Recommender Systems for IT Recruitment

João Almeida and Luís Custódio

Abstract—Recruitment processes have increasingly become dependent on the internet. Companies post job opportunities on their websites or on online job boards and candidates search and apply online. There are inefficiencies in the process: candidates are overloaded with opportunities and companies have trouble reaching interesting candidates. The viability of applying Recommender Systems to Online IT Recruitment is the focus of this work. The IT field is a particularly good candidate for this research due to a shortage of talent and because most of IT recruitment processes already have an online component. Six different Recommender Systems were tested, using the available data: a binary rating of applications, job descriptions, and candidate profiles. The implemented models use a wide range of strategies: Collaborative Filtering, Content Based Filtering, Item-to-Item and User-to-User neighborhood models, Matrix Factorization, and a graph based approach. Based on the results of five different base models an hybrid model was built to combine their strengths and fight their limitations. The resulting system generates valuable recommendations to 60% of the tested users. While the system is not ready to be used autonomously it is useful as a supporting tool. These results pave the way for further studies in this area, showing that Recommender Systems can bring value to Online IT Recruitment.

I. INTRODUCTION

The internet has changed different aspects of our lives, job seeking is one of them. A process that used to involve physically visiting companies and searching in newspapers has been revolutionized [1].

Companies use their websites and online job boards, sites that aggregates job offers for different companies, to advertise their vacancies and candidates use the internet to search for and apply to interesting opportunities.

These processes generate information about both the users and the companies. Users upload their Curriculum Vitae (CVs) and preferences and companies post job opportunity descriptions. While interacting with the online systems the users generate data that implicitly conveys their preferences. This leads to the availability of large amounts of data that can be use to improve parts of the recruitment process [2].

Despite these tools, there are still downsides and inefficiencies to the online recruitment process. Most job boards are not able to provide personalized recommendations to the candidates, with the filtering of opportunities being mostly based on the binary presence of keywords or search terms on job descriptions. This leads to job seekers missing out on potential opportunities and companies to have lower quantity and quality applications.

This problem is more significant on the Information Technology (IT) field, where a lack of talent is preventing companies from growing and is increasing hiring costs. According to [3] this problem is the second biggest threat to IT company goals.

Candidates face varied problems when searching for job offers online:

1) Candidates have difficulties formulating what characteristics they want in a job and sometimes have expectations that do not match their skills and experience [4];
2) Candidates end up being matched to hundreds of different opportunities and have to filter and analyze all those to find the job offers they should apply to;
3) When using online job boards there is even a chance a user will not be shown the best fitting job opportunities [4], due to their queries or profiles being improperly formulated;
4) On most current systems, candidates are given generic recommendations that do not improve with the history of applications and previous actions of the users. The algorithms used are static, meaning that they do not learn from past data. For instance, if a user searches for job opportunities that require django (a python framework for web development) he/she might not be matched to a job offer on web development in python as there is no explicit match. An algorithm with a learning component may be able to understand that these two types of offers are related and that both should be suggested to the user [5].

On this work the use of Recommender Systems (RSs) as a solution to all of these problems is explored. RSs are a subset of information systems that can be applied on domains where there exists a set of users and a set of items and where these users consume/buy/rate the items. Typical applications include movie recommendations, book recommendations or product recommendations on e-commerce websites and some attempts have also been made on the field of online dating and online recruitment but with limited success [2].

In the recruitment context a recommendation given to a candidate consists of a suggestion of a job offer that matches his interests and characteristics. One given to a company is a suggestion of a candidate that matches the company’s needs and expectations.

RSs are a subset of Information systems, an area of research that focus on techniques to process and manage data. Researchers have already attempted to apply these techniques to the online recruitment field, strategies such as keyword search, text vectorization or topic models [5] [6].

However, the fact that the best fit between job and candidates depends on underlying aspects that are hard to measure [2] has deterred extensive research on the application of RSs to the area of personnel selection. The characteristics that influence choices of candidates when looking for a job have objective and subjective components.
II. RECOMMENDER SYSTEMS

RSs work by filtering or ranking the available items. These systems take advantage of data about the user preferences to model user preferences and item characteristics to determine which items would be the best matches. This data can be explicitly given by the user or extracted from its behavior. The most common form is user-items rating pairs, where a given user rates an item on a predefined scale. An implicit feedback example would be view counts of an item by a user.

The most simple form of recommender systems use only generic item ratings to create recommendations. These are not specific for each user and in most RSs there is no information about the user or its preferences.

The mean rating or difference between positive and negative votes are the most common base formulas for non personalized systems. For instance IMDb [7] uses an weighted mean of their user ratings to calculate each movie rating. Amazon [8] also uses the arithmetic mean to define the rating of each product. eBay [9] lists a seller feedback performance based on the difference between positive and negative reviews, with the goal of creating trust between buyers and sellers.

The real power of RSs arises with the use of item and user data to generate specific recommendations. This allows users to find items with characteristics tailored to their needs and enables items that are not popular (not many users are interested in them) to get recommended, increasing the diversity of consumed items.

There are many forms of classifying and characterizing RSs. This work follows the classification presented on Mining Massive Datasets [10] and The Recommender Systems Handbook [11].

Memory Based methods generate recommendations based on similarities to past data, as in Neighborhood based approaches. On these kind of models the computation is entirely done at prediction time.

Model Based systems create a model from the data that describes the user preferences and item characteristics and use that model to generate the recommendations. One example of this approach is Matrix factorization.

Content Based Filtering (CBF) methods analyze user and item characteristics. They generate the recommendations using the information they have about the user profiles and the items descriptions and ignore the user actions. For this reason the generated recommendations tend to be stable and only change on new items or new users.

Collaborative Filtering (CF) systems in contrast ignore the features of users and items, focusing only on their relations. When a user rates or reviews an item the data is called explicit, implicit data is created from the actions of the user towards the items, such as buying or consuming an item. The idea behind the process is that users with a similar history of interactions have similar preferences.

Hybrid Systems combine these different approaches to improve upon the individual performance of each method, for instance a combination of a Content-Based and a Collaborative Filtering approach may be able to overcome the short-comings of each strategy.

A. Related Work

There have been some interesting approaches to online recruitment that made use of RSs. On [2] a survey of this field is made with a very careful analysis of each approach. Rafter, et al. [12] made one the first approaches using a Neighborhood based approach and an altered version that first clustered users with single link clustering and then used the clusters of each user as the neighborhoods. In [13] the data from a recruitment website is modeled as a directed, weighted graph. On that graph, an altered version of the Page Rank algorithm [14], the 3A algorithm introduced by Helou et al. on [15], is used to determine which entities (graph nodes) are more important to other entities. On [16] instead of evaluating the fitness of a candidate for a job offer or of trying to rank the job offers based on the interest for the candidates, the authors attempted to predict the next company a user will be working at based on the previous company and its characteristics. In [17] the authors present a framework to generate job recommendations using content-based approaches, focusing on the processing of textual data.

III. METHODS

A. Data

The dataset used in this work consists of both the profiles of 35485 users and the descriptions of 1532 job offers and their relational information. Its main components are the user profiles, the job descriptions, the applications’ information between user and job offers and some extra relational information such as the companies the job offers belong too and which companies and job offers the candidates are following / bookmarking.

The user profiles contain information about the user’s career, professional goals and skills. This is the information that recruiters use to evaluate if the candidate is a good match for a job opportunity. On top of this information the recruiters have access to the user’s CV, however the users’ CVs are not available for this work.

The job descriptions contain the descriptions of the desired candidates for the position. The job requirements tend to be flexible and fit different types of candidates. For instance an opportunity might require someone who knows Python or Java, while candidates skills are rigid and do not change with the job opportunity. Consequently job descriptions tend to be more text based and, because of that, harder to process automatically.

Besides the user profiles and the job offer descriptions, the dataset contains more information that was explored. There is a list of companies, which only have a name and an internal identifier. Between these three entity types, users, job offers and companies, there are in the dataset relationships between them, some of them with attributes of their own. The relationships consist of application, follow and bookmark.

Application - When a user applies to a job offer, it can have three macro states: rejected, reviewed or hired. These states reveal if the application was considered unfit (rejected), if it was considered good enough by the recruiter, but not by the company (reviewed) or if the candidate was hired for the
job offer, after having the approval of both the recruiter and the company (hired).

**Follow** - A user can follow a company to be notified of new job opportunities from the company.

**Bookmark** - A user can bookmark a job opportunity to save it for later.

Based on this data six models were built: four neighborhood based models, an User-to-User (U2U) Content Based Filtering (CBF), an Item-to-Item (I2I) CBF, a U2U Collaborative Filtering (CF) and an I2I CF; a matrix factorization model, Simon Funk’s SVD model [13], and a graph based model, the 3A algorithm [13].

The data used for these models consists of the 14265 applications made by 2159 users to 1358 job offers, only users with three or more applications were considered. It is important to note that this restriction led to only about 6% of the users being considered. Each application is given a binary classification, 0 if it was rejected and 1 if it was reviewed and considered a good fit. Of the 14265 applications 8093 were rejected and 6172 were reviewed. No distinction was made between applications that led to a candidate being hired and that led to a reviewed application due to the small number of hires (171) in relation to the number of applications. The goal of this RS is that the recommendations would lead to the users applying and being reviewed.

The binary evaluation of the applications, either reviewed or rejected, were used as the training data for the models tested in this work. To be considered successful, the models would have to recommend matches that would pass the review process which happened on every application. This process guarantees a match between the candidate and the job offer, disregarding the subjective factors which determine if the application results in the candidate being hired.

**B. Text Processing**

Parts of the data needed to be preprocessed in order to be used to train the models. The most notorious example is the text data present in fields in the job descriptions and user profiles. These fields were processed using a combination of text processing techniques: Bag of Words (BoW) with Tokenization and Stemming, TF-IDF weighting and Latent Semantic Analysis (LSA) for dimensionality reduction.

The first step of the process is to use a BoW approach to the fields. A tokenizer function was created, this function receives the string with the text and its output is an array with one word length strings. With the tokenizer the vectorization of the fields can be done by counting the number of times each different token appears in a document. This creates for each document an array of fixed length where each entry is the count for each token.

On this processed representation of the text fields the Term Frequency - Inverse Document Frequency (TF-IDF) method is applied. The goal of TF-IDF weighting is to highlight the most informative words in the documents, words that appear in a document but are rare in the collection will have a high weight. This value is calculated as the product of the term frequency (tf) (word count or binary presence) of a word and the inverse document frequency (idf), defined as:

$$idf(t) = \log(1 + \frac{N}{n_t})$$ (1)

where $t$ is the term, $N$ is the number of documents in the corpus and $n_t$ the number of documents where term $t$ appears at least once.

As each document is represented as a vector, each entry being the count or the TF-IDF score, the corpus can be represented as a matrix where each row is a document and each column represents a term in the vocabulary.

The resulting TF-IDF weighted vector representations of the documents are of high dimensionality, which could lead to overfitting and poor performance when using this data. To counter this problem LSA (chapter 18 of [19]) was applied.

LSA (chapter 18 of [19], also called Latent Semantic Indexing (LSI), applies Singular Value Decomposition (SVD) to the term frequency matrix $X$ and selects the $k$ largest singular values to get a transformation from the original space to a lower dimensional one.

After applying these methods the transformed data is represented in a lower dimensional space where each dimension is not a word anymore, but a linear combination of words and is uninterpretable.

This transformation maintains only a part of the variance of $X$, but the amount of variance kept grows with $k$. To find the most appropriate values of $k$ for each different field an empirical technique called "elbow method" or gap statistic was used [20]. The idea behind this approach is to observe the plot of the percentage of variance maintained over $k$. This plot is monotonically increasing and usually there is an "elbow", a value of $k$ for which the variance stops increasing significantly. This "elbow" value is chosen as the best value for $k$, a value that maintains a significant part of the variance without having a too large dimensionality. The values for $k$ varied between 20 and 400 and the maintained variance from 56.29% and 79.53%.

**C. User-to-User and Item-to-Item Similarities**

For the content based methods a similarity measure between users and between job offers is needed to establish the neighborhoods. These measures were created using the expert knowledge of recruiters who helped to define the most important characteristics when considering two similar users or job offers.

The similarity between two users reflects how close their profiles are professionally, it evaluates their professional experience, skills and goals. This function was implemented as a weighted score of the similarity of each field.

The feature weights were chosen based on those that reflected more the professional profile of the users and so if two users have all fields equal they achieve a perfect similarity score.

The similarity of each field has different ways of being estimated. For fields where there are a finite number of possible values, the similarity is one if they are the same.
needed to meet the following properties:

D. User-to-User Neighborhood Models

in defining the important fields and weights. They are more similar and the help of recruiters was also valuable if the similarity is the same. A weighted score function defines which jobs are selected then the offers with the highest predicted rating. The process is described by finding in the neighbors of a user that also applied to i and doing a weighted average of the rating of those users. The process is described by equation 7

\[ r_{u,i} = \sum_{v} s_{u,v} \times r_{v,i} - \lambda \]

where \( s_{u,v} \) is the similarity between user u and user v, \( r_{v,i} \) the rating of the application of user v to job offer i and \( \lambda \) is a regularization factor to give lower scores when the similarity between u and its neighbors is low.

From equation 7 two models were built, a CBF model and a CF model. For the CBF the similarity \( s_{u,v} \) is calculated with the similarity described in section III-C. While the CF model used the Jaccard Similarity between the sets of job offers each user had applied to.

E. Item-to-Item Neighborhood Models

The I2I neighborhood model searches for similar job offers for which the user has applied to when predicting \( r_{u,j} \). Equation 7 becomes:

\[ r_{u,j} = \frac{\sum_{i} s_{i,j} \times r_{u,i}}{\sum_{i} s_{i,j}} + \lambda \]

where \( s_{i,j} \) is the similarity between job offer i and job offer j.

The remainder of the model details are equal to the U2U model, with the I2I model having also the three parameters K, \( \sigma \) and \( \lambda \).

From this equation another two models were built, again a CBF model with the similarity from section III-C and a CF model with the Jaccard Similarity.

F. Funk’s SVD Model

During the Netflix prize competition Simon Funk adapted the SVD algorithm to factorize sparse matrices and then to generate predictions from the decomposed matrices.

The general idea behind matrix factorization methods is to find the low-rank matrices P of size \( k \times n \) and Q of size \( k \times m \) that approximate the ratings matrix R of size \( n \times m \), where n is the number of users, m the number of items, k is the number of factors for each user and item and \( k \ll m \).

\[ R \approx P^T \cdot Q \]

What equation 9 means is that the rating of item \( i \) by user \( u \), \( r_{u,i} \) becomes:

\[ r_{u,i} \approx p_u^T \cdot q_i \]

where \( p_u \in \mathbb{R}^k \) is the vector with the user factors and \( q_i \in \mathbb{R}^k \) is the vector with the item factors. These factors do not necessarily represent interpretable dimensions, however, we can think of the item factors as characteristics (for instance a gender for a movie) and the user factors as the preference of the user to those characteristics (whether a user likes a certain movie gender).

On the authors define a baseline predictor \( b_{u,i} \) to evaluate matrix factorization methods, which is based on the average rating \( \mu \), the user bias \( b_u \) and item bias \( b_i \) defined as the average deviation from the mean of the user ratings and item ratings.

\[ b_{u,i} = \mu + b_i + b_u \]
Incorporating this bias into equation [10] we get:

\[ r_{u,i} \approx \mu + b_i + b_u + p_u^T q_i \]  

(12)

The model can then be trained with Stochastic Gradient Descent (SGD) applied to each of the known ratings. For a given training case \( r_{u,i} \), we update the parameters by following the opposite direction of the gradient of the error \( e_{u,i} = r_{u,i} - \hat{r}_{u,i} \), where \( \hat{r}_{u,i} \) is the estimated value of \( r_{u,i} \), yielding:

\[ b_u(k + 1) = b_u(k) + \gamma * (e_{u,i}(k) - \lambda * b_u(k)) \]  

(13)

\[ b_i(k + 1) = b_i(k) + \gamma * (e_{u,i}(k) - \lambda * b_i(k)) \]  

(14)

\[ q_i(k + 1) = q_i(k) + \gamma * (e_{u,i}(k) * q_i(k) - \lambda * q_i(k)) \]  

(15)

\[ p_u(k + 1) = p_u(k) + \gamma * (e_{u,i}(k) * p_u(k) - \lambda * p_u(k)) \]  

(16)

where \( \lambda \) is the regularization term, \( \gamma \) is the learning rate that controls the speed of SGD and \( k \) denotes the \( k \)th iteration of the SGD algorithm.

The computational cost of one iteration of SGD grows with the number of ratings, which means this approach does not scale well when the ratings matrix \( R \) is not sparse.

This RS model outputs a continuous prediction in the interval \([0, 1]\) which is then used to rank the items for recommendation. The system recommends the items with the highest predicted rating.

G. 3A Algorithm

The 3A Algorithm [13] was implemented to add variety to the studied strategies and as the dataset described in [13] is similar to the one that was available for this work.

One clear advantage of this model is that it is not only dependent on the application data, but also uses the content of the profiles and the other relationships, so it can generate recommendations for users or job opportunities that do not have applications yet. This leads to much more data being available for this model, for instance instead of just using 2159 users as in the other models, all 35485 were used.

The data is modeled as a directed weighted graph, where nodes represent: users, job offers and companies and the edges represent different relationships between nodes. These can also be either bidirectional or unidirectional, which means that an entity can influence another without being influenced back. The relationships are represented by weighted directional edges on the graph and the different types are:

**Similar** - Two entities, two users or items, are considered similar based on their content data. These bidirectional relationships were determined by the metric presented in section III-C with only the 0.1% more similar entities having the connection.

**Match** - A match relationship was created to reveal users that fit job requirements using a function with the same logic as the similarity functions presented in section III-C. The goal is not that this function is a RS by itself, but that it supports more complex RSs by providing an indication of whether the user meets the requirements of the job opportunities. This bidirectional relationship only exists between users and jobs and only the pairs that have the top 0.5% similarity score have the edge on the graph.

**Follow / Bookmark** - Users can specify companies they are interested in and receive notifications of all their new job offers. Users can also bookmark interesting job offers to consider them again in the future. This information is transcribed to the graph as a single unidirectional relationship from users to companies and to job offers.

**Application** - The bidirectional application relationship reflects only reviewed applications done by users to job opportunities, applications that were considered to match the job offer requirements. This justifies the bidirectional property of this relationship as it reflects also the job offers requirements.

**Hire** - This bidirectional relationship describes a successful application, when the user is hired for the job. This is the most important relationship as it reflects the best possible outcome for an application.

**Own** - Each job offer is owned by one company, this is reflected as a bidirectional relationship between job offers and companies. This helps to relate different job offers from the same company.

Each of the relationships has a transition matrix \( T^e \) of shape \( N \times N \), where \( N \) is the number of nodes and \( e \) is the relationship. The entries of the matrix follow:

\[ T^e_{i,j} = \begin{cases} \frac{1}{\text{outdegree}(j)}, & \text{if } i \text{ points to } j \\ 0, & \text{otherwise} \end{cases} \]  

(17)

where the \( i \) and \( j \) are nodes and the \( \text{outdegree}(j) \) is the number of nodes that \( j \) points to.

Accompanying the transition matrix the dangling node matrix \( D^e \) also of shape \( N \times N \) is created. This matrix guarantees the correct behavior of the 3A algorithm by complementing the transition matrix while making sure that the random walker can leave nodes that have no neighbors. Each entry follows:

\[ D^e_{i,j} = \begin{cases} \frac{1}{N}, & \text{if } \text{outdegree}(j) = 0 \\ 0, & \text{otherwise} \end{cases} \]  

(18)

With these two matrices the 3A algorithm creates the matrix \( M \) that represents the random walk starting at node \( i \):

\[ M = \frac{\lambda}{N} I + d \sum_{e \in E} w_e (T^e + D^e) + p_u U \]  

(19)

where \( \lambda \) is the random jump factor, \( I \) is a \( N \times N \) all ones matrix, \( E \) is the set of all relationships, \( d \) is the damping factor, \( w_e \) is the weight of relationship \( e \), \( p_u \) is the personalization factor and \( U \) is a all zeros matrix except on row \( i \). The parameters are also subject to:

\[ \sum w_e = 1 \]  

(20)

\[ \lambda, d, p_u > 0 \]  

(21)

\[ \lambda + d + p_u = 1 \]  

(22)

Each entry \( M_{i,j} \) represents the probability of jumping from node \( i \) to node \( j \) during the random walk.

To calculate the importance of each entity for a given node \( i \), the iterative power method is applied following equation [23]

\[ R^{k+1} = MR^k \]  

(23)
where $R$ is the importance vector for $i$.

From $R$ the recommendations are made by selecting the job offers with the highest importance for the user $i$.

H. Evaluation

The ideal evaluation environment for these models would be a real world scenario with users receiving recommendations and applying to job offers. By measuring the interactions of users with the models’ recommendations we would have an evaluation metric. As this is not possible, another good option would be to serve recommendations to random users and then have recruiter experts to evaluate them, simulating the users behavior. However, this is too expensive and cannot be done in large numbers. So a preliminary automated evaluation was designed to select the best parameters for the models and to estimate their performance. Then, with the models tuned, a manual evaluation was done only for a small number of random users to have a more reliable evaluation.

The Neighborhood models and the Funk’s SVD model use the applications data to generate recommendations. As the goal of these systems is to generate recommendations that will turn into reviewed applications this problem can be framed as the prediction of which pairs of user and job offer will result in a reviewed application.

As this automated result evaluation requires significant computer resources only a 5-fold cross validation was made instead of the usual 10-fold or more complex approaches.

The measure chosen to evaluate the results was the Root Mean Square Error (RMSE), as some models had continuous outputs and others discrete, despite the possible values for each recommendation being only 0 or 1.

For the Manual Evaluation, 15 users with at least 3 applications (to maintain the same environment where the models were trained) were randomly selected. Each of the models generated 3 job offer recommendations for each user. For the Funk SVD and the 3A models these were the 3 job offers with highest predicted rating. For the Neighborhood models these were selected randomly from the set of job offers that the model predicted would fit the user. These recommendations were then evaluated by recruiters, with indications to evaluate the fitness between the user and the job offer as if the user had applied to the job position. To average out some of the subjectiveness of this process all recommendations were evaluated by 3 different recruiters.

As for some users the Neighborhood models do not predict any job offer as a good fit for the user, the amount of recommendations they produce in this experiment might be lower than the 3 offers for each of the 15 users that would be expected.

The measure used to evaluate the models is the Precision @ 3 (p@3), the precision of the model when three recommendations were made. Each recommendation pair is evaluated by 3 recruiters who either classified it as a good or as a poor recommendation. The resulting scores is one of 0, 1/3, 2/3, 1 depending on how many recruiters classified the recommendation as good. The precision of a model being calculated as the average of these scores for all recommendations of that model.

IV. Results

A. Parameter Optimization

The parameters of the models were optimized based on the RMSE of the automated evaluation. The parameter space is infinite and even if discretized into a finite space it is not feasible to try every parameter combination. As such, a simple Genetic Algorithm (GA) was implemented to speed up and automate the search, following the ideas on An Introduction to Genetic Algorithms [25]. This does not guarantee that the best parameter combination is found, but it allows to find a good one in a reasonable amount of time. No other optimization technique was tested as the goal was not to evaluate the optimization techniques, but to find a good set of parameters for each model. GAs were chosen for their flexibility and easiness of implementation.

The four implementations of neighborhood based models have three parameters that control their operation.

1) The number of neighbors ($N$) considered for each prediction;
2) The regularization weight ($\lambda$) that prevents two much importance being given when there are few similar neighbors;
3) The similarity threshold ($\delta$), which determines which items or users are considered similar; The selected parameters as well as the RMSE of the neighborhood models are presented on table I.

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSE</th>
<th>$N$</th>
<th>$\lambda$</th>
<th>$\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>U2U CF</td>
<td>0.4935</td>
<td>70</td>
<td>0.15</td>
<td>0.05</td>
</tr>
<tr>
<td>I2I CF</td>
<td>0.4780</td>
<td>85</td>
<td>0.1</td>
<td>0.01</td>
</tr>
<tr>
<td>U2U CBF</td>
<td>0.5064</td>
<td>100</td>
<td>0.5</td>
<td>0.05</td>
</tr>
<tr>
<td>I2I CBF</td>
<td>0.4879</td>
<td>100</td>
<td>0.5</td>
<td>0.3</td>
</tr>
</tbody>
</table>

The two CBF models gained more value by considering more neighbors (100 and 100 vs 70 and 85). This suggests that the CBF models can find more relevant neighbors than the CF approach. Which is expected as the dataset contains a much smaller amount of data on applications (the information used by CF) than it has on user profiles and job descriptions (used by CBF methods). The regularization weight values are a consequence of the number of neighbors, a low number means the optimal value will be lower as with the CF models in comparison with CBF models. The similarity threshold is a parameter that depends on the similarity data, which is different for each of the 4 models, so there is no clear relation between the values of each model.

Simon Funk’s SVD model is controlled by four parameters:

1) The number of hidden factors ($K$) that represent each user and item;
2) The regularization weight ($\lambda_1$) used in the SGD updates;
3) The regularization weight of the job bias ($\lambda_2$);
4) The regularization weight of the user bias ($\lambda_3$); The resulting RMSE score and chosen parameters are presented on table II.
TABLE II
RMSE SCORES AFTER PARAMETER OPTIMIZATION

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSE</th>
<th>K</th>
<th>λ₁</th>
<th>λ₂</th>
<th>λ₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>Funk’s SVD Model</td>
<td>0.409</td>
<td>30</td>
<td>1</td>
<td>0.1</td>
<td>750</td>
</tr>
</tbody>
</table>

The number of factors, K, controls how much information the model should extract and store for each user and item. Not having enough would lead to an ineffective model and having too many would lead to overfitting the training data. λ₁ is the parameter that controls the operation of SGD and its optimal value depends on the dataset. The very high value for the job bias regularization (λ₂) means that in practice the user bias is ignored by the model. The user bias regularization (λ₃) in contrast has a very small value leading to a great importance when making predictions.

As the 3A Algorithm does not predict a rating, but only ranks the available job offers and because the graph is built with the application data, it was not evaluated through cross validation. Consequently the parameters for the 3A Model were not chosen in an automated way, but with some experimentation, following the original papers, and with domain knowledge.

There are two groups of parameters in the 3A model, the first group controls the random walk and the other the importance of each edge type. The three parameters that define how the random walker travels through the graph are:

1) The random jump factor (λ) which is the probability of jumping to a random node;
2) The damping factor (d) which is the probability of following the edges of the graph during the walk;
3) The personalization factor (pᵤ) which is the probability of the random walker jumping to the original user u;

These parameters are restricted according to equations 21 and 22. For λ the value chosen was 0.01 to make sure the model did not take into account too many nodes that were not connected to the user u. To guarantee a large importance given to the past interaction data of u a large pᵤ (0.69) was selected, even larger than in [13]. To respect equations 21 and 22 d was set to 0.3 .

The second set of parameters are the importance weights given to each of the different possible relationships between two nodes. The chosen weights are presented in table III these were then normalized to sum to 1, to respect equation 20.

TABLE III
RELATIONSHIPS ON THE 3A DATA GRAPH

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Similar</td>
<td>3</td>
</tr>
<tr>
<td>Match</td>
<td>3</td>
</tr>
<tr>
<td>Follow / Bookmark</td>
<td>2</td>
</tr>
<tr>
<td>Application</td>
<td>5</td>
</tr>
<tr>
<td>Hire</td>
<td>3</td>
</tr>
<tr>
<td>Own</td>
<td>3</td>
</tr>
</tbody>
</table>

The ratio between the weights is what determines their relative importance. The values were chosen to reflect the significance that each information type has in terms of predicting which users, job offers and companies match. The highest weight was naturally given to the hire and the reviewed application relationship as these demonstrate a matching between users and job offers. The follow / bookmark relationships was given a small importance as it demonstrates some interest from the user, but no real match. The similar, match and own relationships were given an average importance.

B. Automated Evaluation Results

The results presented on tables [I] and [II] are situated between 0.40 and 0.51 RMSE. These are high values for the RMSE, for comparison the maximum possible for these experiments would be 1, in the case the model predicted exactly 0 when the real value was 1 and 1 when it was 0.

This performance is justified by the small size of the dataset, which is a fraction of what is used in similar research. The restrictions imposed on the dataset, of each user and job offer having 3 recommendations, are also significantly lower than what is common.

Although the RMSE is useful to compare different models on the same dataset, comparisons with works on different datasets are not straightforward due to the difference in scales. It is then reasonable to rescale the scores. One possible way is to use the Normalized Root Mean Square Error (NRMSE) defined as:

\[ \text{NRMSE} = \frac{\text{RMSE}}{y_{\text{max}} - y_{\text{min}}} \] (24)

With the dataset used on this work the NRMSE is equal to the RMSE as the scale is already between 0 and 1. However the dataset used in the Netflix challenge is not normalized as the rating scale varies between 1 and 5. The RMSE of the original Netflix model was of 0.9514 which corresponds to a NRMSE of 0.23785 and the competition was won with a RMSE of 0.8567 corresponding to 0.214175 [23]. This clearly shows that the performance obtained on the this dataset is much worse than that of the models of the Netflix prize where the NRMSE is approximately half.

There are various factors that contribute for this discrepancy between the results on these two datasets. The size of the dataset and the amount of ratings each user and item have is probably the most important reason. The Netflix prize dataset consists of more than 100 million ratings of 18000 movies by almost 500000 users [23], by contrast this dataset has about 0.02% of those ratings. There is also the density of each dataset, Netflix’s is about 1.1 % dense while this one is about 0.038 %.

It is noticeable that the four Neighborhood based models have a similar performance and that Funk’s SVD Model has a significantly lower RMSE. Funk’s SVD outperforms clearly the other models and I2I and CF have a slight advantage over U2U and CBF strategies.

The advantage of Funk’s SVD model can be easily understood by its ability to take into account the entire dataset when making a prediction in contrast with Neighborhood models, which just consider a fixed number of similar items or users.

The reason behind the higher performance of I2I and CF models is not so clear and the difference is also not so accentuated. CF models use past data to define which users
and items are similar, this is a more flexible measure than the similarity metric implemented on CBF strategies. It learns the similarity based on the user actions and is able to understand characteristics that might not be reflected on the user profiles or on the job descriptions. I2I models focus on comparing similar job offers, this might be advantageous because each job has on average more applications, which will allow more neighbors to be found and from which the predictions can be based on.

C. Influence of the Dataset characteristics

When creating the dataset for training the models it was imposed that all users and all job offers should have at least 3 applications. To investigate the influence of these conditions an experiment was created. Taking the models with the same parameters, the dataset restrictions were changed and the RMSE measured. The limit was changed from a minimum of 3 applications per user and job offer up to 15 applications. Consequently the dataset density changed significantly, from 0.04% up to 7.67%. The density of the dataset, which is a matrix, is calculated as:

\[
\text{density} = \frac{\text{number of ratings}}{\text{number of users} \times \text{number of items}}
\]

This measure gives an indication of how much of the dataset is known, the empty entries represent a possible pair of user and item which has no rating.

A sharp decrease in the number of users, job offers and applications was clear with the increase of the restrictions. Going from 14265 applications when requiring 3 applications per user and job offer to 2179 when requiring 10 per user and 5 per job offer.

All models improved their RMSE significantly on datasets with more restrictive requirements. With the largest improvements ranging from about 0.06 up to 0.09 in RMSE. An impressive detail is that the models show performance improvements even with less data to be trained on.

One interesting thing to note regarding the RMSE scores is that the order between the scores of the different models was the same for almost all datasets. Which demonstrates that, although there is a change in the performance of the models, the conditions do not invalidate the results, at least in the range of the dataset restrictions tested.

There is also a correlation between the density of the dataset and the model’s performance, however it is not monotonous as the influence of the data restrictions. Performance tends to improve with dense datasets, but not always as it can be seen by the fact that the best performance is not achieved on the most dense dataset.

By analyzing the results we can determined that the number of applications per user is the requirement that has the largest influence on performance. This is due to the characteristics of the dataset, on average a job offer has more applications than a user. For instance while there are about 40% job offers with more than 10 users applying, there are less than 1% of users with more than 10 applications. Raising the required minimum number of applications for a user leads to a significant increase in the average number of applications per user.

D. Manual Evaluation

During the manual evaluation, besides the p@3, two new measures were analyzed: The percentage of satisfied users (SE) is the ratio between the number of users that received at least one good recommendation and all that were selected to receive recommendations. The percentage of satisfied served users (SSE) is the ratio between the number of users that received at least one good recommendation and the users that received recommendations, essentially we exclude the users for which the models were unable to generate recommendations. The results of this evaluation are compiled in table IV.

<table>
<thead>
<tr>
<th>Model</th>
<th>p@3</th>
<th># Evaluations</th>
<th>SE</th>
<th>SSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>U2U-CF</td>
<td>21.93%</td>
<td>38</td>
<td>46.7%</td>
<td>50.0%</td>
</tr>
<tr>
<td>I2I-CF</td>
<td>39.22%</td>
<td>17</td>
<td>33.3%</td>
<td>71.4%</td>
</tr>
<tr>
<td>U2U-CBF</td>
<td>22.73%</td>
<td>44</td>
<td>40.0%</td>
<td>40.0%</td>
</tr>
<tr>
<td>I2I-CBF</td>
<td>33.33%</td>
<td>12</td>
<td>26.7%</td>
<td>80.0%</td>
</tr>
<tr>
<td>Funk’s SVD</td>
<td>12.59%</td>
<td>45</td>
<td>26.7%</td>
<td>26.7%</td>
</tr>
<tr>
<td>3A Model</td>
<td>24.44%</td>
<td>45</td>
<td>46.7%</td>
<td>46.7%</td>
</tr>
</tbody>
</table>

The low performance of Funk’s SVD model is the most interesting result, as it goes from best performing model in the previous experiment to worst performing. The model recommended always the same 3 job offers for every user regardless of the user data. The reason behind this was that the model was predicting for those 3 job offers that any candidate would be considered a good candidate. This happened due to the fact that these did not have many applications (maximum 4 applications) and that all of those had been accepted. The problem was not visible in the automated evaluation because there the model was evaluated globally on all job offers and on all users using metrics based on the global error such as the RMSE or the Mean Absolute Error (MAE).

The remaining model’s results were inline to what was expected from the automated evaluation.

V. HybridMethod

Based on the results from the automated and manual evaluation a model was created that combined the two item based neighborhood models and used the 3A as the fallback model. This model generates the recommendations for a given user with the following procedure:

1) Each of the two item based models generates its set of recommendations for the user, not only the job offers which it predicts that are good matches, but also the ones it is not sure. With each recommended job offer having an associated score that varies between 0.4 and 1;
2) From the average of the scores of the job offers of the intersection of the two sets the top 3 are selected;
3) If there are not 3 in the intersection then the remaining are chosen from the sets of each model;
4) If even then there are not enough job offers the remaining are generated by the 3A model;

The hybrid model has the same p@3 score, see table V as the I2I-CBF model, but a lower score than the highest
performing model, the I2I-CF. It might seem that the hybrid model is not more useful than any of these two base models, however its main advantage is that it is able to generate recommendations to all users, having 100% coverage. This results in a higher % of satisfied users than any of the base models.

Both the base models and the hybrid model were evaluated on users that have more than 3 applications. However, the 3A model is able to generate recommendations for users that do not have any application. This allows this model to be used in conjunction with the Hybrid model to serve all users regardless of how many applications they have. A new experiment was designed to evaluate the 3A model in this situation, serving 3 recommendations for 15 random users without any application.

In this experiment the 3A model has a lower p@3 score (17.0%), see table VI, than when recommending for users with 3 or more applications (24.44%). This is expected as it shows that the model is able to make use of the applications data to provide better recommendations. The model gives valuable recommendations to 20% of the users, which compares to 60% by the hybrid model or 46.7% by the 3A model. This is clearly a worse result, however even without application data some users receive quality recommendations.

VI. Discussion

The achieved results show that there is a benefit in using RSs for online recruitment, the final hybrid system is able to make valuable recommendations to the majority of the users it was tested on. However, the system is not perfect, being recommended a job offer does not guarantee a match or that the user will want to apply to that opportunity.

The system also does not guarantee that the best matching job offer to a candidate will be recommended, even without a RS the only way of finding such a job offer is through an exhaustive search. According to the results, the hybrid system is able to give a good recommendation to 60% of the users, which reduces significantly the time needed to search for job offers for most users.

The results also prove that there is value in combining base models to achieve better performance with higher accuracy and coverage. The Hybrid system outperformed each of the base models in percentage of satisfied users and only had a worse p@3 than the I2I-CF.

Due to the nature of online recruitment most of the available research results, cannot be directly related to the ones achieved in this work.

One of the major drawbacks of this work results is the evaluation method. The manual evaluation is subjective, depending on the different criteria each recruiter has and applies, and is based on a small sample of users. However, it gives good indications of the models and sets a clear path for future research. With an online implementation of this system a reliable evaluation could be done through the measurement of the user’s interaction with the recommendations.

VII. Conclusion

The results show that the final system generates a gain in utility for the three parties involved in online recruitment, when compared to most current systems that rely on the candidate to search for job opportunities. The candidates get suggestions of job offers, that have a high change of matching their characteristics, and that they would have to spend time searching for. The companies get their job offers shown to the candidates that matter to them, saving on sourcing costs, the act of actively searching for candidates. Online recruitment platforms and job boards gain another tool to push their job opportunities to the user and to improve his experience when using their platforms.

This work resulted in a valuable introductory study of the use of RSs to online IT recruitment. Six different models were evaluated, covering some of the most studied RS strategies, but also including a recent and distinct graph based model, the 3A algorithm. The evaluation of the models was done in two steps and measuring slightly different model characteristics to give a better picture of the strengths and weaknesses of each algorithm. In result it became clear that the I2I-CF strategy is better fitted for this situation with the given dataset.

To complete the investigation of the models, an hybrid model was created. As a proof that this was a valid intuition the resulting hybrid model outperformed the studied base models. This investigation opens the door to further research into ways of combining the base models to improve performance. Further research was also done on the capacity of the 3A model of recommending job offers to users without any application data. The model was able to serve satisfying recommendations to one fifth of the users, giving good indications on its use as a fallback model.

To support this work and to better understand its details, other techniques were developed and used. For instance the data processing techniques, the similarity metrics created and the use of GAs to optimize the models can serve as base for future work. Besides these supporting tools, an investigation was done to determine the influence of the dataset’s characteristics on the model performance. These confirmed the suspicions that the amount of applications per user and job offer has a significant influence on the performances.

REFERENCES