.; Reddy Dareddy, M.; Cho, J.; Lex, E. Using autoencoders for session-based job recommendations. *User Model.-User-Adapt. Interact.* **2020**, *30*, 617–658. [CrossRef]
22. Li, L.; Wang, Z.; Li, C.; Chen, L.; Wang, Y. Collaborative filtering recommendation using fusing criteria against shilling attacks. *Connect. Sci.* **2022**, *34*, 1678–1696. [CrossRef]
23. Javed, U.; Shaukat, K.; Hameed, I.A.; Iqbal, F.; Alam, T.M.; Luo, S. A Review of Content-Based and Context-Based Recommendation Systems. *Int. J. Emerg. Technol. Learn. (iJET)* **2021**, *16*, 274–306. [CrossRef]
24. Joseph, A.; Benjamin, M.J. Movie Recommendation System Using Content-Based Filtering And Cosine Similarity. In *Proceedings of the National Conference on Emerging Computer Applications (NCECA)*; Amal Jyothi College of Engineering: Kanjirapally, India, 2022; pp. 405–408. [CrossRef]
25. Pérez-Almaguer, Y.; Yera, R.; Alzahrani, A.A.; Martínez, L. Content-based group recommender systems: A general taxonomy and further improvements. *Expert Syst. Appl.* **2021**, *184*, 115444. [CrossRef]
26. Forestiero, A.P.G. Recommendation platform in Internet of Things leveraging on a self-organizing multiagent approach. *Neural Comput. Appl.* **2022**, *34*, 16049–16060. [CrossRef]
27. Forestiero, A. Heuristic recommendation technique in Internet of Things featuring swarm intelligence approach. *Expert Syst. Appl.* **2022**, *187*, 115904. [CrossRef]
28. Burke, R. Hybrid Recommender Systems: Survey and Experiments. *User Model.-User-Adapt. Interact.* **2002**, *12*, 331–370. [CrossRef]

29. Rhanoui, M.; Mikram, M.; Yousfi, S.; Kasmi, A.; Zoubeidi, N. A hybrid recommender system for patron driven library acquisition and weeding. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 2809–2819. [CrossRef]
30. Çano, E. Hybrid Recommender Systems: A Systematic Literature Review. *Intell. Data Anal.* **2017**, *21*, 1487–1524. [CrossRef]
31. Guia, M.; Silva, R.; Bernardino, J. A Hybrid Ontology-Based Recommendation System in e-Commerce. *Algorithms* **2019**, *12*, 239. [CrossRef]
32. Pande, C.; Witschel, H.F.; Martin, A. New Hybrid Techniques for Business Recommender Systems. *Appl. Sci.* **2022**, *12*, 4804. [CrossRef]
33. Kohavi, R.; Longbotham, R. Online Controlled Experiments and A/B Testing. In *Encyclopedia of Machine Learning and Data Mining*; Sammut, C., Webb, G.I., Eds.; Springer: Boston, MA, USA, 2017; pp. 922–929. [CrossRef]
34. Sammut, C.; Webb, G.I. (Eds.) Mean Absolute Error. In *Encyclopedia of Machine Learning*; Springer: Boston, MA, USA, 2010; p. 652. [CrossRef]
35. Chai, T.; Draxler, R.R. Root mean square error (RMSE) or mean absolute error (MAE)?—Arguments against avoiding RMSE in the literature. *Geosci. Model Dev.* **2014**, *7*, 1247–1250. [CrossRef]
36. Ting, K.M. Precision. In *Encyclopedia of Machine Learning*; Sammut, C., Webb, G.I., Eds.; Springer: Boston, MA, USA, 2010; p. 780. [CrossRef]
37. Powers, D.M.W. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *J. Mach. Learn. Technol.* **2011**, *2*, 37–63.
38. Ferjani, I.; Sassi Hidri, M.; Frihida, A. SiNoptiC: Swarm intelligence optimisation of convolutional neural network architectures for text classification. *Int. J. Comput. Appl. Technol.* **2002**, *68*, 82–100. [CrossRef]
39. Tan, P.N. Receiver Operating Characteristic. In *Encyclopedia of Database Systems*; Liu, L., Özsu, M.T., Eds.; Springer: Boston, MA, USA, 2009; pp. 2349–2352. [CrossRef]
40. Järvelin, K.; Kekäläinen, J. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* **2002**, *20*, 422–446. [CrossRef]
41. Voorhees, E.M.; Tice, D.M. The TREC-8 Question Answering Track. In Proceedings of the Second International Conference on Language Resources and Evaluation (LREC), Athens, Greece, 31 May–2 June 2000; pp. 77–82.
42. Herlocker, J.L.; Konstan, J.A.; Terveen, L.G.; Riedl, J.T. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* **2004**, *22*, 5–53. [CrossRef]
43. Egger, R.; Kroner, M.; Stöckl, A. Web Scraping. In *Applied Data Science in Tourism: Interdisciplinary Approaches, Methodologies, and Applications*; Egger, R., Ed.; Springer International Publishing: Berlin/Heidelberg, Germany, 2022; pp. 67–82. [CrossRef]
44. Salunke, S.S. *Selenium Webdriver in Python: Learn with Examples*, 1st ed.; CreateSpace Independent Publishing Platform: North Charleston, SC, USA, 2014.
45. Kenter, T.; de Rijke, M. Short Text Similarity with Word Embeddings. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, Melbourne, Australia, 18–23 October 2015; pp. 1411–1420. [CrossRef]
46. Mihalcea, R.; Corley, C.; Strapparava, C. Corpus-Based and Knowledge-Based Measures of Text Semantic Similarity. In Proceedings of the Twenty-First National Conference on Artificial Intelligence, Boston, MA, USA, 16–20 July 2006; pp. 775–780. [CrossRef]
47. Pakray, P.; Bandyopadhyay, S.; Gelbukh, A. Textual entailment using lexical and syntactic similarity. *Int. J. Artif. Intell. Appl.* **2011**, *2*, 43–58. [CrossRef]
48. Barzilay, R.; Elhadad, N. Sentence Alignment for Monolingual Comparable Corpora. In Proceedings of the Empirical Methods in Natural Language Processing, Sapporo, Japan, 11–12 July 2003; pp. 25–32. [CrossRef]
49. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**. arXiv:1301.3781.
50. Sternitzke, C.; Bergmann, I. Similarity measures for document mapping: A comparative study on the level of an individual scientist. *Scientometrics* **2007**, *78*, 113–130. [CrossRef]
51. Rong, X. Word2vec Parameter Learning Explained. *arXiv* **2014**. arXiv:1411.2738.
52. Herremans, D.; Chuan, C. Modeling Musical Context with Word2vec. *arXiv* **2017**. arXiv:1706.09088.