Fairness in Job Recommendations

Thesis by
Stavros Davidopoulos

In Partial Fulfillment of the Requirements

For the Degree of

Masters of Science
Internet of Things
Intelligent environments in next-generation networks

Information and Communication Systems Engineering
University of Aegean

Supervisor
Associate Professor Panagiotis Symeonidis

Members of the examination committee
Associate Professor Theodoros Kostoulas
Associate Professor Alexis Kaporis

Contents

List of Figures

List of Tables

Abstract

The successful job search for employment and generally the professional rehabilitation of a person is an important goal that affects his life. With the development of the internet and the emergence of business-oriented social networks, recommender systems have been employed to create of successful job offers.

This thesis consists of two parts. In the first part we try to find the most accurate and fair model that accurately recommends jobs to job seekers, in respective of individual fairness, while in the second part we try to find the fairest model that fairly recommends job seekers to jobs, in respective of group fairness.

Firstly, we do a extensive review of related work on job recommendations in terms of improving accuracy and fairness of proposed models.

We are still describe the problems we are trying to solve through toy examples and all models we have used in our experiments, which we use for comparisons to extract the best of them.

In first part, we show the results of different basic algorithms as well as the results we get in the proposed models. The best of basic models, in accuracy metrics, it was the Collaborative Deep Ranking (CDR), which we improved, creating three variations of it. The Collaborative Deep Ranking using Variational autoencoders (CVDR) variation shows the best results in the used datasets, using information about the user-job interactions and job features. In detail, the performance of the algorithms is shown through evaluation metrics and graphical representations.

Also in this part, we study the fairness of the algorithms, looking at the problem from the perspective of individual fairness, and try to quantify it and improve it using re-ranking algorithms. From the experiments we conducted, we decide to improve the re-ranking algorithm Deterministic Greedy algorithm (DetGreedy), in case we have more than one protected attributes of

job seekers. The proposed Expanded Deterministic Greedy algorithm (xGF) re-ranking algorithm has better results than DetGreedy. In detail, the performance of the algorithms is shown through evaluation metrics.

In second part, we study the fairness of the Neural Matrix Factorization (NeuMF) algorithm, looking at the problem from the perspective of group fairness and try to quantify it and improve it using the same re-ranking algorithms, using information about user-job scores of NeuMF. From the experiments we conducted, the proposed xGF re-ranking algorithm has better results than DetGreedy. In detail, the performance of the algorithms is shown through graphical representations.

Finally, we formulate the conclusions of the thesis and its future extensions.

Acknowledgments

I would like to thank my supervisor Professor Panagiotis Symeonidis for the opportunity he gave me to deal with the field of recommender systems. The encouraging and targeted comments in the conduct of this research were a catalyst for the writing of this thesis.

I would also like to thank Professor Dimitris Sacharidis for his guidance in conducting this research.

Chapter 1

Introduction

Recommendation systems have gained immense popularity in various domains, such as e-commerce, streaming services, and tourism services, due to their ability to present personalized recommendations based on individual users' interests and preferences. These systems leverage information about users, items, and their interactions to generate recommendations.

People increasingly use business-oriented social networks such as LinkedIn or XING to attract recruiters and to look for jobs. Users of such networks make an effort to create personal profiles that best describe their skills, interests, and previous work experience. Even with such carefully structured content, it remains a non-trivial task to find relevant jobs. As a consequence, the feld of job recommender systems has gained much traction in academia and the industry. The main challenge that job recommender systems tackle is to retrieve a list of jobs for a user based on his preferences or to generate a list of potential candidates for recruiters based on the job's requirements. Besides, most online job sites offer the option to browse the available jobs anonymously in order to attract users to the site.

Job recommendation systems warrant extensive research due to their influence on users' lives and companies' competitiveness [1]. A distinct challenge faced by job recommendation systems is the limited number of available job vacancies. In general recommendation systems, the inventory of recommendable items is often vast, and frequently recommending popular or highly relevant items can be beneficial. However, job vacancies typically recruit only a few candidates; thus, even if certain job openings are recommended to many users, only a few will ultimately match. Job recommendation systems, therefore, need to provide recommendations that maximize the total number of

matches while accounting for users' preferences [1].

Another important challenge of job recommender systems is to recommend jobs to candidates fairly. The data used to train recommender systems usually contain biases towards specific candidate groups (e.g specific gender or age groups). The goal of job recommender systems is to not perpetuate these biases and recommend jobs fairly to diverse groups of candidates.

1.1 Motivations

The design of a job recommendation system that is accurate is imposed by the labor market in order to have more successful recruitments and consequently successful careers.

The users of such a recommendation system first of all want to be quickly recommended a job position, which is relevant to the qualifications they possess and which will eventually lead to employment. Secondary, they will prefer to be offered a job that has the highest salary, or the best development possibilities, or is close to their home, etc. From the other hand, the recruiters want to be quickly recommended the most suitable candidates for available job positions.

Previous research focus on collaborative filtering models [1, 3], content-based models [4, 5], context-based models [12], hybrid and ensemble-based models [2, 6, 7, 8, 9, 10, 11], embedding-based models [15], reciprocal recommender models [13] and representation learning models [14], to improve accuracy and beyond-accuracy metrics for valuable job recommendation , using real datasets or techinal datasets. But, no one has attempted to use hybrid models combining pair-wise ranking and various autoenconders.

Also previous research on job recommendation models has tried to improve the fairness of the job recommender models, mitigating algorithmic bias, either by using re-ranking algorithms [20, 18], or by modifying the training process to generate a bias-free model [17], or by removing bias with pre-processing of job and user features [19], using fairness metrics on real datasets or tehnical datasets. But, as far as we know, no work has used re-ranking algorithms to improve the fairness of job recommender system for candidate groups with more than one protected attributes.

1.2 Aims and Objectives

The study's aim is to predict acccurate and fair job recommendations. To do so, we use a variety of methodologies, models, and metrics to try to increase the effectiveness, accuracy and fairness of a recommender system. Finally, we assess how well they performed.

In order to achieve the aim of the study ours objectives are as:

O1 : Examine the literature on recommender systems, paying special attention to the most recent studies of job recommendations and fair job recommendations.

O2 : Analyze and evaluate the algorithms' performance by defining a minimum attribute subset that will result in more accurate and fair recommendations.

O3 : Compare these recommender models using accuracy, beyond-accuracy and fairness evaluation metrics.

1.3 Contributions

The key contribution of ours research can be summarized as follows:

C1 : To begin, we choose four algorithms from the Cornac library, feed them with user-job interactions and evaluate their results, with accuracy, beyond-accuracy and fairness metrics.

C2 : We pick one Cornac model and create an expanded versions of the technique that can handle job features and evaluate their results, with accuracy, beyond-accuracy and fairness metrics.

C3 : We evaluate our models in context of individual fairness, using a re-ranking algorithm.

C4 : We create a re-ranking model to handle more than one protected attributes and evaluate his results, with accuracy, beyond-accuracy and fairness metrics.

C5 : We study the fairness of a recommender model in respective of group fairness, applying re-ranking algorithms and evaluate their results, with accuracy and fairness metrics.

1.4 Thesis Outline

The rest of this thesis is divided into two parts and eight chapters:

- **Chapter 2** : This chapter includes a extensive review of the related work on accurate and fair job recommendations.
- **Part 1** : This part deals with the accurate and fair recommendation of jobs to job seekers and spans chapters 4-5.
- **Chapter 3** : Thesis' conceptual framework is described, as well as an in-depth investigation of our proposed models. We also give a quick description of the difficulties that these methods were used to solve.
- **Chapter 4** : Reflects the most important aspect of our work to recommending jobs to job seekers accurately and fairly. We started by presenting the datasets and evaluation metrics. Then define the experimental evaluation process and run experiments to evaluate the quality of the models using various model settings. We also do tests to see how accurate and fair our recommended explanation suggestion tactics are.
- **Part 2** : This part deals with fairly recommending job seekers to job and spans chapters 6-7.
- **Chapter 5** : Thesis' conceptual framework is described, as well as an in-depth investigation of our suggested re-ranking algorithms. We also give a quick description of the difficulties that these methods were used to solve.
- **Chapter 6** : Reflects the most important aspect of our work to recommending fair candidates for jobs. We started by presenting the datasets setups and evaluation metrics. Then define the experimental evaluation process and run experiments to evaluate the quality of the models. We also do tests to see how fair our recommended explanation suggestion tactics are.
- **Chapter 7** : We share our research's overall conclusion and findings. We also discussed future works that could be considered as extensions to the current project.

Chapter 2

Related Work

In this chapter we present the related work in the subject of improving the accuracy and the fairness in job recommendations.

2.1 Improvement of Accuracy in Job Recommendations

In this section we present the related work in the subject of accurate job recommendations. In Figure 2.1 we see the taxonomy of proposed models in these works.

2.1.1 Collaborative Filtering Models

In this subsection we present the related work on collaborative filtering models.

Optimized Collaborative Filtering Model

A Job Recommendation Method Optimized by Position Descriptions and Resume Information. In [3] the authors improve the basic item-based collaborating filtering algorithm, with historical delivery weight calculated by position descriptions and similar user weight calculated by resume information, which they were added as two influencing factors in the preference prediction formula.

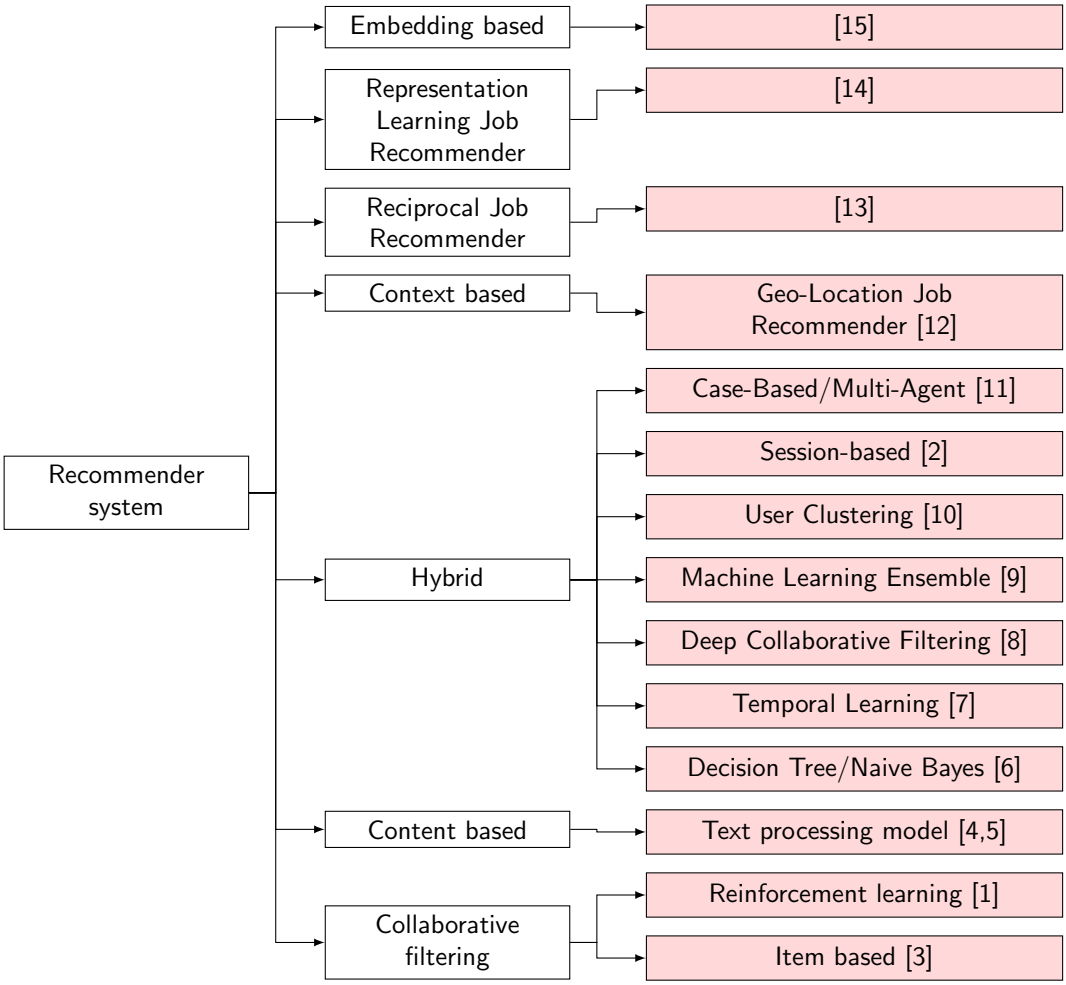

Figure 2.1: Taxonomy of proposed recommender models.

They evaluate their model using the accuracy metrics: Precision, Recall and F1-measure in real data. Their optimization method have largely improved the precision and recall rate when 1-7 positions have been recommended.

Reinforcement Learning Model

Matching Maximization in Job Recommendation System with Reinforcement Learning. In [1] the authors developing a reinforcement

learning-based job recommendation method that maximizing the number of successful matches between job seekers and jobs, which outperform from existing methods.

They used a synthetic dataset and evaluate the performance of their model in various accuracy and beyond-accuracy metrics, such as Match Efficiency Ratio, Match Success Rate, Job Fulfillment Rate, Average Applications, Recommendation Diversity, and Application Diversity.

2.1.2 Content-Based Models

In this subsection we present the related work on content-based models.

Job Recommendation based on Job Seeker Skills: An Empirical Study. In [4] the authors proposed a framework for job recommendation task. This framework facilitates the understanding of job recommendation process and it allows the use of a variety of text processing and recommendation methods according to the preferences of the job recommender system designer. Moreover, they created, used and did publicly available a new dataset containing job seekers profiles and job vacancies.

They used the Term Frequency-Inverse Document Frequency (TF-IDF) model and variants of the Word2vec model for text processing in their experiments. Their framework creates a ranking list of jobs to users according the distance between the word embeddings of user profile and job skills. They evaluated their framework using the Precision, a Minimum Effectiveness and a Score from Resource Human professionals.

Learning-Based Matched Representation System for Job Recommendation. In [5] the authors proposed a recommender system which is based on matching available job offers and job seekers' resumes. It aims to extract relevant data from posted jobs. As a result, a learning model is used to match skills and calculate similarity using the Jaccard coefficient and Cosine distance.

They first scanned and processed available job descriptions from the Indeed jobs site to match them with the skills included in the job seekers' resumes. Then, analyzed the skills required for the job, and to which domain the user fall. Finally, they recommended a job offer that is similar to the users' skills on their resumes. They evaluated the proposed model using Precision, Recall and F1-score metrics, which show the superiority of Cosine distance as a measure of similarity.

2.1.3 Hybrid and Ensemble-Based Models

In this subsection we present the related work on hybrid and ensemble-based models.

Decision Tree/Naive Bayes Hybrid Model

Machine learned Job Recommendation. In [6] the authors studied the problem of recommending jobs to people. They developed a supervised learning model that predict the next job transition of a person and recommend jobs to people.

They use a large number of job transitions obtained from the Web, and train and evaluate a machine learning model used the Weka machine-learning toolkit. They proposed the decision table/naıve Bayes hybrid classifier (DTNB), which achieves the highest accuracy, among others. A brief description of the DTNB algorithm is as follows. The set of attributes is split into two subsets by evaluating a gain function. One subset of attributes is used to build a decision table and the other to build a naıve Bayes model. Consequently, the algorithm employs a forward selection search process, where at each step, a set of selected attributes are modeled by naıve Bayes and the rest by the decision table. At each step, the algorithm considers dropping entirely a feature from the model.

They repeat their experiments for three different setups, each using a different sample from our data. In all setups, the predicted class is an institution among the most frequent 25 companies in the full data. However, in each setup, they pick the instances where the current employee position is limited to a specific set of companies or universities.

Temporal Learning Hybrid Model

Job Recommendation through Progression of Job Selection. In [7] the authors have introduced a blended approach of recommending jobs that uses job selection progression of candidates and tries to make recommendation task serendipitous. Recommendations are suggested to candidates based on the history of their job preference as well as finding jobs where candidates similar to a given candidate have applied and finding similar jobs to the jobs where a candidate has applied. Their solution overcomes the problem of candidate and job cold-start in the absence of interaction history. Their

methodology have measured an increase in click-through rates of candidates visiting their website and applying for jobs.

The proposed model combines a Bidirectional Long Short Term Memory Network with Attention (BLSTM-A), context-based methods and a method covering the edge cases where they use fuzzy matching of candidate-job features. This method is used when there are no jobs to recommend to a candidate with previous methods. The final recommendation is composed using a blended strategy that encapsulates all the methods.

Hybrid Deep Collaborative Filtering Model

Hybrid Deep Collaborative Filtering for Job Recommendation. In [8] the authors proposed a hybrid model, which combines the Collaborative Deep Learning (CDL) algorithm with a Context-Based Filtering (CBF). The use of context-based model is to overcame the problems of cold start processing, real-time training and text sparseness of job data processing of CDL model. They trained their model using real data.

They evaluate their model using the Recall metric and snow that their proposed model outperforms the CDL, the Propablistic Matrix Factorization (PMF) model and the CBF.

Machine Learning Ensemble Model

JobFit: Job Recommendation using Machine Learning and Recommendation Engine. In [9] the authors created an ensemble model, which combines different types of machine learnig algorithms for different input features. This job recommendation system also makes use of Natural Language Processing (NLP) and correlation to map user skills with the job requirements and also accounts for related skills. Their System also accounts for the applicant's personality fit for the job, the satisfaction probability and the retention probability which are not features that are considered in the other recommendation systems.

Their work is a approach to job recommendation systems using a combination of different machine learning models, NLP and Collaborative Filtering techniques. Their recommendation system (JobFit) takes into consideration various important aspects of the applicant which are relevant for recruitment. Using different datasets, they trained several machine learnimg models to assess those important aspects of the applicant thereby simulating the decision

making process of a Human Resourses Professional. They used the Random Forest classifier, the Spacy Phrase Matcher and linear regression.

User Clustering on Job Recommender Model

A Job Recommender System Based on User Clustering. In [10] the authors designed, developed and deployed an online job recommender system (iHR) for choosing the suitable recommendation approaches based on users' characteristic. They addressed the challenge caused by a single recommendation approach in a job recommender system, they group users into different clusters, using the k-means algorithm and employed different recommendation approaches (content-based filtering, collaborative filtering anf hybrid methods) for different user clusters. They evaluated their model using the the satisfaction rate of the accepted jobs in the list of Top-N recommendation.

Session-based Model

Using autoencoders for session-based job recommendations. In [2] the authors present a recommendation approach, which uses diferent autoencoder architectures to encode sessions from the job domain and use the inferred latent session representations in a k-nearest neighbor manner to recommend jobs within a session.

They evaluate the efficacy of their proposed model on three datasets: one of theirs dataset collected from the online student job portal Studo Jobs (Studo), a publicly available job dataset that was provided by XING after the RecSys Challenge 2017 (Recsys17) and a publicly available job dataset from the job platform CareerBuilder (Careerbuilder12).

Also they train and test the autoencoders on two sources of job-related data: (1) interaction data from sessions and (2) content features of job postings, for which interactions took place during a session. Their results show that variational autoencoders provide competitive job recommendations in terms of accuracy compared to the state-of-the-art session-based recommendation algorithms. The evaluation of all session-based job recommender approaches on accuracy is done with Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain (NDCG) metrics.

Finally they evaluate all session-based job recommender approaches with beyond-accuracy metrics: system-based and session-based novelty and cov-

erage and find that autoencoders can produce more novel recommendations compared to the baselines and, at the same time, provide relevant jobs for the user while maintaining a high coverage.

Case-Based/Multi-Agent Hybrid Model

Hybrid job offer recommender system in a social network. In [11] the authors presented a hybrid recommender system based on agent technology. Through the use of intelligent techniques, it is able to carry out analyses of user profiles and job offers found on the social network beBee.

Except from the agents, the hybrid system is also composed of a Content-Based Reasoning (CBR) system. It manages a series of metrics which calculate the affinity between job offers and users. Moreover, an argumentation framework is included, which expands the functionality of CBR in such way that the agents have the possibility of establishing a dialogue between them. When a query arises in the system, each agent first analyses the social network individually and each comes up with a solution. Then a dialogue is established where each agent generates arguments in order to oppose the solutions of other agents and defend their own solution as the most optimal for the problem until finally an agreement is reached. Thus, dialogue between agents is an essential feature in the system that allows to find a unique and the most optimal solution.

They evaluated the proposed system using the acceptance rate metric on job offers both for companies and for users.

2.1.4 Context-Based Model

A smart Geo-Location Job Recommender System Based on Social Media Posts. In [12] the authors proposed a context-based model which is composed into phases of data collection, data set preparation, Natural Language Processing for feature extraction, Geo-location detection, and best match recommendation.

They explored various data mining techniques to design the proposed recommendation system. Job seekers posts and employers job postings are collected from various fields of specialization and unsupervised learning methods were used to prepare the shortlisted database. The resulting database were optimized to recommend best match opportunities for both seekers and employers.

Text mining techniques are applied on posts to structure the in between lines information. NLP classifiers such as Naive Bayes, Support Vector Machine (SVM), and Random Forests, are used for extracting information from textual analysis to enrich the description within a predictive algorithm. The proposed recommendation system adapted to the cold start solution, this method proved promising results during experiments on a real data set [12].

2.1.5 Reciprocal Job Recommender Model

Applying Different Classification Techniques in Reciprocal Job Recommender System for Considering Job Candidate Preferences. In [13] the authors presented a reciprocal job recommendation system named Classification-Candidate Reciprocal Recommendation (CCRS). In order to recommend jobs to the candidates, the proposed system takes advantage of applications of similar candidates, and preference information on jobs. The created system has been tested using different classification methods. As a result, it was accepted that the proposed system has the best performance for Mean Averaged Precision (MAP) measurement with SVM classification methods. Experimental results show that the performance of the proposed system is better than other compared systems. They used a dataset received from the job seeking website Kariyer.Net.

2.1.6 Representation Learning Model

A Combined Representation Learning Approach for Better Job and Skill Recommendation. In [14] the authors proposed a representation learning based model to address the job and skill recommendation task. Their representation learning model utilizes the pairwise ranking objective which learns job and skill vector representations into a shared latent space using three pre-processed graphs. This joint embedding approach provides high quality job recommendation and also provides skill suggestions required to obtain the new job. Their experimental results on the CareerBuilder dataset and case studies demonstrate that the proposed methodology consistently outperforms several existing state-of-the-arts for the job and skill recommendation. They evaluated their proposed model, named Joint-Bayesian Personalized Ranking (Joint-BPR) and Joint-Margin, with Hit-Rate, NDCG and Area Under Curve (AUC) metrics.

A limitation of their proposed representation learning framework is that it learns representation vectors of jobs and skills that are available in the input graphs. But, in real settings, we often observe new job titles and skills, and their model is needed to be retrained to obtain representation vectors of these entities so that we can utilize them in the job and skill suggestion.

2.1.7 Embedding-based Recommender Model

Embedding-based Recommender System for Job to Candidate Matching on Scale. In [15] the authors proposed a two-stage embedding-based recommender system for job to candidates matching. The architecture of this system consists of a two-stage recommendation procedure, a fused-embedding component for candidate retrieval and a fine-tuning re-ranking module. Their two-stage job to candidate matching system has shown a significant improvement over the baseline model by measures of Click-Through-Rate (CTR) and NDCG in real world production environment.

2.2 Improvement of Fairness in Job Recommendations

In this section we present the related work in the subject of fair job recommendations.

2.2.1 Fair Job Recommendations

In this subsection we present the related work on implementation of models, which try to improve the fair spread of proposed jobs to certain focus groups of users.

Job Recommendations using Optimal Transport

ReCon: Reducing Congestion in Job Recommendation using Optimal Transport. In [17] the authors proposed a approach, named ReCon, for reducing congestion in job recommendation systems using optimal transport theory. Their approach aimed to ensure a more equal spread of jobs over job seekers by combining optimal transport and a given job recommendation model in a multi-objective optimization problem. The evaluation results showed that their approach had good performance on both congestion-related (Coverage, Gongestion, Gini Index) and desirability measures (Recall, NDCG, Hit Rate). The proposed method is a promising solution to the problem of congestion in job recommendation systems, and can potentially lead to more efficient and effective job matching.

Job Recommendations under Quantity Constraints

Fairness in Job Recommendation under Quantity Constraints. In [18] the authors proposed a framework to achieve fair resource allocation for job recommendations, both for quantity and fairness constraint, through a post-processing method. Experimental results show that their method is able to improve the recommendation fairness between protected and advantaged user groups while maintaining desirable recommendation accuracy.

They used the ranking metrics Hit Rate, F1 score, and NDCG to evaluate the top-N recommendation quality and User Group Fairness (UGF) metric to evaluate user fairness in a real-world dataset.

Adversarial Fairness in Job Recommendation

Closing the Gender Wage Gap: Adversarial Fairness in Job Recommendation. In [19] the authors focused on removing gender bias from word embeddings of job vacancy texts and user resumes with the goal of creating debiased job recommendations. They showed that gender can be predicted extremely well from anonymised resume embeddings and that naive resume-to-job recommendations based on these embeddings can perpetuate the salary difference that exists between women and men. The proposed adversarial debiasing model improved statistical parity for industry classification based on user resume and eliminated the female/male salary difference in job recommendations.

In their experiments used real data and evaluated their model using accuracy, AUC and True Positive Rate (TPR) metrics. Fairness was evaluated using statistical parity metric.

Fairness-Aware Ranking

Fairness-Aware Ranking in Search and Recommendation Systems with Application to LinkedIn Talent Search. In [20] the authors proposed a framework for fair re-ranking of results based on desired proportions over one or more protected attributes. They proposed several measures to quantify bias with respect to protected attributes such as gender and age, and presented fairness-aware ranking algorithms. They demonstrated the efficacy of these algorithms in reducing bias without affecting utility, and compared their performance via extensive simulations. They also deployed the proposed framework for achieving gender-representative ranking in LinkedIn Talent Search, where their approach resulted in huge improvement in the fairness metrics without impacting the business metrics.

We use one of these fair measures and one of their proposed re-ranking algorithms to conduct our experiments.

Part I

Individual Fairness in Job Recommendations

Chapter 3

Proposed Methodology

In this chapter we present the problem we are trying to solve through a toy example and describe all the models we have used in our experiments, starting with the description of basic models and continuing with the proposed models.

In recommender systems, the terms ***individual fairness*** refers to the fact that a list of recommended items is structured in such a way that the proportion of items pertaining to *different categories of interest (output bias)* is the same as the proportion present in the preferences of the target user (*input bias*). We also want similar users to be treated similarly. In cases of such disproportion, an algorithm should be applied to re-rank the list of recommended items and achieve *fairness*, so that representative items relevant to all categories of user's interests are included. In our Job recommendation scenario, for a target job seeker, a list of recommended jobs is fairly structured if it contains jobs that cover different job requirements that the job seeker has in his profile. For example, a job seeker has studied economics and programming. Thus, he should receive jobs equally from both industries. This means, that the job recommendation list should maintain a proportion of job requirements that is same to the proportion of the job requirements which are included in the target job seeker's profile. For example, if a job seeker had 7 previous job roles related to Programming, and 3 job roles related to Economics, a job recommendation list would be relatively fairer if it contained 70% programming jobs and 30% economics jobs.

3.1 Toy Example

To understand the problem of predicting an adequate job to a user we present the following example.

In Figure 3.1 the interactions between job seekers and jobs are shown. The mapping between job seekers and jobs is not one-to-one because a job seeker may have interacted with many jobs. As can be seen in Figure 3.1, the job seeker 4 has interacted with jobs 45 and 10.

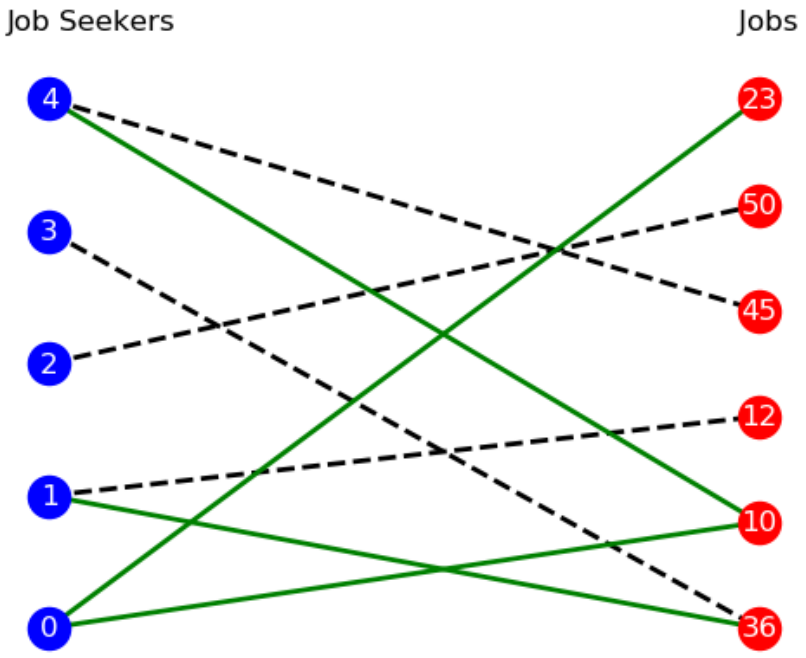

Figure 3.1: The user-jobs bipartite graph of the **Random Algorithm**. Blue nodes represent Job Seekers, whereas red nodes represent Jobs. A dashed black edge corresponds to a job recommendation that the user did not show interest and did not apply for it. The green line corresponds to a job recommendation which has followed with a job application (i.e., the user had received a job recommendation and he also applied for it).

A random algorithm, which suggests jobs to job seekers, would have the results of the Figure 3.1. For example, the random algorithm suggests to job seeker 4 the jobs 10 and 45, and the target user eventually applied for job 10

(the green line denotes that a job is both recommended to a user and the user also applied for it), which means a 50% hit ratio of the random algorithm for job seeker 4. Thus, the random algorithm succeeds in suggesting to job seeker 4 a job he prefers and recommends also a second one that job seeker 4 was not interested and he did not apply for it. In conclusion we see that the random algorithm has succeeded in predicting correct 4 out of 8 user-job interactions.

In Table 3.1 we see the jobs requirements of jobs 10, 45, 23 and 12. From this table we see that the jobs 10 and 45 have only one job requirement in common (*industry_id*), while the jobs 10, 23 and 12 have many features in common. From this we understand why the job seeker 4, applied for job 10, but he didn't apply for job 45, which was recommended by the random algorithm. Thus, the random algorithm failed to suggest a similar job to his interests that might interest him.

Table 3.1: The Job Requirements of Jobs 10,45, 23 and 12

job_id	10	45	23	12
career_level	2	3	2	2
discipline_id	20	3	20	20
industry_id	22	22	22	22
country	de	ch	de	at
region	1	0	13	0

From the above it seems that our problem is the best matching to each job seeker with the right jobs, which he prefers. An ideal algorithm would recommend to user 4, jobs 10, 23, and 12, as shown in Figure 3.2, which presents that all job recommendations are followed by job applications from the user 4.

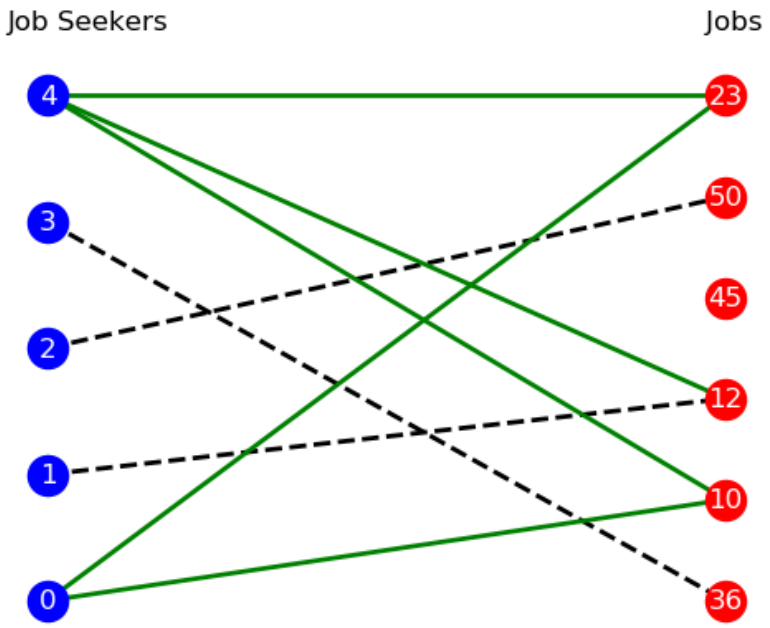

Figure 3.2: The user-jobs bipartite graph of the **Ideal Algorithm**. Blue nodes represent Job Seekers, whereas red nodes represent Jobs. A dashed black edge corresponds to a job recommendation that the user did not show interest and did not apply for it. The green line corresponds to a job recommendation which has followed with a job application (i.e., the user had received a job recommendation and he also applied for it).

In Table 3.2 we see the qualifications of user 4. From this table we see that the job 10 and 23 have three features in common with user 4, while the job 12 have all features in common. As far as the individual fairness is concerned, we understand that the ideal algorithm is biased from the respective of location, because the job seeker 4 is located in a different geographic area than the jobs 10 and 23 offered. Thus, the ideal algorithm failed to suggest two jobs geographically close to the user, wheareas only the job 12 is from the same country and location with the job seeker. This means that the input bias of the job seeker is covered only by the output bias of the recommended job 12.

Table 3.2: The Characteristics of User 4

user_id	4
career_level	2
discipline_id	20
industry_id	22
country	at
region	0

From the above it seems that another problem is the creation of unbiased recommendations for a target job seeker. An unbiased algorithm would have recommend only jobs which maintain the user proportions for particular important attributes (e.g., geographical features, job roles). After experiments we propose the use of the re-ranking algorithm **xGF**, which improves the fairness of the results of a recommedation algorithm.

3.2 Autoenconders

Autoencoders are a type of neural network, which was used as a more effective method than Principal Component Analysis (PCA) with respect to describing and reducing the dimensionality of data.

Autoencoders are trained in an unsupervised manner where the network is trying to reconstruct the input by passing the information to the output layer through a bottleneck architecture. In this thesis, we employ three variants of autoencoder architectures to represent side information about jobs : (1) a Stacked Autoencoder (SAE), (2) a Stacked Denoising Autoencoder (SDAE) and (3) a Variational Autoencoder (VAE).

3.2.1 Stacked Autoencoders

Generally, a good representation of side information about items can improve performance of a recommender system. Autoencoders (AEs) learn an compressed representation from input text to recover this input through a feed-forward neural network. SAE stacks AEs to form a deep network by feeding the output code of a AE found lower on the stack as input to a AE

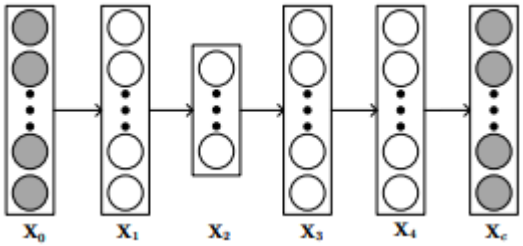

Figure 3.3: The graphic model of SDAE with $L = 4$.

found after. An SAE network is to minimize the regularized optimization problem as below:

$$\min_{\{W_l\},\{b_l\}} \|X_c - X_L\|_F^2 + \lambda_w \sum_l \left(\|W\|_F^2 + \|b\|_F^2\right), \tag{3.1}$$

where X_c is the input, W_l and b_l is the weight matrix and bias vector of layer l,L is the number of layers,and λ_w is the regularization hyperparameter. After this process, $X_{\frac{L}{2}}$ could effectively present the latent feature representation of side information about all items.

3.2.2 Stacked Denoising Autoencoders

Extending SAE by corrupting the input we costruct a SDAE . The idea of a SDAE is to learn representations that are robust to small, irrelevant changes in the input. Corrupting the input can be done on either one or multiple layers before we calculate the final output.

In this thesis we use the original corruption method of the Collaborative Deep Ranking (CDR) algorithm, where SDAE employs a mixture of edge detectors and grating filters (i.e. masking noise) with a noise level of 30% to obtain the corrupted input X_0 from the clean input X_c. In Figure 3.3 shows the graphic model of SDAE.

3.2.3 Variational Autoencoders

Another approach to extract the latent representation z_s is to use variational inference. The VAE model constructs a generative latent variable model for

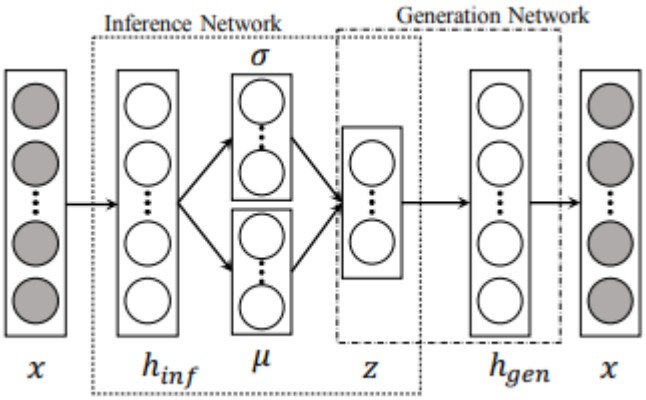

Figure 3.4: The graphic model of VAE.

the content and assigns a latent content variable z_j to each item j. The content of the item x_j is generated from its latent variable z_j through generation neural network parameterized by θ:

$$x_j \sim p_\theta \left(x_j \mid z_j \right) \tag{3.2}$$

The generation network is illustrated in Figure 3.4, where, given the latent variable z, x is generated through a Multi-Layer Perseptron (MLP).

3.3 Collaborative Deep Ranking

The CDR algorithm [16] integrates pair-wise ranking models and side information about the items on implicit feedback recomendation scenarios.

Notation and Problem Definition. Let U denote the set of users and I denote the set of items.The size of U and I are n and m, respectively and $R \in U \times I$ is the implicit interaction matrix . Specifically, the elements $r_{ij} = 1$ in matrix R denotes user i prefers item j, while $r_{ij} = 0$ implies that user i is not interesting in item j or might not observe item j yet. For a given user i, pair-wise algorithms suppose that user i prefers item j over item k if and only if $j \in I^+$ and $k \in I \setminus I^+$, where $I^+ = \{j : r_i j = 1\}$.

Except the observed binary matrix R, side information about items could be collected in many scenarios. Given an $m \times s$ matrix X_c presents the side information about all items, the j-th row denotes the bag-of-words vector of

item j based on vocabulary of size S (i.e. $X_{c,j}$).Let u_i, v_j denote the latent factor with low dimension K of user i and item j, respectively. The objective is to learn the latent factor $U = (u_i)_{i=1}^n$ and $V = (v_j)_{j=1}^m$ from implicit interaction and item information matrix for recommending an personalized ranking list for users.

Collaborative Deep Ranking. CDR exploits pair-wise preferences and content-based items feature together for collaborative filtering. Figure 3.5 shows the graphic model of CDR. There are two generative processes in model. First, the original SDAE process (in the red dashed rectangle) extracts feature representation from side information about items and then integrates them into latent factor of items in pair-wise ranking model. Second, the pair-wise ranking model captures special relationship $\delta_{ijk} = r_{ij} - r_{ik}$, which delegates the preference of user i on item j and k. Unlike the point-wise approach directly predicting the value r_ij, pair-wise approach instead classifies the difference of $r_{ij} - r_{ik}$.

The algorithm has loss function:

$$\begin{aligned}\mathcal{L} = &- \sum_{ijk} \frac{c_{ijk}}{2} \left(\delta_{ijk} - \left(u_i^T v_j - u_i^T v_k\right)\right)^2 - \frac{\lambda_w}{2} \sum_j \left(W_l^2 + b_l^2\right) \\ &- \frac{\lambda_u}{2} \sum_i u_i^T u_i - \frac{\lambda_v}{2} \sum_j \left(v_j - X_{\frac{L}{2},j*}^T\right)^2 - \frac{\lambda_n}{2} \sum_j \left(X_{L,j*} - X_{c,j*}\right)^2\end{aligned} \tag{3.3}$$

where confidence parameter c_{ijk} denotes how much user i prefers item j than item k and $\lambda_u, \lambda_v, \lambda_n$ are the regulation hyperparameters.

The first term in equation 3.3 extracts user preference from implicit matrix R to construct pair-wise ranking loss, while the fourth term integrates content of items. Therefore, if two item j and k have similar side information (i.e. similar $X_{\frac{L}{2}}$), the distance between v_j and v_k will be reduced. $X_{\frac{L}{2}}$ serves as a bridge between pair-wise ranking and SDAE model. When λ_v/λ_n goes to positive infinity, the disappeared reconstruction error will lead to invalid feature representation $X_{\frac{L}{2}}$, meanwhile $X_{\frac{L}{2}}$ dominate the learning process of V. On the other hand, when λ_v/λ_n approaches to zeros, CDR will decouple into two models and the learned V is not influenced by side information about items. Both extreme cases demonstrate bad performance in the experiments.

Prediction. After learning the optimal parameters U, V, W^+, they predict R_{ij} from its expectation:

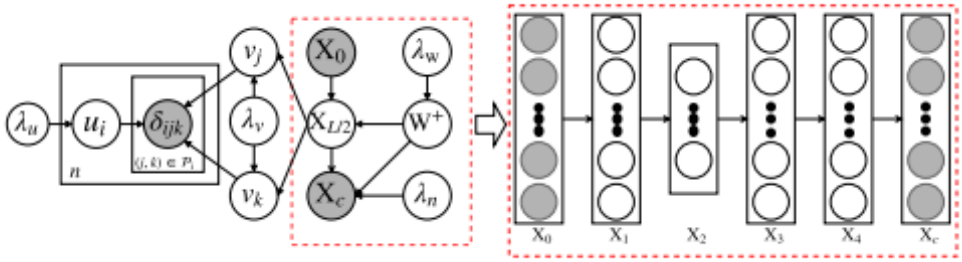

Figure 3.5: The graphic model of CDR. SDAE with $L = 4$ is presented inside the dashed rectangle. Note that W^+ denotes the set of weight matrices and bias vectors of all layers [16].

$$E\left[R_{ij} \mid U, V, W^+, \ldots\right] \approx u_i^T \left(X_{\frac{L}{2}} + \epsilon_j\right) = u_i v_j,$$

(where ϵ_j is the latent item offset) and then a ranked list of items is generated for each user based on these prediction values.

3.4 Collaborative Deep Ranking using SAE

The Collaborative Deep Ranking using stacked Autoencoder (CADR) algorithm, same as CDR, integrates pair-wise ranking models and side information about the items on implicit feedback recomendation scenarios. Differs from CDR because uses SAE for representation of side information and not SDAE. In CADR, we use the same notation and problem definition from CDR.

3.5 Collaborative Deep Ranking using VAE

The CVDR algorithm integrates pair-wise ranking models and side information about the items on implicit feedback recomendation scenarios. Differs from CDR because uses VAE for representation of side information and not SDAE.

Using the same notation and problem definition from CDR, we create the CVDR algorithm.

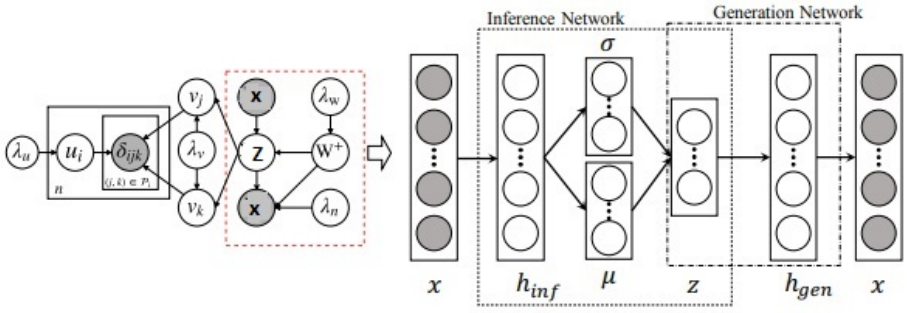

Figure 3.6: The graphic model of CVDR. VAE is presented right. Note that W^+ denotes the set of weight matrices and bias vectors of all layers.

Collaborative Deep Ranking using VAE. CVDR exploits pair-wise preferences and content-based items feature together for collaborative filtering. Figure 3.6 shows the graphic model of CVDR. There are two generative processes in model. First, the original VAE process extracts feature representation from side information about items and then integrates them into latent factor of items in pair-wise ranking model. Second, the pair-wise ranking model captures special relationship $\delta_{ijk} = r_{ij} - r_{ik}$, which delegates the preference of user i on item j and k. Unlike the point-wise approach directly predicting the value $r_i j$, pair-wise approach instead classifies the difference of $r_{ij} - r_{ik}$.

CVDR change the loss function of CDR as follows:

$$\begin{aligned}\mathcal{L} = &- \sum_{ijk} \frac{c_{ijk}}{2} \left(\delta_{ijk} - \left(u_i^T v_j - u_i^T v_k\right)\right)^2 - \frac{\lambda_w}{2} \sum_j \left(W_l^2 + b_l^2\right) \\ &- \frac{\lambda_u}{2} \sum_i u_i^T u_i - \frac{\lambda_v}{2} \sum_j \left(v_j - Z_{j*}^T\right)^2 - \frac{\lambda_n}{2} \sum_j \left(X_{L,j*} - X_{j*}\right)^2\end{aligned} \tag{3.4}$$

where confidence parameter c_{ijk} denotes how much user i prefers item j than item k and $\lambda_u, \lambda_v, \lambda_n$ are the regulation hyperparameters.

The first term in equation 3.4 extracts user preference from implicit matrix R to construct pair-wise ranking loss, while the fourth term integrates content of items. Therefore, if two item j and k have similar side information (i.e. similar Z), the distance between v_j and v_k will be reduced. Z serves as a bridge between pair-wise ranking and VAE model. When λ_v/λ_n goes to

positive infinity, the disappeared reconstruction error will lead to invalid feature representation Z, meanwhile Z dominate the learning process of V. On the other hand, when λ_v/λ_n approaches to zeros, CVDR will decouple into two models and the learned V is not influenced by side information about items. Both extreme cases demonstrate bad performance in the experiments.

Prediction. After learning the optimal parameters U, V, W^+, they predict R_{ij} from its expectation:

$$E\left[R_{ij} \mid U, V, W^+, \ldots\right] \approx u_i^T \left(Z + \epsilon_j\right) = u_i v_j,$$

(where ϵ_j is the latent item offset) and then a ranked list of items is generated for each user based on these prediction values.

3.6 A Fair Recommendation Algorithm

In this section we will describe a fair re-ranking algorithm called Deterministic Greedy (DetGreedy) algorithm that has been proposed in the work [20], and tries to re-rank an existing ranking list of job candidates. The fairness of a ranked recommendation list should obey to the following constraint, so that there is no state (e.g., male or female) of an attribute (e.g., genre) that concerns the candidates which is benefited versus the other states of the protected attribute:

$$\exists\ r \text{ s.t. } \forall k \leq |\tau_r| \ \ \& \ \forall s_i \in S,\ \text{counts}_k\left(s_i\right) \geq \lfloor p_{s_i} \cdot k \rfloor \quad (3.5)$$

The above formula ensures that there is at least one ranking request r, such that the generated recommendation list τ_r does not break the condition $counts_k(s_i) \geq \lfloor p_{s_i} \cdot k \rfloor$ for each k and attribute state s_i of an attribute a, where $counts_k(s_i)$ denotes the number of candidates with attribute state s_i among the top k candidates and p_{s_i} is the desired proportion of candidates possessing attribute state s_i (e.g., 50% males).

The **DetGreedy** algorithm tries to re-rank an existing list of job candidates by searching each protected attribute's state s_i and if it violates certain predefined conditions, it replaces it with another state of the protected attribute, which has the next maximum score.

We present (in pseudocode form) the **DetGreedy** algorithm, as shown in Algorithm 1. Among the input data of the algorithm is the *Data* table

of data of attribute a, the P table, with the propabilities of appeareance of attribute a and k_{max} the number of desired results.

In the preudo-code format 1, $counts[s_i]$ denotes the number of candidates with state s_i of attribute a among the top k recommended candidates, p_{s_i} the desired proportion of candidates possessing state s_i of attribute a and $Grade_{s_i,j}$ denotes the grade of j^{th} candidate possessing state s_i.

The **DetGreedy** algorithm in each iteration (lines 6-17), counts two sets $(StatesBelowMin, StatesBelowMax)$. The $StatesBelowMin$ is the set of states of an attribute that their representation is below the desired proportion. The $StatesBelowMax$ is the set of states of an attribute that their representation is below the maximum possible allowed, so that they do not have advantage over the other states of the attribute. If inside the $StatesBelowMin$ set (line 9) there one or more states, then choose the state (s_i) that has the highest next grade $(Grade_{s_i,j})$ among them (line 10). Otherwise, choose the candidate with attribute state with the highest next grade among those that belong to $StatesBelowMax$ set (line 12). At the end of each iteration $rankedCandidatesList$ (line 14) maintains the number of best candidates included in the top k results and $rankedGradesList$ (line 15) the grades of best candidates.The algorithm outputs the best re-ranking list of candidates with their grades (line 18). In the following, we present the **DetGreedy** algorithm in pseudo-code format 1.

The **DetGreedy** algorithm, however, manages a combined attribute without taking into account the probabilities of appearance of the attributes of which it is composed. This problem motivates us to create an extension of the algorithm, which also takes into account the probability of appearance of the individual protected attributes that make up a combined attribute. In section 6.1 shows in more detail that the algorithm fails.

3.7 eXpanded Group Fairness Algorithm

The e**X**panded GF **xGF** algorithm extends the **DetGreedy** algorithm and tries to improve the results of **DetGreedy** algorithm for each individual attribute of a combination of two or more attributes.

In this subsection we present (in pseudocode form) the **xGF** algorithm, as shown in Algorithm 2. Among the input data of the algorithm, S is a set that holds the states of each attribute, P is a set with the propabilities of occurence of each individual attribute $(a, b, .., n)$ and k_{max} the number of

Algorithm 1 DetGreedy

Input:
Data of attributes: $Data = \{S, G\}$
where S are the states of attribute a : $S = \{s_1, s_2, ..., s_n\}$,
and G is the grades of candidates:
$G = \{\{Grade_{s_1,1}, Grade_{s_1,2}, ..., Grade_{s_1,m}\}, \{Grade_{s_2,1}, Grade_{s_2,2}, ..., Grade_{s_2,m}\},$
$\{Grade_{s_n,1}, Grade_{s_n,2}, ..., Grade_{s_n,m}\}\}$
Propabilities of appearence of attribute a: $P = \{p_{s_1}, p_{s_1}, ..., p_{s_n}\}$
Number of desired top results: k_{max}
Output:
An ordered list of candidates with their grades: $[rankedCandidatesList, rankedGradesList]$

procedure DetGreedy($Data, P, k_{max}$)
  for each state s_i of attribute a **do**
    $counts\,[s_i] := 0$
  end for
  rankedCandidatesList := [ ]; rankedGradesList := [ ]
  for each recomended candidate $k \in \{1, \ldots, k_{\max}\}$ **do**
    StatesBelowMin := $\{s_i : counts\,[s_i] < \lfloor k \cdot p_{s_i} \rfloor\}$
    StatesBelowMax := $\{s_i : counts\,[s_i] \geq \lfloor k \cdot p_{s_i} \rfloor \,\&\, counts\,[s_i] < \lceil k \cdot p_{s_i} \rceil\}$
    if StatesBelowMin $\neq \emptyset$ **then**
      nextState:= $\text{argmax}_{s_i \in \text{ StatesBelowMin}}\ Grade_{s_i,\ counts\ [s_i]}$
    else
      nextState := $\text{argmax}_{s_i \in \text{ StatesBelowMax}}\ Grade_{s_i,\ counts\ [s_i]}$
    end if
    rankedCandidatesList [k] := nextState
    rankedGradesList[k] := $Grade_{\text{nextState},counts[\text{nextState }]}$
    $counts$[nextState]++
  end for
  return [rankedCandidatesList, rankedGradesList]
end procedure

candidates to be recommended.

In the preudo-code of Algorithm 2, $counts[s_i^k]$ denotes the number of candidates with attribute state s_i^x of the attribute x so far, $p_{s_i^x}$ denotes the the desired proportion of candidates possessing attribute state s_i^x of attribute x and $Grade_{s_i^{comb},j}$ denotes the score of j^{th} candidate of combined attribute state $s_i^{comb} \rightarrow s_i^a s_i^b..s_i^n$.

The **xGF** algorithm in each iteration k (lines 7-38), follows the steps of **DetGreedy** algorithm, with one difference, counts the two sets $(StatesBelowMin, StatesBelowMax)$ for each attribute $a, b, .., n$ of a combined attribute *comb* (lines 8-11). Then, if even one individual attribute $a, b, .., n$ has filled his $StatesBelowMin$ set then the $StatesOpen$ set (line 14) is filled with this set, else the $StatesOpen$ set (line 16) is filled with $StatesBelowMax$ set of this attribute. Then it searches next from the current position for the first violations of the minimum representation for each attribute and fills the set $NextViolations$ (line 21) and if there are such violations, in the set $StateOpen$ it keeps only the states in common with the set $NextViolations$ (line 23). Finally the $StateOpenComb$ set for the combined attribute (line 29) is filled with the combinations of states in $StatesOpen$ set. The next steps are the same as in the **DetGreedy** algorithm by choosing each time the state belonging to the $StatesOpenComb$ set with the highest score (line 31). At the end of each iteration $rankedCandidatesList$ (line 32) it maintains the number of best candidates included in the top k results, $rankedGradesList$ the grades of best candidates (line 33). The algorithm outputs the best re-ranking list of candidates with their grades (line 39). In the following, we present the **xGF** algorithm in pseudo-code format 2.

We performed experiments on different models examining and trying to improve their fairness on selected attributes, using the proposed fairness metric and the **DetGreedy** fair re-ranking algorithm of work [20] and our proposed **xGF** fair re-ranking algorithm.

Algorithm 2 xGF

Input:

$S^{comb} = \{s_1^{comb}, s_2^{comb}, ..., s_m^{comb}\}$, $S^a = \{s_1^a, s_2^a, ..., s_m^a\}$, $S^b = \{s_1^b, s_2^b, ..., s_m^b\}$, $S^n = \{s_1^n, s_2^n, ..., s_m^n\}$: S is a set that holds the states of each attribute.

$G = \{\{Grade_{s_1^{comb},1}, Grade_{s_1^{comb},2}, ..., Grade_{s_1^{comb},k}\}, \{Grade_{s_2^{comb},1}, Grade_{s_2^{comb},2}, ..., Grade_{s_2^{comb},k}\}, \{Grade_{s_n^{comb},1}, Grade_{s_n^{comb},2}, ..., Grade_{s_n^{comb},k}\}\}$: G is a set that holds the ranked grades of candidates for each state of an attribute.

$P_a = \{p_{s_1^a}, p_{s_1^a}, ..., p_{s_m^a}\}$, $P_b = \{p_{s_1^b}, p_{s_1^b}, ..., p_{s_m^b}\}$, $P_n = \{p_{s_1^n}, p_{s_1^n}, ..., p_{s_m^n}\}$: P is a set that holds the propabilities of required occurrence of each state of an attribute.

k_{max}: Number of job candidates to be recommended:

Output:
An ordered list of job candidates with their grades: $[rankedCandidatesList, rankedGradesList]$

procedure xGF(S, G, P, k_{max})
  for each attribute $x \in \{a, b, .., n, comb\}$ **do**
    $counts\left[s_i^x\right] := 0$
  end for
  rankedCandidatesList := [ ]; rankedGradesList := [ ]
  $StatesBelowMin := [\,]; StatesBelowMax := [\,]$
  for each recomended candidate $k \in \{1, \ldots, k_{\max}\}$ **do**
    for each attribute $x \in \{a, b, ..., n\}$ **do**
      $StatesBelowMin[x] := \left\{s_i^x : counts\left[s_i^x\right] < \left\lfloor k \cdot p_{s_i^x} \right\rfloor\right\}$
      $StatesBelowMax[x] := \left\{s_i^x : counts\left[s_i^x\right] \geq \left\lfloor k \cdot p_{s_i^x} \right\rfloor \,\&\, counts\left[s_i^x\right] < \left\lceil k \cdot p_{s_i^x} \right\rceil\right\}$
    end for
    for each attribute $x \in \{a, b, ..., n\}$ **do**
      if $StatesBelowMin[x] \neq \emptyset$ **then**
        $StatesOpen[x] := StatesBelowMin[x]$
      else
        $StatesOpen[x] := StatesBelowMax[x]$
      end if
    end for
    for each attribute $x \in \{a, b, ..., n\}$ **do**
      for each position $pos \in \{k+1, .., k_{max}\}$ **do**
        $NextViolations[x][pos] := \left\{s_i^x : counts\left[s_i^x\right] < \left\lfloor pos \cdot p_{s_i^x} \right\rfloor\right\}$
        if $NextViolations[x][pos] \neq 0$ **then**
          $StatesOpen[x] := StatesOpen[x] \cap NextViolations[x][pos]$
          break
        end if
      end for
    end for
    for each attribute $x \in \{a, b, ..., n\}$ **do**
      $StatesOpenComb := StatesOpenComb \otimes StatesOpen[x]$
    end for
    nextState:= $\text{argmax}_{s_i^{comb} \in\ StatesOpenComb}\ Grade_{s_i^{comb},\ counts\ [s_i^{comb}]}$
    rankedCandidatesList [k] := nextState
    rankedGradesList[k] := $Grade_{\text{nextState},counts[\text{nextState }]}$
    $counts$[nextState]++
    for each individual attribute x which combines to nextState **do**
      $counts[s_i^x]$++
    end for
  end for
  return [rankedCandidatesList, rankedGradesList]
end procedure

Chapter 4

Experimental Evaluation

In this chapter we will show the characteristics of the datasets that have been used. We will also compare the results of different basic algorithms as well as the results we get in the proposed model CVDR before and after the application of the re-ranking algorithm xGF. Each model has been properly parameterized by searching the field of parameters. In detail, the performance of the algorithms is shown through evaluation metrics.

4.1 Dataset description

We used two datasets, Recsys16 which is data provided by XING after the RecSys Challenge 2016 and Careerbuilder12 which is from an open Kaggle competition, called Job Recommendation Challenge, provided by the online employment Web site CareerBuilder. In the following subsections we show the characteristics of each dataset, but also the percentage we used from each one and the pre-processing of each.

4.1.1 XING Dataset

The RecSys16 dataset contains six diferrent interaction types that were performed on the job items. In this study, we only keep the click, bookmark and apply interactions as positive interactions (the user interests about the job, which interact with). We remove the delete recommendation and recruiter interest interactions as these are irrelevant in our setting. Moreover, we discard impression interactions as they are created when XING shows a

job to a user. The dataset consists of interactions from a period of three months. Also, the RecSys16 dataset contains content features about the job postings, such as career_level or type of employment. From this set, we select ten features as content-based input for our approaches and discarded the numeric IDs of *created_at* because time is not important feature for no section-based approaches. More specifcally, from RecSys16, we use the following features: *title*, *career_level, discipline_id, industry_id, country, region, latitude, longitude, employment* and *tags*. The *region* content feature is a categorical feature with 17 possible value, like the *employment* feature with 6 values. The *discipline_id* is a categorical feature, that represent disciplines such as consulting or human resources. The categorical feature *career_level* can take 7 values, the *title* represents the title of the job, *latitude* and *longitude* represent the coordinates of the job position, the *industry_id* represents industries such as finance, *country* denotes the code of the country in which the job is offered and *tags* represent concepts that have been extracted from the tags, skills or company name. In Table 4.1 we see the job features, which we used, aggregated.

For the experiments on fairness we use three user features. The RecSys16 dataset contains content features about the user resumes, such as career level or job roles. From this set, we select three features as content-based input for our fairness re-ranking algorithms. We use the categorical feature *experience_ years_experience* which can take 8 values and represents the total years of experience of the user, the categorical feature *region* which can take 17 values and represents the region in which the user live and *discipline_id* which describes disciplines in which the user belongs. For the needs of the experiments we divided the users into 2 groups according to their years of experience and 3 groups according to region in which they live. In Table 4.2 we see the user features of the dataset.

In this dataset we binarized the ratings to create an implicit dataset with 0 in usage rating meaning that the user does not prefer this job and 1 meaning the opposite. The ratings 1,2 and 3 of the original dataset (positive interactions), change to 1 and the ratings 4 change to 0, because are negative interactions for a user. Still due to low computing power, in this part, we only used the latest and most recent user interactions and a total of 176679 out of 8826678 records. Each experiment was conducted by splitting the recordings into 80% for the training set and 20% for the test set. In Table 6.1, we show the statistics of training and test set. In the remainder of the thesis we will refer to the Recsys16 dataset as the XING dataset for simplicity.

Table 4.1: The used job features of Recsys16 dataset.

Feature	Description
id	anonymized ID of the item
title	concepts that have been extracted from the job title of the job posting
career_level	career level ID
discipline_id	anonymized IDs represent disciplines such as "Consulting", "HR", etc.
industry_id	anonymized IDs represent industries such as "Internet", "Automotive", "Finance", etc.
country	code of the country in which the job is offered
region	is specified for some users who have as country de.
latitude	latitude information (rounded to ca. 10km)
longitude	longitude information (rounded to ca. 10km)
employment	the type of employment
tags	concepts that have been extracted from the tags, skills or company name

Table 4.2: The user features of Recsys16 dataset.

Feature	Description
id	anonymized ID of the user
jobroles	comma-separated list of job role terms (numeric IDs) that were extracted from the user's current job title
career_level	career level ID
discipline_id	**anonymized IDs represent disciplines such as "Consulting", "HR", etc.**
industry_id	anonymized IDs represent industries such as "Internet", "Automotive", "Finance", etc.
country	describes the country in which the user is currently working
region	**is specified for some users who have as country de**
experience_n_entries_class	identifies the number of CV entries that the user has listed as work experiences
experience_years_experience	**is the estimated number of years of work experience that the user has**
experience_years_in_current	is the estimated number of years that the user is already working in her current job
edu_degree	comma-separated fields of studies that the user studied
edu_fieldofstudies	entries refer to broad field of studies such as Engineering, Economics and Legal, ...

Table 4.3: The statistics of Recsys16 used data

	Training Data	Test Data
Users	13607	5590
Items	80688	13754
Ratings	115191	18456

4.1.2 CAREER BUILDER Dataset

The CareerBuilder12 dataset contains job applications from a period of almost three months. No other interaction types are present in this dataset. The interactions in the dataset happen over 13 weeks. Thus, similar to the previous dataset, it happens over almost three months. Regarding content features, the CareerBuilder12 dataset contains textual descriptions of the jobs as well as categorical data for the location. From the content data, the 55 diferent states are used in the form of one-hot encodings. In this thesis we adopt the way of preprocessing the data of report [2]. In that work the authors categorize the context features *Description, Requirements* and *Title* of the original data, using a latent Dirichlet allocation models and create the job context features: *City, State, Country, Zip5, RecTopic, DescTopic, TitTopic*. We don't use the feature *City*, because increases the total vocabulary with disastrous results for our proposed model and we use all the other features. The categorical feature *Country* denotes the code of the country in which the job is offered, *Zip5* its postal code of the job position, *RecTopic* represents the topic of the requirements, *DesTopic* represents the topic of the description and *TitTopic* represents the topic of the title, of the job. In Table 4.4 we see the job features, which we used, aggregated.

For the experiments on fairness we use three user features. At first categorize the user context feature *Major* together with the job feature *Requirements* of the original data, because they are related to each other, using a latent Dirichlet allocation model and create again the job context feature: *RecTopic* and the new user feature: *MajTopic*. The Careerbuilder12 dataset contains content features about the user resumes, such as graduation date or degree type. From this set, we select three features as content-based input for our fairness re-ranking algorithms. We use the feature *Total Years Experience* which represents the total years of experience of the user, the feature *State* which represents the state of America in which the user live and the feature *MajTopic* which represents the topic of work area of user. For the needs of the experiments we divided the users into 2 groups according to their years of experience and 3 groups according to state in which they live. In Table 4.5 we see the user features of the dataset.

In this dataset we binarized the ratings to create an implicit dataset with 0 (negative interaction) in usage rating meaning that the user does not prefer this job and 1 (positive interaction) meaning the opposite. The original dataset has not positive and negative interactions. We use the interactions

Table 4.4: The used job features of CareerBuilder12 dataset.

Feature	Description
JobID	job posting unique id number
TitTopic	topic title of job posting
DescTopic	topic description of job posting
RecTopic	topic requirements of job posting
State	state of job posting
Country	country of job posting
Zip5	zip5 of job posting

Table 4.5: The user features of CareerBuilder12 dataset.

Feature	Description
UserID	user unique id number
City	city where the user resides
State	**state where the user resides**
Country	country where the user resides
ZipCode	zip code of user
DegreeType	degree type of user
MajTopic	**topic work area of user**
GraduationDate	graduation date of user
WorkHistoryCount	work history count of user
TotalYearsExperience	**total years of experience of user**
CurrentlyEmployed	is the answer 'yes' or 'no' the user currently employed
ManagedOthers	is the answer 'yes' or 'no' the user managed others
ManagedHowMany	how many managed people from the user

of original dataset as positive and create random negative interactions for every user on random jobs, which has not interact with. The total amount of positive and negative interactions for a user is the same. Still due to low computing power, in this part, we only used the user interactions of jobs of

Table 4.6: The statistics of Careerbuilder12 used data

	Training Data	Test Data
Users	40303	23901
Items	92673	36811
Ratings	290925	63446

job7 features window of dataset and a total of 363668 out of 3206222 records. Each experiment was conducted by splitting the recordings into 80% for the training set and 20% for the test set. In Table 6.2, we show the statistics of training and test set. In the remainder of the thesis we will refer to the Careerbuilder12 dataset as the CAREER BUILDER dataset for simplicity.

4.2 Evaluation

In the following subsections we show the accuracy, beyond-accuracy and fairness metrics. Then we compare the basic models with each other in each dataset in terms of their performance with the evaluation metrics, as well as graphical evaluation methods, before and after re-ranking, with the fair re-ranking algorithms. Finally we compare the proposed method CVDR with its various variations in each dataset, before and after re-ranking, with the fair re-ranking algorithms and we analyze the conclusions of each one.

4.2.1 Evaluation Metrics

In this subsection we describe the accuracy, beyond-accuracy and fairness metrics, which we have used to compare the performance of the different models.

Accuracy Metrics

We use two accuracy metrics, **NDCG** and **Recall**, to compare our models. Their definition is shown below.

The **Recall** metric is based on the number of proposed items recorded in each of the cells of the confusion matrix, shown in the Table 4.7 and has the following mathematical formula:

Table 4.7: Confusion Matrix

		Predicted Values	
		Positive	Negative
Actual Values	Positive	$TruePositive$	$FalseNegative$
	Negative	$FalsePositive$	$TrueNegative$

$$\text{Recall } = \frac{\text{True Positive } (TP)}{\text{True Positive } (TP) + \text{ False Negative } (FN)} \tag{4.1}$$

Recall measures the number of relevant items (True Positives) suggested by the recommendation algorithm divided by the number of all relevant elements (True Positives and False Negatives). The use of the Recall metric is applicable to the problem we are trying to solve, because we know which jobs the user likes (positive interactions), while for the rest, which he has not interact, we do not know precisely, if he really does not like them or not has seen them yet. Because of this we don't use the Precision metric.

The Recall metric does not take position into account ranking of items within the recommendation list, so we use the **NDCG** metric, which is normalized because it is the ratio of Discounted Cumulative Gain (DCG) to Ideal Discounted Cumulative Gain (IDCG).

The DCG considers the relative position of the proposed elements within the list of recommendations L. For each element in L there is a "profit", which however we calculate cumulatively reduced based on the ranking position of the element under consideration. The lower, that is, in the recommendation list there is an item, the smaller the "profit" will be.Therefore, the total "profit" accumulated up to a certain ranking position pos defined as follows:

$$DGC@pos = rel_1 + \sum_{i=2}^{pos} \frac{rel_i}{\log_2 i} \tag{4.2}$$

where rel_i is the degree of relevance of the item in position i of the sub's preference list user review.

The "ideal profit" results from the fact that the items in L are ranked in descending order order by their degree of relevance. Thus, the ideal DCG is calculated as follows:

$$IDGC@pos = rel_1 + \sum_{i=2}^{|N|-1} \frac{rel_i}{\log_2 i} \tag{4.3}$$

where N is the number of recommended items in the list.

After all the mathematical formula of the **NDCG** metric is:

$$NDCG@pos = \frac{DCG}{IDCG} \tag{4.4}$$

Beyond-Accuracy Metrics

We use one beyond-accuracy metric, **Kendall**τ, for measuring the the distance of the job recommendations lists of two users with equal qualifications, to compare our models. Its definition is shown below.

Let c denote the total number of job pairs in *concordance* (i.e. number of pairs that have the same order position in the job recommendation list) and d the total number of job pairs in *disconcordance* (number of pairs that have a different order position in the job recommendation list). The **Kendall**τ computes the difference between the two aforementioned quantities normalized by the total number of job pairs $\begin{pmatrix} n \\ 2 \end{pmatrix}$. We emphasize that for a number n of jobs the total number of job pairs (per two) that can be generated is $\frac{n!}{2!(n-2)!}$. Thus, the **Kendall**τ distance function for two job recommendation lists $\rho, \sigma \in S_n$ is defined by the following equation:

$$Kendall\tau(\rho, \sigma) = \frac{c - d}{\begin{pmatrix} n \\ 2 \end{pmatrix}} \tag{4.5}$$

For two job recommendation lists ρ and σ, then, the comparison for their pair of jobs (i, j) is in *(concordant)* if the rank order of the jobs $\{\rho_i, \sigma_i\}$ and $\{\rho_j, \sigma_j\}$ in the two lists is the same:

$$\rho_i > \rho_j, \quad \sigma_i > \sigma_j \quad \text{or} \quad \rho_i < \rho_j, \quad \sigma_i < \sigma_j.$$

In contrast, for two job recommendation lists ρ and σ the comparison for their job pairs (i, j) is in *(disconcordant)* if the rank order of the job pairs $\{\rho_i, \sigma_i\}$ and $\{\rho_j, \sigma_j\}$ in the two lists is opposite:

$$\rho_i > \sigma_j, \quad \rho_i < \sigma_j \quad \text{or} \quad \rho_i < \sigma_j, \quad \rho_i > \sigma_j$$

We emphasize that when $\rho_i = \sigma_j$, then a job pair is characterized neither as *concordant* nor as *disconcordant*.

In the work [21] the authors define the minimum kendall distance $Kendall\tau@k$ between two top k lists $\tau 1$ and $\tau 2$.

If τ is a top k list and σ is a permutation on $D \supseteq D_\tau$ and D_τ the members of the top k list τ, then we say that σ is an extension of τ, which we denote $\sigma \succeq \tau$, if $\sigma(i) = \tau(i)$ for all $i \in D_\tau$.

The minimum kendall distance $Kendall\tau@k$ between two top k lists $\tau 1$ and $\tau 2$ is defined to be the minimum value of $Kendall\tau(\sigma 1, \sigma 2)$ where $\sigma 1$ and $\sigma 2$ are each permutations of $D_{\tau_1} \cup D_{\tau_2}$ and where $\sigma_1 \succeq \tau_1$ and $\sigma_2 \succeq \tau_2$.

Fairness Metrics

We use one fairness metric, **xNDKL**, to compare our models. Its definition is shown below.

The **NDKL** metric having the benefit of providing a single measure of bias over all attribute states and a holistic view over the whole ranked list. Given a ranked list τ_r of candidates for a search request r, the **NDKL** of τ_r has the following formula:

$$\mathrm{NDKL}\left(\tau_r\right) = \frac{1}{Z} \sum_{i=1}^{|\tau_r|} \frac{1}{\log_2(i+1)} d_{KL}\left(D_{\tau_r^i} \| D_r\right), \tag{4.6}$$

where $d_{KL}\left(D_1 \| D_2\right) = \sum_j D_1(j) \log_e \frac{D_1(j)}{D_2(j)}$ is the KL-divergence of distribution D_1 with respect to distribution D_2, $Z = \sum_{i=1}^{|\tau_r|} \frac{1}{\log_2(i+1)}$ and $D_{\tau_r^i}$ and D_r denote the discrete distribution assigning to each attribute state in set of attribute states S, the proportion of candidates having that state, over the top i candidates in the given ranked list τ_r and over the desired distribution respectively.

The **NDKL** metric is non-negative, with a larger value denoting greater divergence between the two distributions and a shorter value denoting greater convergence between the two distributions.

We extend the use of the **NDKL** metric in the case that we want the proposed algorithm to be fair and for the dispersion of the recommendations in all protected attributes by summing up the **NDKL** results for each protected attribute. The extension of **NDKL** is called e**X**panded Normalized Discounted cumulative KL-divergence (**xNDKL**), of τ_r and can be represented with the following formula:

$$\text{xNDKL}\left(\tau_r\right) = \sum_{a \in \mathcal{A}} \frac{1}{Z} \sum_{i=1}^{|\tau_r|} \frac{1}{\log_2(i+1)} d_{KL}\left(D_{\tau_r^i} \| D_r\right), \tag{4.7}$$

where a denotes the type of the protected attribute (e.g., gender, age, etc.), where $a \in \mathcal{R}$.

4.3 Evaluation Strategy

As mentioned above, we divided the recordings of each dataset into 80% training and 20% test pieces. We conducted experiments on different algorithms and parameters for each algorithm with the help of the Cornac library and finally ended up selecting the four best models CDR, NeuMF, Maximum Margin Matrix Factorization (MMMF) and Indexable Bayesian Personalized Ranking (IBPR), based on the different evaluation metrics. The best parameters of used models are shown in Appedix A.

4.4 Comparison of Basic Models

In this section we will present the results of the experiments on the basic models CDR(UJ), NeuMF(UJ), MMMF(UJ) and IBPR(UJ), separately for each dataset and compare the results before and after the application of fair re-ranking algorithms DetGreedy and xGF.

All models use user-job interaction information (UJ) and only this. We have suitably modified the CDR model so that it does not use job features information, so that it can be compared fairly with the rest of the basic models and we call this model CDR(UJ).

We used the application of the DetGreedy and xGF algorithms to improve the fairness of the basic models regarding features of the job seeker (user features: *discipline_id* and *region* of the XING dataset, *MajTopic* and *State* of the CAREER BUILDER dataset).

In each dataset we conducted the experiments to measure the fairness of algorithms in recommending jobs, which have the same selected attributes states, such as an individual user, with the probability 3/4, and the jobs, with different states of selected attributes, with probability 1/4. For measuring the the distance of the job recommendations lists of two users with equal

qualifications, because of low computing power we measure in 3 runs the distance of 20 random users.

4.4.1 Basic Models on XING dataset

In the Tables 4.8 and 4.9 we see the results of the experiments on the XING dataset before and after the application of DetGreedy and xGF algorithms, respectively. In the Figure 4.1 we see the performance of the algorithms in Receiver Operating Characteristic (ROC) Curve and in the Figure 4.2 their performance in Precision-Recall (PR) Curve.

Table 4.8: Results of basic models on XING dataset before re-ranking

	Recall@20	NDCG@10	xNDKL@10	Kendallτ@10
CDR(UJ)	**0.3953**	**0.2259**	1.12	[0.22, 0.15, 0.26]
NeuMF(UJ)	0.3900	0.2137	**1.04**	[0.19, 0.08, 0.19]
MMMF(UJ)	0.0077	0.0032	1.11	[0.23, 0.08, 0.28]
IBPR(UJ)	0.1187	0.0659	1.13	[0.17, 0.15, 0.28]

Table 4.9: Results of basic models on XING dataset after re-ranking

DetGreedy				
	Recall@20	NDCG@10	xNDKL@10	Kendallτ@10
CDR(UJ)	**0.3953**	**0.2169**	0.60	[0.31, 0.09, 0.14]
NeuMF(UJ)	0.3900	0.2073	**0.57**	[0.21, 0.05, 0.12]
MMMF(UJ)	0.0077	0.0031	0.61	[0.24, 0.09, 0.16]
IBPR(UJ)	0.01187	0.0633	0.61	[0.23, 0.09, 0.18]
xGF				
	Recall@20	NDCG@10	xNDKL@10	Kendallτ@10
CDR(UJ)	**0.3953**	0.1400	**0.23**	[0.15, 0.09, 0.17]
NeuMF(UJ)	0.3900	**0.1410**	**0.23**	[0.18, 0.17, 0.14]
MMMF(UJ)	0.0077	0.0020	**0.23**	[0.19, 0.18, 0.16]
IBPR(UJ)	0.1187	0.0402	**0.23**	[0.14, 0.10, 0.15]

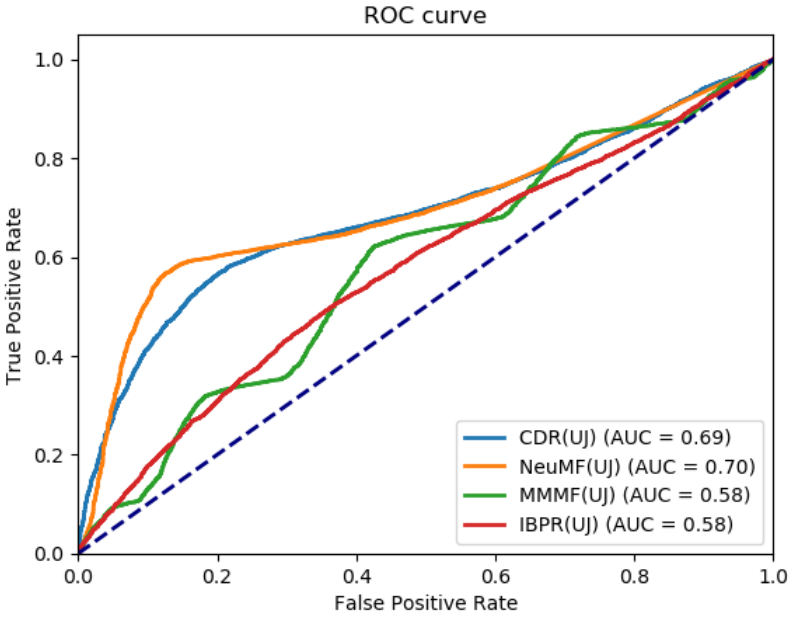

Figure 4.1: ROC Curve of basic models on XING dataset.

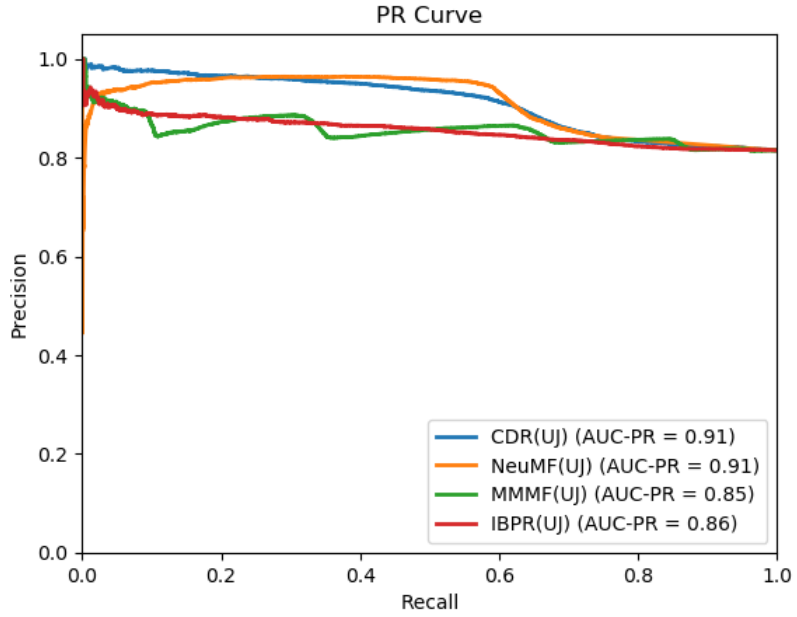

Figure 4.2: PR Curve of basic models on XING dataset.

Before applying the re-ranking algorithms, from the above results we see that the CDR(UJ) model has the best performance in the Recall@20 and NDCG@10 metrics, the third best performance in the xNDKL@10 metric, but also the second best performance in the ROC Curve and finally the best performance itself with the NeuMF(UJ) algorithm on the PR Curve. Second best model is NeuMF(UJ), which has the second best performance in the Recall@20 and NDCG@10 metrics, the best performance in the xNDKL@10 metric, the best performance in the ROC Curve and the best in the PR Curve together with the CDR(UJ) model. The IBPR(UJ) model comes third in the performance ranking, but shows the second best performance in the xNDKL@10 metric. Finally, the performance of the MMMF(UJ) model is not that good in accuracy and fairness metrics.

After re-ranking, the NDCG@10 and the xNDKL@10 metrics decrease, showing us that there is a trade-off between accuracy and fairness. In this results, the proposed model CDR(UJ) is the model with the best performance in the majority of the metrics in both re-ranking algorithms. Between the two re-ranking algorithms the proposed xGF improves more the fairness of the basic models while decreasing the accuracy.

The results in the Kendallτ@10 metric show us that the xGF methodically removes users who had similar suggestions, while bringing together users who did not have the same suggestions before the re-ranking. So similar users are treated the same, the algorithm DetGreedy does not do this methodically so it does not respect the rules of individual fairness.

Observing the above results we see that the performance of CDR(UJ) and NeuMF(UJ) models is relatively close. But, we decide to continue improving the CDR(UJ) model in the continuation of the work, because it is ranked first. The results in the MMMF(UJ) model in accuracy metrics are expected, due to the method is designed to work with sparse explicit data, i.e. it assume that there are many missing values and to have much less pairs than in an implicit setting.

4.4.2 Basic Models on CAREER BUILDER dataset

In the Table 4.10 and 4.11 we see the results of the experiments on the CAREER BUILDER dataset before and after the application of DetGreedy and xGF algorithms, respectively. In the Figure 4.3 we see the performance of the algorithms in ROC Curve and in the Figure 4.4 their performance in PR Curve.

Table 4.10: Results of basic models on CAREER BUILDER dataset before re-ranking

	Recall@20	NDCG@10	xNDKL@10	Kendallτ@10
CDR(UJ)	**0.0337**	0.0068	0.97	[0.22, 0.19, 0.05]
NeuMF(UJ)	0.0188	**0.0080**	1.13	[0.33, 0.26, 0.28]
MMMF(UJ)	0.0020	0.0007	**0.96**	[0.14, 0.28, 0.004]
IBPR(UJ)	0.0062	0.0023	0.97	[0.14, 0.15, 0.02]

Table 4.11: Results of basic models on CAREER BUILDER dataset after re-ranking

DetGreedy				
	Recall@20	NDCG@10	xNDKL@10	Kendallτ@10
CDR(UJ)	**0.0337**	0.0065	**0.48**	[0.40, 0.12, 0.0]
NeuMF(UJ)	0.0188	**0.0077**	0.54	[0.55, 0.16, 0.12]
MMMF(UJ)	0.0020	0.0007	0.49	[0.30, 0.10, 0.0]
IBPR(UJ)	0.0062	0.0022	**0.48**	[0.30, 0.0, 0.07]
xGF				
	Recall@20	NDCG@10	xNDKL@10	Kendallτ@10
CDR(UJ)	**0.0337**	0.0048	0.25	[0.17, 0.10, 0.12]
NeuMF(UJ)	0.0188	**0.0050**	**0.24**	[0.19, 0.11, 0.08]
MMMF(UJ)	0.0020	0.0005	0.25	[0.11, 0.11, 0.13]
IBPR(UJ)	0.0062	0.0016	0.25	[0.10, 0.10, 0.04]

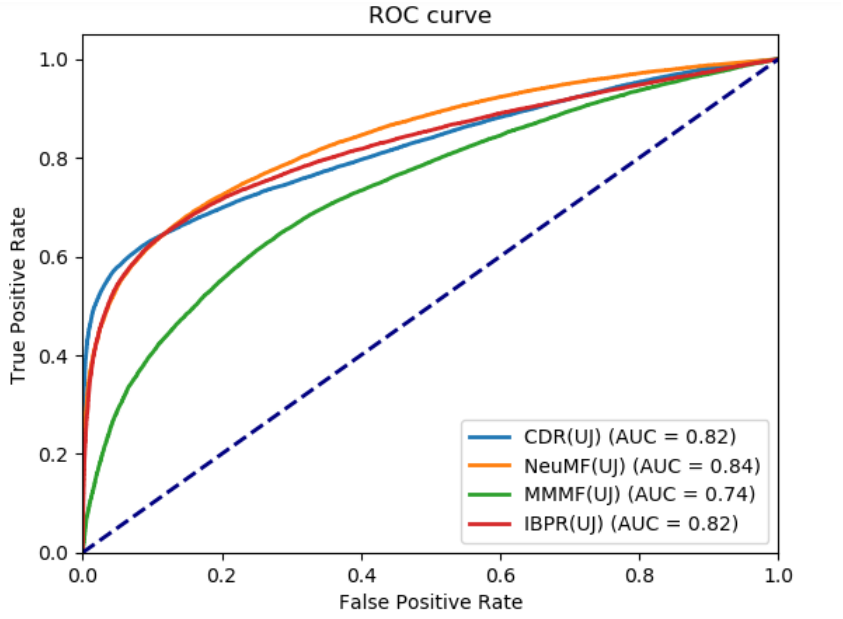

Figure 4.3: ROC Curve of basic models on CAREER BUILDER dataset.

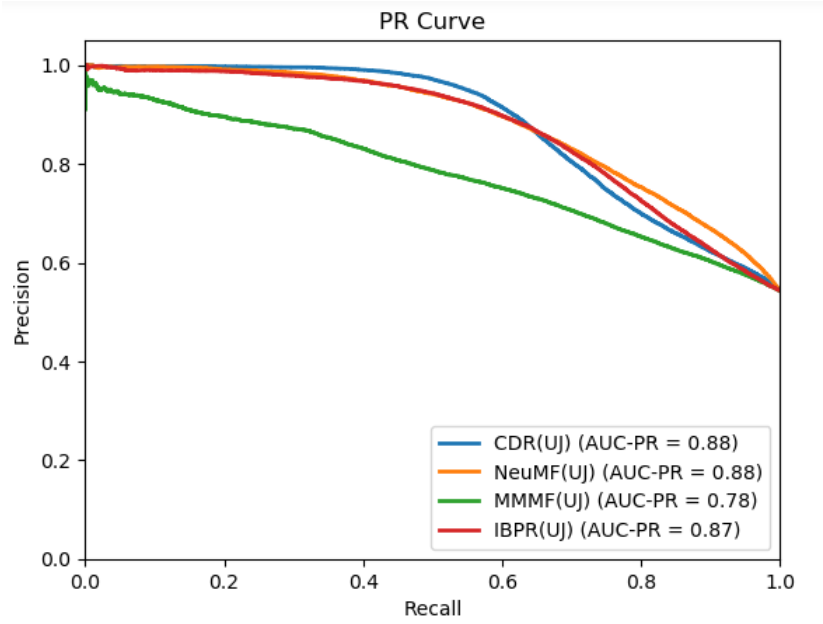

Figure 4.4: PR Curve of basic models on CAREER BUILDER dataset.

Before applying the re-ranking algorithms, from the above results we see that the CDR(UJ) model has the best performance in the Recall@20 metric and second best in the NDCG@10 metric, the second best performance in the xNDKL@10 metric, same as the IBPR(UJ) algorithm, but also the second best performance in the ROC Curve same as IBPR(UJ) and finally the best performance itself with the NeuMF(UJ) algorithm on the PR Curve. Second best model is NeuMF(UJ), which has the second best performance in the Recall@20 metric, the best performance in the NDCG@10 metric, the worst performance in the xNDKL@10 metric, but the best performance in the ROC Curve and the best in the PR Curve together with the CDR(UJ) model. The IBPR(UJ) model comes third in the performance ranking, but shows the second best performance in the xNDKL@10 metric. Finally, the performance of the MMMF(UJ) model is not that good in accuracy metrics, but has the best performance in the xNDKL@10 metric.

After re-ranking, the NDCG@10 and the xNDKL@10 metrics decrease, showing us that there is a trade-off between accuracy and fairness. In this results, the proposed models CDR(UJ) and NeuMF(UJ) are the models with the best performance in the majority of the metrics. Between the two re-ranking algorithms the proposed xGF improves more the fairness of the basic models while decreasing the accuracy.

The results in the Kendallτ@10 metric show us that the xGF methodically removes users who had similar suggestions, while bringing together users who did not have the same suggestions before the re-ranking. So similar users are treated the same, the algorithm DetGreedy does not do this methodically so it does not respect the rules of individual fairness.

Observing the above results we see that the performance of CDR(UJ) and NeuMF(UJ) models, in accuracy metrics, is relatively close. But, we decide to continue improving the CDR(UJ) model in the continuation of the work, because it is ranked first. The results in the MMMF(UJ) model are expected, due to the method is designed to work with sparse explicit data, i.e. it assume that there are many missing values and to have much less pairs than in an implicit setting.

4.5 Comparison of CDR variants

In this section we will present the results of the experiments on the CDR variants models CDR(UJ), CDR(UJ+JF), CADR(UJ+JF) and CVDR(UJ+JF), separately for each dataset and compare the results.

All models use user-job interaction information (UJ) and job features (JF), expect the basic model CDR(UJ) which uses only user-job interaction information, as mentioned above. Here we repeat the presentation of the results of the CDR(UJ) model for comparative purposes.

We used the application of the DetGreedy and xGF algorithms to improve the fairness of the CDR variants regarding features of the job seeker (user features: *discipline_id* and *region* of the XING dataset, *MajTopic* and *State* of the CAREER BUILDER dataset).

In each dataset we conducted the experiments to measure the fairness of algorithms in recommending jobs, which have the same selected attributes states, such as an individual user, with the probability 3/4, and the jobs, with different states of selected attributes, with probability 1/4. For measuring the the distance of the job recommendations lists of two users with equal qualifications, because of low computing power we measure in 3 runs the distance of 20 random users.

4.5.1 CDR variants on XING dataset

In the Table 4.12 and 4.13 we see the results of the experiments on the XING dataset before and after the application of DetGreedy and xGF algorithms, respectively. In the Figure 4.5 we see the performance of the algorithms in ROC Curve and in the Figure 4.6 their performance in PR Curve.

Table 4.12: Results of CDR variants on XING dataset before re-ranking

	Recall@20	NDCG@10	xNDKL@10	Kendallτ@10
CDR(UJ)	0.3953	0.2259	1.12	[0.22, 0.15, 0.26]
CADR(UJ+JF)	0.3965	0.2259	1.13	[0.21, 0.12, 0.26]
CDR(UJ+JF)	0.3955	0.2283	1.12	[0.19, 0.12, 0.28]
CVDR(UJ+JF)	**0.4082**	**0.2454**	**1.09**	[0.19, 0.14, 0.19]

Table 4.13: Results of CDR variants on XING dataset after re-ranking

DetGreedy				
	Recall@20	NDCG@10	xNDKL@10	Kendallτ@10
CDR(UJ)	0.3953	0.2169	**0.60**	[0.31, 0.09, 0.14]
CADR(UJ+JF)	0.3965	0.2169	0.61	[0.31, 0.09, 0.15]
CDR(UJ+JF)	0.3955	0.2192	0.61	[0.30, 0.09, 0.14]
CVDR(UJ+JF)	**0.4082**	**0.2380**	**0.60**	[0.21, 0.11, 0.14]
xGF				
	Recall@20	NDCG@10	xNDKL@10	Kendallτ@10
CDR(UJ)	0.3953	0.1400	**0.23**	[0.15, 0.09, 0.17]
CADR(UJ+JF)	0.3965	0.1378	**0.23**	[0.18, 0.09, 0.17]
CDR(UJ+JF)	0.3955	0.1415	**0.23**	[0.18, 0.09, 0.14]
CVDR(UJ+JF)	**0.4082**	**0.1546**	**0.23**	[0.23, 0.11, 0.11]

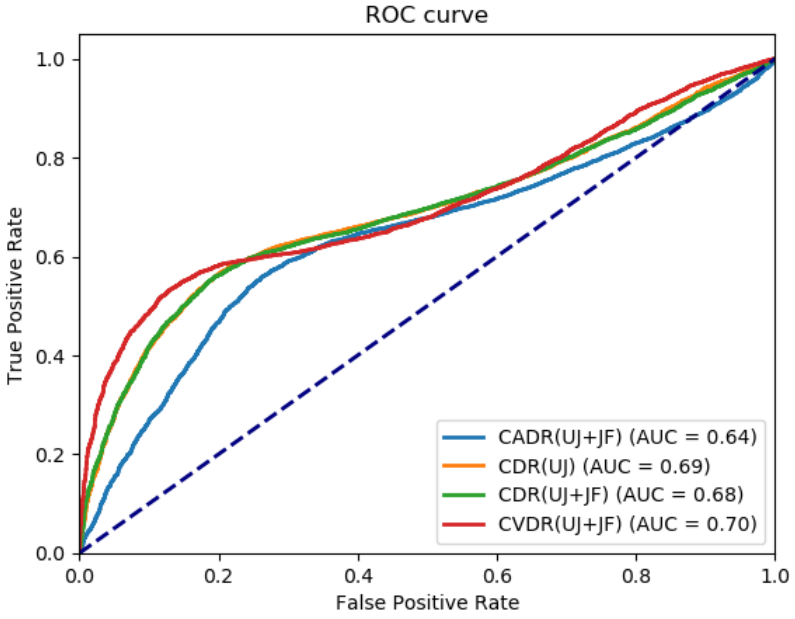

Figure 4.5: ROC Curve of CDR variants on XING dataset.

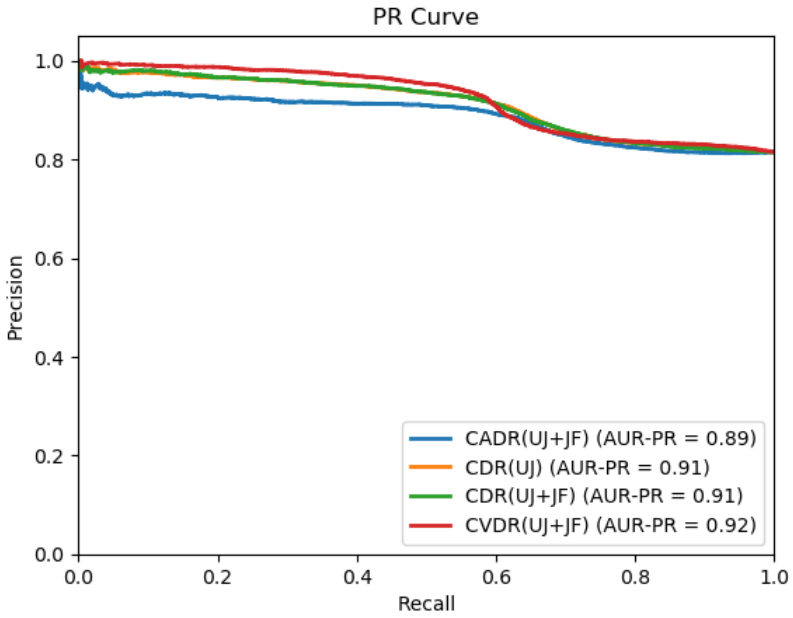

Figure 4.6: PR Curve of CDR variants on XING dataset.

Before applying the re-ranking algorithms, from the above results we see that the CVDR(UJ+JF) model has the best performance in the Recall@20 metric and NDCG@10 metrics, the best performance in the xNDKL@10 metric, but also the best performance in the ROC Curve and finally the best performance in the PR Curve.

The existence of job features improves the performance of the algorithms that use them. The CVDR(UJ+JF) algorithm uses VAE. The use of VAE exploit better the item space than SDAE in CDR(UJ+JF) and SAE in CADR(UJ+JF) algorithms, the metrics Recall@20 and NDCG@10 increase by $+0.0127$ and $+0.0171$, respectively and and the xNDKL@10 metric decrease -0.03, comparing the CVDR(UJ+JF) and CDR(UJ+JF), the second model in ranking performance .

After applying the DetGreedy and xGF re-ranking algorithms, the NDCG@10 and the xNDKL@10 metrics decrease, showing us that there is a trade-off between accuracy and fairness. In this results, the proposed model CVDR(UJ+JF) is the model with the best performance. Between the two re-ranking algorithms the proposed xGF improves more the fairness of the models while decreasing the accuracy.

The results in the Kendallτ@10 metric show us that the xGF methodically removes users who had similar suggestions, while bringing together users who did not have the same suggestions before the re-ranking. So similar users are treated the same, the algorithm DetGreedy does not do this methodically so it does not respect the rules of individual fairness.

4.5.2 CDR variants on CAREER BUILDER dataset

In the Table 4.14 and 4.15 we see the results of the experiments on the CAREER BUILDER dataset before and after the application of DetGreedy and xGF algorithms, respectively. In the Figure 4.7 we see the performance of the algorithms in ROC Curve and in the Figure 4.8 their performance in PR Curve.

Table 4.14: Results of CDR variants on CAREER BUILDER dataset before re-ranking

	Recall@20	NDCG@10	xNDKL@10	Kendallτ@10
CDR(UJ)	**0.0337**	0.0068	0.97	[0.22, 0.19, 0.05]
CADR(UJ+JF)	0.0285	0.0064	0.98	[0.08, 0.14, 0.03]
CDR(UJ+JF)	0.0288	0.0066	0.98	[0.17, 0.10, 0.06]
CVDR(UJ+JF)	0.0326	**0.0126**	**0.94**	[0.22, 0.12, 0.0]

Table 4.15: Results of CDR variants on CAREER BUILDER dataset after re-ranking

DetGreedy				
	Recall@20	NDCG@10	xNDKL@10	Kendallτ@10
CDR(UJ)	**0.0337**	0.0065	**0.48**	[0.40, 0.12, 0.0]
CADR(UJ+JF)	0.0285	0.0061	**0.48**	[0.31, 0.14, 0.10]
CDR(UJ+JF)	0.0288	0.0063	**0.48**	[0.29, 0.0, 0.0]
CVDR(UJ+JF)	0.0326	**0.0121**	**0.48**	[0.45, 0.0, 0.09]
xGF				
	Recall@20	NDCG@10	xNDKL@10	Kendallτ@10
CDR(UJ)	**0.0337**	0.0048	**0.25**	[0.17, 0.10, 0.12]
CADR(UJ+JF)	0.0285	0.0045	**0.25**	[0.07, 0.11, 0.04]
CDR(UJ+JF)	0.0288	0.0046	**0.25**	[0.15, 0.06, 0.17]
CVDR(UJ+JF)	0.0326	**0.0091**	**0.25**	[0.18, 0.08, 0.05]

Before applying the re-ranking algorithms, from the above results we see that the CVDR(UJ+JF) model has the second best performance in the Recall@20 metric, the best performance in NDCG@10 metric and the best performance in xNDKL@10 metric, also the worst performance in the ROC Curve and in the PR Curve.

The existence of job features does not improve the performance of all the algorithms that use them in accuracy metrics, showing us that the item space is difficult to explore in this dataset. The CVDR(UJ+JF) algorithm uses VAE. The use of VAE exploit better the item space than SDAE in CDR(UJ+JF) and SAE in CADR(UJ+JF) algorithms, the metric NDCG@10 increase by $+0.0058$ and the xNDKL@10 metric decrease -0.03, comparing the CVDR(UJ+JF) and CDR(UJ), the second model in ranking performance.

After applying the DetGreedy and xGF algorithms re-ranking algorithms, the NDCG@10 and the xNDKL@10 metrics decrease, showing us that there is a trade-off between accuracy and fairness. In this results, the proposed model CVDR(UJ+JF) is the model with the best performance in the majority of the metrics along with the model CDR(UJ). Between the two re-ranking algorithms the proposed xGF improves more the fairness of the models while

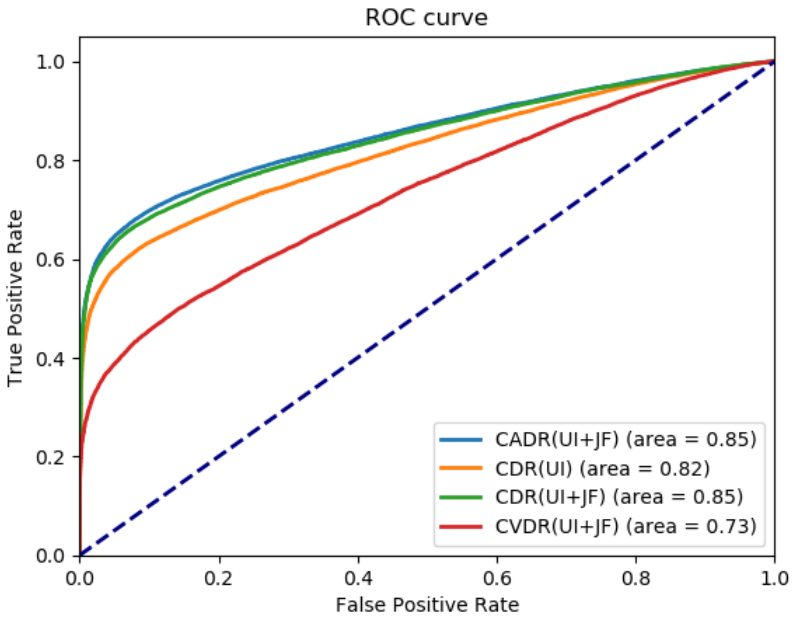

Figure 4.7: ROC Curve of CDR variants on CAREER BUILDER dataset.

decreasing the accuracy.

The results in the Kendallτ@10 metric show us that the xGF methodically removes users who had similar suggestions, while bringing together users who did not have the same suggestions before the re-ranking. So similar users are treated the same, the algorithm DetGreedy does not do this methodically so it does not respect the rules of individual fairness.

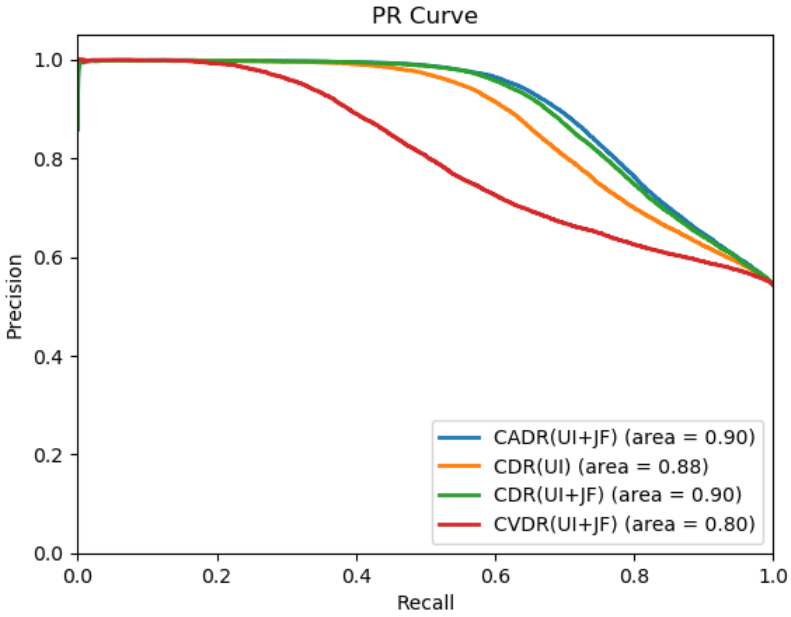

Figure 4.8: PR Curve of CDR variants on CAREER BUILDER dataset.

Part II

Group Fairness in Job Recommendations

Chapter 5

Proposed Methodology

In this chapter we present the problem we are trying to solve through a toy example.

In recommender systems, the term ***group fairness*** refers to the elimination of any kind of *discrimination or bias* that may be reflected in the list of recommended items against *protected groups* (e.g. *persons with disabilities*). As far as the *group fairness* is concerned, recommendation systems reproduce social biases, because of the fact that prediction models are trained using biased data. For example, in a job recommender engine, a female user interested in working in the military would expect to be offered army jobs, reflecting the actual distribution in the general population of men versus women (about 50%-50%). If a recommendation engine therefore recommends a disproportionately small number of army jobs to female users, it can be perceived as *biased*. Such a *biased* job recommendation system potentially influences users' perceptions and beliefs about jobs. Therefore, to eliminate the above *bias* and make a recommendation system *fair*, there will often need to be a *trade off* between the *accuracy* and the *unbiaseness* of the recommendation system, as will be shown in the following experiments.

5.1 Defining a Fairness Metric for Measuring the Bias in Recommender Systems

To understand the problem of predicting adequate candidates to a specific job position with fairness and for the presentation of our proposed algorithm, we present the following example.

Let us assume that, we have the following list of unemployed people from which a recommender system suggest the top-4 candidates (Table 5.1) for a specific position in the military, according to the exam grade of the candidates.

Table 5.1: Top-4 results of recommender system.

	Unemployed People	With Exams Score
1	**John 25 years old**	**9.0**
2	**Jack 25 years old**	**8.5**
3	**Bob 45 years old**	**8.0**
4	**Maria 35 years old**	**7.5**
5	Liam 35 years old	7.5
6	Elena 25 years old	7.0
7	Sofia 45 years old	6.5
8	Luna 35 years old	6.5

From the above results, therefore, it is easy to see that the predictions of the recommendation system are not unbiased and fair. The reason is that the percentage of women (1/4) in correct predictions is extremely disproportionate to the percentage of men (3/4) in Table 5.1, also the percentage of group age [20-30) (2/4) in correct predictions is extremely disproportionate to the percentages of the other group ages in this table and thus the recommendation system is biased. On the contrary, based on the Equal Opportunities principle, the recommender system should guarantee that the different types of classification errors (e.g. false negatives, false positives) should be equal (1/2) in terms of both sexes and (1/3) in terms of group ages.

In [20] the authors proposed the fairness metrics, **Skew**, **MinSkew**, **MaxSkew** and Normalized Discounted cumulative KL-divergence (**NDKL**), to compare theirs models. Their definition is shown below.

The **Skew** metric computes the extent to which the set of top k ranked results for a search or recommendation task differ over an attribute state with respect to the desired proportion of that attribute state and has the following mathematical formula:

$$Skew_{s_i}@k\left(\tau_r\right) = \log_e\left(\frac{p_{\tau_r^k,r,s_i}}{p_{q,r,s_i}}\right) \tag{5.1}$$

where τ_r is a ranked list of candidates for a search request r, s_i is an attribute

state of an attribute a, P_{q,r,s_i} is the desired proportion of candidates with attribute state s_i and P_{τ_r,r,s_i} is proportion of candidates in τ_r with state s_i.

$Skew_{s_i}@k$ is the (logarithmic) ratio of the proportion of candidates having the attribute state s_i among the top k ranked results to the corresponding desired proportion for s_i. A negative $Skew_{s_i}@k$ corresponds to a lesser than desired representation of candidates with state s_i in the top k results, while a positive $Skew_{s_i}@k$ corresponds to favoring such candidates.

In the previous example the desired proporsion of women candidates is $P_{q,r,s_i} = 1/2$ and the proportion of candidates, in the results of the algorithm, is $P_{\tau_r,r,s_i} = 1/4$ and finally the **Skew** metric is:

$$Skew_{women}@4\,(\tau_r) = \log_e\left(\frac{1/4}{1/2}\right) = -0.69$$

which means that the specific position in the army is not offered to many women and so the algorithm is biased.

The **MinSkew** metric signifies the worst disadvantage in representation given to candidates with a specific attribute state and has the following mathematical formula (where S is the set of all desired attribute states):

$$\text{MinSkew@k }\ (\tau_r) = \min_{s_i \in S} Skew_{s_i}@k\,(\tau_r) \tag{5.2}$$

For a search request r, $MinSkew@k$ provides the minimum skew among all attribute states.

The **MaxSkew** metric signifies the largest unfair advantage provided to candidates with an attribute state and has the following mathematical formula:

$$\text{MaxSkew@k}(\ \ \tau_r) = \max_{s_i \in S} Skew_{s_i}@k\,(\tau_r) \tag{5.3}$$

For a search request r, $MaxSkew@k$ provides the maximum skew among all attribute states.

Since both $\sum p_{\tau_r^k,r,s_i} = 1$ and $\sum p_{q,r,s_i} = 1$, it follows that for any ranked list, and for any k, $MaxSkew@k \geq 0$ and $MinSkew@k \leq 0$.

In the previous example the desired proporsion of men candidates is $P_{q,r,s_i} = 1/2$ and the proportion of candidates, in the results of the algorithm, is $P_{\tau_r,r,s_i} = 3/4$ and finally the **Skew** metric is:

$$Skew_{men}@4\,(\tau_r) = \log_e\left(\frac{3/4}{1/2}\right) = 0.40$$

which means that the specific position in the army is offered to many men and so the algorithm is biased.

The **MinSkew** metric is:

$$\text{MinSkew@4}(\ \tau_r) = \min_{s_i \in S}\{-0.69, 040\} = -0.69$$

The **MaxSkew** metric is:

$$\text{MaxSkew@4}(\ \tau_r) = \max_{s_i \in S}\{-0.69, 040\} = 0.40$$

The **NDKL** metric for the top-4 results of Table 5.1 for the gender attribute is:

$$\begin{aligned} \text{NDKL}\left(top4\right) &= \frac{1}{2.56}\sum_{i=1}^{4}\frac{1}{\log_2(i+1)}d_{KL}\left(D_{\tau_r^i}\|D_r\right), \\ \text{NDKL}\left(top4\right) &= \frac{1}{2.56}\left(0.69 + 0.44 + 0.35 + 0.05\right), \\ \text{NDKL}\left(top4\right) &= 0.6 \end{aligned}$$

The **NDKL** metric for top-4 results and age groups attribute is:

$$\begin{aligned} \text{NDKL}\left(top4\right) &= \frac{1}{2.56}\sum_{i=1}^{4}\frac{1}{\log_2(i+1)}d_{KL}\left(D_{\tau_r^i}\|D_r\right), \\ \text{NDKL}\left(top4\right) &= \frac{1}{2.56}\left(1.1 + 0.69 + 0.23 + 0.02\right), \\ \text{NDKL}\left(top4\right) &= 0.8 \end{aligned}$$

In the same way we can calculate and the other metrics for the attribute age groups. The **xNDKL** metric for this list and both attributes is:

$$\text{xNDKL}\left(top4\right) = 0.8 + 0.6 = 1.4$$

The **DetGreedy** fair re-ranking algorithm process the above list (Table 5.1) and suggest the following top-4 list of unemployed people (Table 5.2) for a specific job.

Table 5.2: Top-4 results of DetGreedy Fair Re-Ranking Algorithm.

	Unemployed People	With Exams Score
1	**John 25 years old**	**9.0**
2	**Bob 45 years old**	**8.0**
3	**Maria 35 years old**	**7.5**
4	**Elena 25 years old**	**7.0**
5	Liam 35 years old	7.5
6	Sofia 45 years old	6.5
7	Jack 25 years old	8.5
8	Luna 35 years old	6.5

The **NDKL** metric for this list and gender attribute is:

$$\begin{aligned} \mathrm{NDKL}\left(top4\right) &= \frac{1}{2.56}\sum_{i=1}^{4}\frac{1}{\log_2(i+1)}d_{KL}\left(D_{\tau_r^i}\|D_r\right), \\ \mathrm{NDKL}\left(top4\right) &= \frac{1}{2.56}\left(0.69+0.44+0.03+0\right), \\ \mathrm{NDKL}\left(top4\right) &= 0.45 \end{aligned}$$

The **NDKL** metric for top-4 results and age groups attribute is:

$$\begin{aligned} \mathrm{NDKL}\left(top4\right) &= \frac{1}{2.56}\sum_{i=1}^{4}\frac{1}{\log_2(i+1)}d_{KL}\left(D_{\tau_r^i}\|D_r\right), \\ \mathrm{NDKL}\left(top4\right) &= \frac{1}{2.56}\left(1.1+0.25+0+0.02\right), \\ \mathrm{NDKL}\left(top4\right) &= 0.53 \end{aligned}$$

The **xNDKL** metric for this list and both attributes is:

$$\mathrm{xNDKL}\left(top4\right) = 0.45 + 0.53 = 0.98$$

which means the algorithm is fairer because decreases the **xNDKL** metric.

The **DetGreedy** algorithm, however, manages a combined attribute (both gender and age groups) without taking into account the probabilities of appearance of the attributes of which it is composed. This can be

seen from the results of the algorithm in position two in which it proposes a man 45 years old, while it would be preferable to have a woman of any age. This problem motivated us to create the **xGF** algorithm.

The **xGF** fair re-ranking algorithm process the above list (Table 5.1) and suggest the following top-4 list of unemployed people (Table 5.3) for a specific position in the military.

Table 5.3: Top-4 results of xGF Fair Re-Ranking Algorithm.

	Unemployed People	With Exams Score
1	**John 25 years old**	**9.0**
2	**Maria 35 years old**	**7.5**
3	**Bob 45 years old**	**8.0**
4	**Elena 25 years old**	**7.0**
5	Liam 35 years old	7.5
6	Sofia 45 years old	6.5
7	Jack 25 years old	8.5
8	Luna 35 years old	6.5

The **NDKL** metric for this list and gender attribute is:

$$\begin{aligned} \mathrm{NDKL}\left(top4\right) &= \frac{1}{2.56}\sum_{i=1}^{4}\frac{1}{\log_2(i+1)}d_{KL}\left(D_{\tau_r^i}\|D_r\right), \\ \mathrm{NDKL}\left(top4\right) &= \frac{1}{2.56}\left(0.69+0+0.03+0\right), \\ \mathrm{NDKL}\left(top4\right) &= 0.28 \end{aligned}$$

The **NDKL** metric for top-4 results and age groups attribute is:

$$\begin{aligned} \mathrm{NDKL}\left(top4\right) &= \frac{1}{2.56}\sum_{i=1}^{4}\frac{1}{\log_2(i+1)}d_{KL}\left(D_{\tau_r^i}\|D_r\right), \\ \mathrm{NDKL}\left(top4\right) &= \frac{1}{2.56}\left(1.1+0.25+0+0.02\right), \\ \mathrm{NDKL}\left(top4\right) &= 0.53 \end{aligned}$$

The **xNDKL** metric for this list and both attributes is:

$$\text{xNDKL}\,(top4) = 0.28 + 0.53 = 0.81$$

which means the **xGF** algorithm is fairer than **DetGreedy** because the **xNDKL** metric is decreasing more, in more than one attribute setting.

We performed experiments on NeuMF model examining and trying to improve its fairness on selected attributes, using the proposed fairness metric and the **DetGreedy** fair re-ranking algorithm of work [20] and our proposed **xGF** fair re-ranking algorithm.

Chapter 6

Experimental Evaluation

In this chapter we will show the setups of the datasets that have been used. We will also compare the results of DetGreedy and xGF fair re-ranking algorithms. In detail, the performance of the algorithms is shown through evaluation metrics.

6.1 Dataset Setup

We used the same datasets as Part I of this thesis, but with different setups. In this section we show the percentage we used from each dataset.

In the XING dataset due to low computing power we only used the latest and most recent user interactions and a total of 26678 out of 8826678 records. Each experiment was conducted by splitting the recordings into 80% for the training set and 20% for the test set. In Table 6.1, we show the statistics of training and test set.

Table 6.1: The statistics of XING used data

	Training Data	Test Data
Users	2262	751
Items	15249	1739
Ratings	17370	1951

In the CAREER BUILDER dataset due to low computing power we only used the user interactions of jobs of job7 features window of dataset and a

total of 103667 out of 3206222 records. Each experiment was conducted by splitting the recordings into 80% for the training set and 20% for the test set. In Table 6.2, we show the statistics of training and test set.

Table 6.2: The statistics of CAREER BUILDER used data

	Training Data	Test Data
Users	10886	5321
Items	47936	9083
Ratings	82930	12511

6.2 Evaluation

In the following subsections we show the accuracy and fairness metrics, we used to compare the different models. Then we compare the NeuMF model with each other in each dataset in terms of their performance with the evaluation metrics, as well as graphical evaluation methods. Finally, we analyze the conclusions of each one.

6.2.1 Evaluation Metrics

In this subsection we describe the accuracy and fairness metrics, which we have used to compare the performance of the different models.

Accuracy Metrics

We use the aacuracy metric **NDCG**, to compare our models. Its dedfinition is shown previously in subsection 5.2.1.

Fairness Metrics

We use the fairness metric **xNDKL** to compare our models. Its definition of is shown previously in subsection 5.2.1.

6.3 Evaluation Strategy

As mentioned above, we divided the recordings of each dataset into 80% training and 20% test pieces. We conducted experiments on NeuMF model and re-ranking algorithms. The best parameters of NeuMF are the same as Part I and are shown in Appedix A.

6.4 Comparison of algorithms in terms of Fairness

In this section we will present the results of the experiments on fairness tests on NeuMF algorithm, using the fairness metrics and the re-ranking algorithms: DetGreedy, and xGF, separately for each dataset and compare the results.

6.4.1 Fair Re-Ranking on XING dataset

In the XING dataset we conducted the experiments to measure the fairness of algorithms in recommending jobs fairly and with the same probability to all users depending on their years of experience and the region in which they live.

We conducted experiments in one setup measuring the fairness of the algorithm if we want it to recommend jobs to users in conjunction with a probability of 1/2, creating 2 user groups (< 10 years, ≥ 10 years), based on years of experience attribute and with a probability of 1/3, creating 3 user groups (west regions of Germany, east regions of Germany, foreign regions), based in the region attribute.

In the Figures 6.1 and 6.2, we see the performane of NeuMF, DetGreedy and xGF algorithms in NDCG nad xNDKL metrics, respectively.

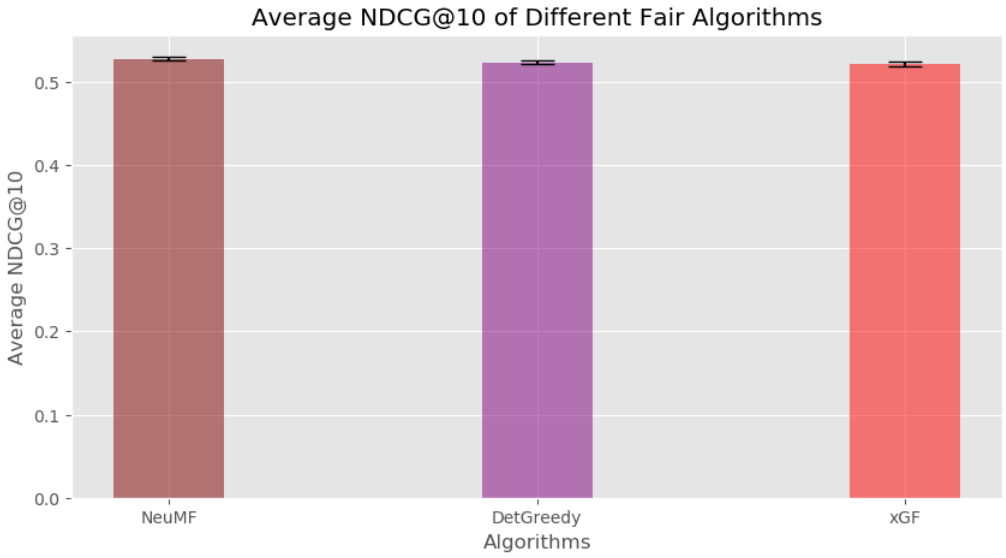

Figure 6.1: NDCG@10 of NeuMF and re-ranking algorithms on XING dataset.

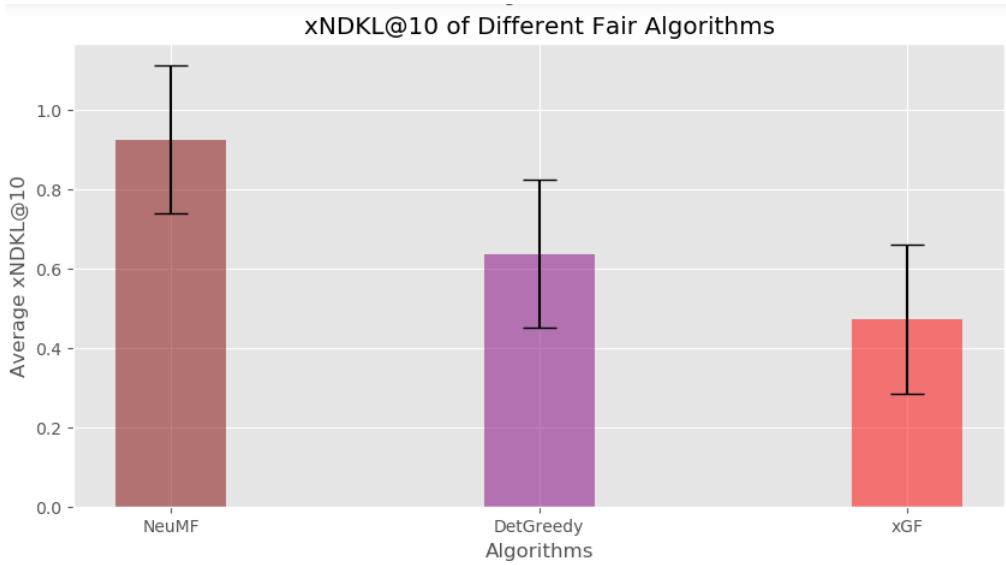

Figure 6.2: xNDKL@10 of NeuMF and re-ranking algorithms on XING dataset.

For the above figures we see that the use of proposed xGF re-ranking algorithm on results of NeuMF improves its performance in xNDKL metric more than the DetGreedy. Finally, the accuracy of the re-ranking algorithms (NDCG metric) is slightly smaller than NeuMF algorithm, showing us that there is a trade-off between accuracy and fairness.

6.4.2 Fair Re-Ranking on CAREER BUILDER dataset

In the CAREER BUILDER dataset we conducted the experiments to measure the fairness of algorithms in recommending jobs fairly and with the same probability to all users depending on their total years of experience and the region in which they live.

We conducted experiments in one setup measuring the fairness of the algorithms if we want it to recommend jobs to users in conjunction with a probability of 1/2, creating 2 user groups (< 10 years, ≥ 10 years), based on total years of experience attribute and with a probability of 1/3, creating 3 user groups (west states of America, east states of America, foreign regions), based in the state attribute.

In the Figures 6.3 and 6.4, we see the performane of NeuMF, DetGreedy and xGF algorithms in NDCG and xNDKL metrics, respectively.

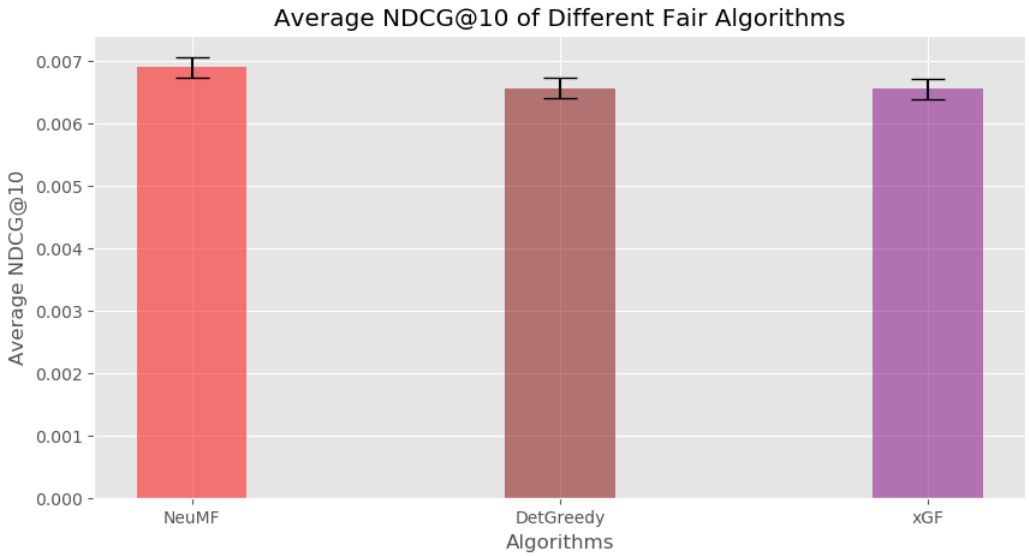

Figure 6.3: NDCG@10 of NeuMF and re-ranking algorithms on CAREER BUILDER dataset.

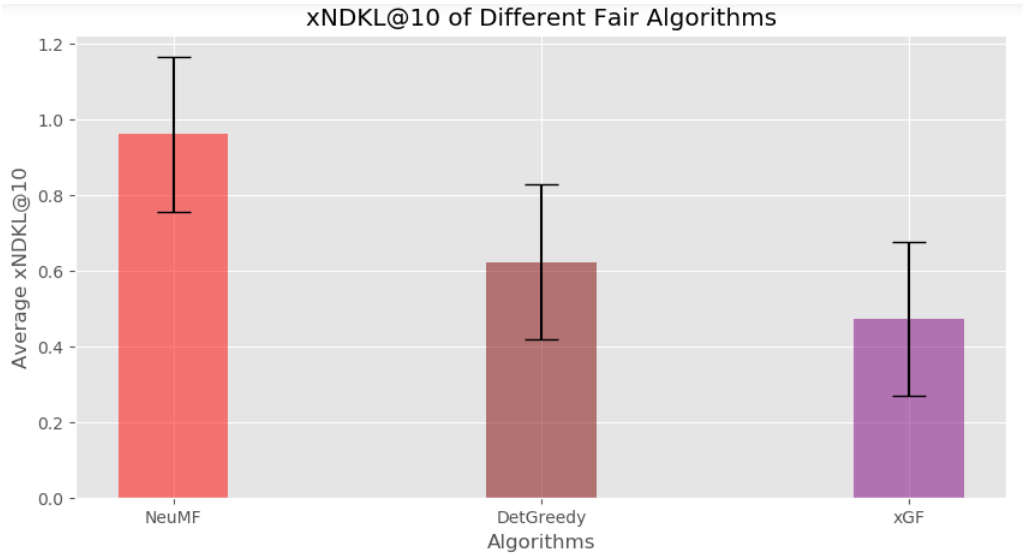

Figure 6.4: xNDKL@10 of NeuMF and re-ranking algorithms on CAREER BUILDER dataset.

For the above figures we see that the use of proposed xGF re-ranking algorithm on results of NeuMF improves its performance in xNDKL metric more than the DetGreedy. Finally, the accuracy of the re-ranking algorithms (NDCG metric) is smaller than NeuMF algorithm, showing us that there is a trade-off between accuracy and fairness.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

One purpose of this thesis was to study the previously proposed models for job recommendation and propose a suitable model, combining any pre-existing models, but also using as much as possible any additional information that exists, with the aim of increasing accurate recommendations. In this direction we compared various models of the Cornac library and selected the four most accurate ones, which are: MMMF, CDR, IBPR and NeuMF.

From the previous models the CDR algorithm gave us the best results in the accuracy metrics and we decided to extend it further. We created three variants of the CDR(UJ+JF) model in Cornac library, CADR(UJ+JF), CDR(UJ) and CVDR(UJ+JF), using two more types of autoendoders, VAE and SAE, and we showed the results.

The results from the previous variants showed us the superiority of the CVDR model in accuracy metrics across the two real world datasets.

The CVDR model employs VAE to extract deep feature representation from side information of jobs and then integrates into pair-wise ranking model. This action mitigates the sparsity of interactions between users and jobs.

The results also show us that VAE can encode job features better than SAE and SDAE autoencoders. They thus achieve the improvement of the model in the accuracy metrics. However we see that the improvement in the CAREER BUILDER dataset is not great, showing us that in a more complex item space the VAE's do not achieve good results. The same conclusion is

drawn from report [2].

Additionally, the other purpose of this thesis was to study the previously proposed models for fair job recommendation. In this direction we conducted experiments in two directions from the concepts of individual fairness in Part I and group fairness in Part II. On Part I we studied the fairness on all previous mentioned models and tried to improve it using the pre-existing re-ranking algorithm DetGreedy. All pre-existing re-ranking algorithms try to improve the fairness based on a protected attribute. So we created an improved version of DetGreedy. The xGF takes into account more than one protected attribute and its results relative to DetGreedy are more fair and respects the rules of individual fairness. On Part II we studied the fairness in the NeuMF model and tried to improve it using the pre-existing re-ranking algorithm DetGreedy and the proposed re-ranking algorithm xGF, which also showed us its superiority in group fairness setting.

The thesis was limited to the possibilities of the hardware and software that existed for it its development. With a faster processor and different software it will they extended to the use of entire datasets and to present different metric results, which better show us the trend of the algorithms. Specifically, we ran the experiments on Anacoda 2.4.0 environment, with Jupyter Notebook 6.4.12 on a personal Laptop with an Intel i3 processor and memory 4 GB RAM, while the Windows 10 operating system was used.

7.2 Future Work

A future extension of the present thesis is to conduct experiments on the entire datasets using a computer with sufficient computing power. Still conducting experiments on other real world datasets. Thus we will see in more detail the trend of the proposed models.

The use of entire datasets or even their possible pre-processing, using methods of known reports, will make our proposed models comparable with pre-existing results and will give the appropriate motivation for the further improvement of our models.

Another possible extension, in the study of accuracy, is to conduct experiments on other types of autoencoders and different architectures of existing used types. In ours experiments the use of different autoencoder architectures do not improve the results of the used algorithms.

This thesis examines the performance of algorithms in accuracy, beyond-

accuracy metrics, in the experiments we conducted, the use and presentation of different metrics did not make sense, but it may make sense in conducting different experiments.

Possible extension in the study of the fairness is to examine and the use of other fairness methods and metrics.

We would still like to do an online evaluation of the proposed models and further to be able to see the satisfaction of the users, but also how many of the model's proposals finally became recruitments.

Bibliography

[1] S. Waki, T. Suzumura, H. Kanezashi, and S. Kobayashi, Matching Maximization in Job Recommendation System with Reinforcement Learning. 2018.

[2] E. Lacic, M. Reiter-Haas, D. Kowald, M. R. Dareddy, J. Cho, and E. Lex, Using autoencoders for session-based job recommendations. Springer, 2020.

[3] P. Yi, C. Yang, C. Li, and Y. Zhang, A Job Recommendation Method Optimized by Position Descriptions and Resume Information. IEEE Xplore, 2016.

[4] J. Valverde-Rebaza, R. Puma, P. Bustios, and N. C. Silva, Job Recommendation based on Job Seeker Skills: An Empirical Study. Grenoble, France: Text2StoryIR'18 Workshop, 2018.

[5] S. A. Alsaif, M. S. Hidri, H. A. Eleraky, I. Ferjani, and R. Amami, Learning-Based Matched Representation System for Job Recommendation. Computers, 2022.

[6] I. Paparrizos, B. B. Cambazoglu, and A. Gionis, Machine Learned Job Recommendation. Chicago, Illinois, USA.: RecSys'11, 2011.

[7] A. Nigam, A. Roy, H. Singh, and H. Waila, Job Recommendation through Progression of Job Selection. IEEE Xplore, 2019.

[8] W. Chen, X. Zhang, H. Wang, and H. Xu, Hybrid Deep Collaborative Filtering for Job Recommendation. IEEE Xplore, 2017.

[9] K. Appadoo, M. B. Soonnoo, and Z. Mungloo-Dilmohamud, JobFit: Job Recommendation using Machine Learning and Recommendation Engine. IEEE Xplore, 2020.

[10] W. Hong, S. Zheng, H. Wang, and J. Shi, A Job Recommender System Based on User Clustering. Journal of Computers, 2013.

[11] A. Rivas, P. Chamoso, A. González-Briones, R. Casado-Vara, and J. M. Corchado, Hybrid job offer recommender system in a social network. Wiley Online Library, 2019.

[12] A. Mughaid, I. Obeidat, B. Hawashin, S. AlZu'bi, and D. Aqel, A smart Geo-Location Job Recommender System Based on Social Media Posts. IEEE Xplore, 2019.

[13] G. Ozcan and S. G. Oguducu, Applying Different Classification Techniques in Reciprocal Job Recommender System for Considering Job Candidate Preferences. IEEE Xplore, 2016.

[14] V. S. Dave, B. Zhang, M. Al Hasan, K. AlJadda, and M. Korayem, A Combined Representation Learning Approach for Better Job and Skill Recommendation. Torino, Italy: CIKM '18, 2018.

[15] J. Zhao, Embedding-based Recommender System for Job to Candidate Matching on Scale. Singapore: KDD'21 IRS Workshop, 2021.

[16] H. Ying, L. Chen, Y. Xiong, and J. Wu, Collaborative Deep Ranking: a Hybrid Pair-wise Recommendation Algorithm with Implicit Feedback. Springer, 2016.

[17] Y. Mashayekhi, B. Kang, J. Lijffijt, and T. De Bie, ReCon: Reducing Congestion in Job Recommendation using Optimal Transport. Singapore, Singapore: RecSys '23, 2023.

[18] Y. Li, M. Yamashita, H. Chen, D. Lee, and Y. Zhang, Fairness in Job Recommendation under Quantity Constraints. 2023.

[19] C. Rus, J. Luppes, H. Oosterhuis, and G. H. Schoenmacker, Closing the Gender Wage Gap: Adversarial Fairness in Job Recommendation. Seattle, USA: RecSys in HR'22, 2022.

[20] S. C. Geyik, S. Ambler, and K. Kenthapadi, Fairness-Aware Ranking in Search & Recommendation Systems with Application to LinkedIn Talent Search. ACM KDD, 2019.

[21] R. Fagin, R. Kumar and D. Sivakumar, Comparing top k lists. ACM-SIAM (SODA), 2003.

List of Abbreviations

AE Autoencoder. 32

AUC Area Under Curve. 23, 26

BLSTM-A Bidirectional Long Short Term Memory Network with Attention. 20

CADR Collaborative Deep Ranking using stacked Autoencoders. 36, 60, 62, 64, 82

CBF Context-Based Filtering. 20

CBR Content-Based Reasoning. 22

CCRS Classification-Candidate Reciprocal Recommendation. 23

CDL Collaborative Deep Learning. 20

CDR Collaborative Deep Ranking. 9, 33–37, 53, 56, 59, 60, 62, 64, 82

CTR Click-Through-Rate. 24

CVDR Collaborative Deep Ranking using Variational autoencoder. 9, 36–38, 43, 49, 60, 62, 64, 82, 91

DetGreedy Deterministic Greedy algorithm. 9, 10, 38, 39, 41, 53, 54, 56, 57, 59, 60, 62–65, 71, 72, 74, 75, 77, 79–81, 83

DTNB Decision Table/Naıve Bayes hybrid classifier. 19

IBPR Indexable Bayesian Personalized Rank- ing. 53, 56, 59, 82

JF Job Features. 60, 62, 64, 82

Joint-BPR Joint-Bayesian Personalized Ranking. 23

MAP Mean Averaged Precision. 23

MLP Multi-Layer Perseptron. 34

MMMF Maximum Margin Matrix Factorization. 53, 56, 59, 82

MRR Mean Reciprocal Rank. 21

NDCG Normalized Discounted Cumulative Gain. 21, 24, 49–51, 76, 77, 79–81

NDKL Normalized Discounted cumulative KL-divergence. 52, 69, 71–73

NeuMF Neural Matrix Factorization. 10, 53, 56, 59, 74, 76, 77, 79–83

NLP Natural Language Processing. 20, 23

PCA Principal Component Analysis. 32

PMF Propablistic Matrix Factorization. 20

PR Precision-Recall. 54, 56, 57, 59, 60, 62–64

ROC Receiver Operating Characteristic. 54, 56, 57, 59, 60, 62–64

SAE Stacked Autoencoder. 32, 33, 36, 62, 64, 82

SDAE Stacked Denoising Autoencoder. 32, 33, 35, 36, 62, 64, 82

SVM Support Vector Machine. 23

TF-IDF Term Frequency-Inverse Document Frequency. 18

TPR True Positive Rate. 26

UJ User-Job interactions. 53, 56, 59, 60, 62, 64, 82

VAE Variational Autoencoder. 32, 33, 36, 37, 62, 64, 82

xGF eXpanded Group Fairness algorithm. 10, 32, 39, 41, 43, 53, 54, 56, 57, 59, 60, 62–65, 73–75, 77, 79–81, 83

xNDKL eXpanded Normalized Discounted cumulative KL-divergence. 52, 71–74, 76, 77, 79–81

Appendix A

Parameters of used models

In this appendix we show the best parameters of used models in each dataset.

The best parameters in the XING dataset for the basic models are shown in the Table A.1, while the best parameters in the CAREER BUIDER dataset are shown in the Table A.2.

Regarding the proposed model CVDR and its variants we also performed various experiments and the best parameters, in each dataset, are shown in the Tables A.3 (XING), A.4 (CAREER BUILDER).

Table A.1: Parameters for the basic models in XING dataset

	CDR(UJ)	NeuMF(UJ)	MMMF(UJ)	IBPR(UJ)
k	40	-	10	10
max_iter	200	-	200	-
learning_rate	0.001	0.001	0.01	-
lambda_u	0.01	-	-	-
lambda_v	0.1	-	-	-
num_factors	-	10	-	-
layers	-	[64,32,16,8]	-	-
act_fn	-	tanh	-	-
learner	-	adam	-	-
num_epochs	-	60	-	-
num_neg	-	40	-	-
batch_size	128	256	-	-
seed	123	123	-	-

Table A.2: Parameters for the basic models in CAREER BUILDER dataset

	CDR(UJ)	NeuMF(UJ)	MMMF(UJ)	IBPR(UJ)
k	100	-	10	10
max_iter	4	-	200	-
learning_rate	0.001	0.001	0.01	-
lambda_u	0.01	-	-	-
lambda_v	0.1	-	-	-
num_factors	-	6	-	-
layers	-	[64,32,16,8]	-	-
act_fn	-	tanh	-	-
learner	-	adam	-	-
num_epochs	-	20	-	-
num_neg	-	40	-	-
batch_size	128	256	-	-
seed	123	123	-	-

Table A.3: Parameters for the CDR variants in XING dataset

	CDR(UJ)	CADR(UJ+JF)	CDR(UJ+JF)	CVDR(UJ+JF)
k	40	40	40	200
max_iter	200	200	200	30
learning_rate	0.001	0.001	0.001	0.001
lambda_u	0.01	0.01	0.01	0.01
lambda_v	0.1	0.1	0.1	0.1
lambda_w	-	0.0001	0.0001	0.0001
lambda_n	-	5	5	0.01
autoencoder_structure	-	[150]	[150]	-
vae_layers	-	-	-	[350,250]
act_fn	-	-	-	sigmoid
batch_size	128	128	128	128
seed	123	123	123	123

Table A.4: Parameters for the CDR variants in CAREER BUILDER dataset

	CDR(UJ)	CADR(UJ+JF)	CDR(UJ+JF)	CVDR(UJ+JF)
k	100	100	100	25
max_iter	4	6	6	30
learning_rate	0.001	0.001	0.001	0.001
lambda_u	0.01	0.01	0.01	0.01
lambda_v	0.1	0.1	0.1	0.1
lambda_w	-	0.0001	0.0001	0.001
lambda_n	-	5	5	0.01
autoencoder_structure	-	[250]	[250]	-
vae_layers	-	-	-	[200,100]
act_fn	-	-	-	sigmoid
batch_size	128	128	128	128
seed	123	123	123	123