Erasmus University Rotterdam
Erasmus School of Economics
Master Thesis Data Science & Marketing Analytics

Enhancing Job Matching: Leveraging State-of-Art Recommendation Systems To Handle Data Sparsity At Magnet.me

Eleftherios Tranakos (617019lt)

Thesis Supervisor: Kathrin Gruber
Second Assessor: Bas Donkers
Company Mentor: Alex Walterbos
Date: 31st October 2024

Acknowledgment

I would like to express my gratitude to Magnet.me Company for providing me with the opportunity to undertake this thesis internship. The research project was both challenging and interesting, offering me a broad range of experience and knowledge that has significantly contributed to my academic and professional development. I am also deeply thankful to my parents and my sister for their ongoing support and encouragement throughout this academic journey.

Finally, I want to specially thanks Alex Walterbos from Magnet.me, whose guidance and mentorship have been more than inspiring. His insights and support throughout the whole thesis internship were essential in helping me complete this research. I dedicate this work to him in recognition of the invaluable experience and knowledge I gained through his mentorship.

Abstract

This thesis investigates Bayesian Personalized Ranking (BPR) as a solution to enhance personalized job recommendations at the Magnet.me Company, alleviating the issue of data sparsity. By Utilizing the BPR model with its pairwise ranking approach, it outperforms the baseline model of Magnet.me across key metrics, such as Recall, Precision, NDCG, and MRR. The results of BPR model indicate that it can be applied to other recommendations domains struggling with data sparsity difficulties, beyond its original implementation. Despite limited tuning due to time and computational constraints, BPR exhibits in offline testing that it can improve job recommendations at Magnet.me, proving its suitability for the recruitment sector. Future work could include the evaluation of BPR performance in a live production environment for more representative results and the combination of it with a content-based approach for more accurate job recommendations.

Keywords

Jobs, recruitment platform, Magnet.me, job recommendations, recommendation systems, personalized recommendations, Bayesian Personalized Ranking, BPR, user engagement, data sparsity, pairwise ranking.

Contents

Chapter 1

Introduction

The massive expansion and diversity of information on the Web and the advent of new e-commerce services provide to users with too many of options leading them often to make hasty decisions(e.g., choose Product A over Product B), which are frequently not the most suitable for their needs. As was referred to Ricci et al. (2010), the regular presence of e-commerce sites on Web and the rapid evolution of them, drove the necessity of automate recommendations of products and services to users by sorting all the available options and providing them the most relevant.

1.1 Recommendation Systems

The advent of Recommendation Systems (RSs) represents an important advancement in data-driven technology, recommending relevant options to people. The RSs is defined as a combination of software tools and techniques aims to not only increasing personalised recommendations and improving customer satisfaction but also determining business data-driven strategies as noted by Das et al. (2017). RSs are applicable in a great variety of domains such as e-commerce sites (e.g.,, Ebay and Netflix) recommending personalised products or movies, respectively , social media (e.g., Instagram) suggesting friends to users that possibly know or recruitment platforms(e.g., CareerBuilder.com) for job recommendations, etc.

It is worth mentioning that, despite the fact that each domain serves a different goal, RSs across these domains work in similar ways, using algorithms to provide options based on user interests. For example, in the recruitment industry, a part of RSs match jobs to job seekers based on skills and experience through content analysis focusing on common key words between the two sides. On the other hand, for movies or products, some suggestions are offered based on past viewing, interactions or purchasing history. However, in both cases, the objective is to personalize recommendations, even though the business target and the types of data used differ sometimes.

1.2 Feedback Collection

In the context of RSs, a general term "item" is used to represent an "option" (e.g., movie, product, friends), which is provided to users as a prediction of what they will potentially prefer based on their preferences as stated by Ricci et al. (2010). The preferences, are captured through the feedback collection which assist RSs to propose relevant recommendations to users by selecting items depending on whether this matches with the interest of a user. The feedback methods are categorized into three parts:

1. Implicit Feedback which is gathered without the awareness of the users as it is obtained through their actions during the navigation process. Such actions can be browsing history, search trend, clicks and consumption of time on the web.

2. Explicit Feedback in which the user has the major role for rating and assigning values to items for evaluation purposes. Allowing users rating items can be seen as a way of interacting with the item based on their preferences. The most well know types of this feedback are like or dislike for interested or not interested items (e.g., interaction), ratings (e.g, scale 1-5) or text and reviews.

3. Hybrid Feedback combines both of the above mentioned techniques, numeric scores and navigation behavior of the user, in order to forecast relevant items to the users.

1.3 Information about Magnet.me

This thesis will focus on improving personalised job recommendations at Magnet.me Company by developing a new recommendation system that handles sparsity and comparing it with benchmark model of the the company that is currently used. Magnet.me is the first career network for students,graduates and employers in the Netherlands since 2012. The company started this adventurous journey after discovering that the process of finding an internship or job for students was complicated and time-consuming.

Today, twelve years later, in 2024, more than three hundred thousand students take advantage of the Magnet.me network to initiate their careers and over five thousand companies make use of the company service to connect with young professionals and successfully recruit users for internships and jobs. The goal of Magnet.me for efficient job matching, for both students and recruiters, is to "Make it easier to find a job you love or a talent you need". That goal is achieved by utilizing Machine Learning (ML) techniques to enable students to connect with potential employers and find jobs that match their profile and preferences. More details will be given on the Methodology chapter where the benchmark model of our research is explained thoroughly.

1.4 Major Challenges of RSs

One important concern of RSs is data sparsity, particularly within the recruitment industry as indicated by Shalaby et al. (2017). This problem grows when users interact with a minor portion of all available job (or in general 'items'), making it challenging for algorithms to accurately predict preferences and deliver relevant recommendations. Connecting this with Magnet.me platform for job matching, a user cannot interact with one hundred seventy four thousand live jobs. Thus, it is understandable that the proportion of a user rating a job is disproportional in contrast to available jobs and the recommendations of RSs will not be that relevant due to this. Data sparsity can be seen as missing values in the rating matrix.

In addition, a second paramount difficulty is the cold start problem, stated by Panteli & Boutsinas (2023), originating either from new users who have not been engaged with the items in the past or by new items for which the user has not interacted with it. This contributes to missing info or in mathematical terms cause the data matrix to be sparse since there is no past interactions between user-item. With the term "sparse" is described the fact that within data we have a lot of zero values on the user - items interaction showing that there was not an engagement yet.

Lastly, the third challenge that RSs face (especially CF approaches) is scalability as reported by Roy & Dutta (2022). Being known, RSs from its nature, need massive amount of data to produce results. As a result, the extreme amount of data and the velocity that the data is integrated into these systems can cause increased computational load and memory usage, leading to the scalability issue. In fact, in the current epoch, every day new users and new items are introduced automatically to recommendation systems leading the problem to be more intense than ever before.

1.5 Main Objectives and Research Question

Having seen the core challenges that RSs face, this thesis aims at developing a recommendation system (e.g., model) for Magnet.me to increase personalised job recommendations for job seekers by handling sparsity . Thus the research question that is established and this thesis aims to address is the following:

How can state-of-the-art recommendation systems improve personalized job recommendations at Magnet.me by handling the data sparsity problem?

1.6 Significance of the study

Having been mentioned earlier in this research, RSs is a wide-spread tool across many domains such as streaming services and social media. However, in the modern recruitment

industry, job recommendations is less explored as indicated by Shalaby et al. (2017). This occurs because a great extent of existing recommendation systems take advantage of semantic analysis (i.e. content-based filtering) for resumes and job descriptions, which fail to capture hidden trends of user behaviour within the data. On the contrary, collaborative filtering approach tend to uncover these hidden patterns. Both mentioned approaches, are described in detail in the literature part. Thus, this research is paramount as will exhibit how Bayesian Personalized Ranking (BPR), a collaborative-filtering approach, can enhance job matching in the recruitment industry proving that this model can be applied for job recommendations apart from movies.

Furthermore, as was observed by Shalaby et al. (2017), the increasing amount of jobs through online platforms setting to millions of job seekers a challenging task to find the most relevant jobs to them. The above process, in the majority of times can not only be time consuming but also tiring for the users as the latter need to devote quite enough time in order to read job descriptions and match manually their skills with the ones from job descriptions. Thus, the necessity for new solutions for effective automated job recommendations is more crucial than ever before.

Lastly, the impact of this thesis will have a direct effect on the RSs of Magnet.me as it aims to improve personalised job recommendations of the company, implementing the model in a real-world production scenario. From business perspective, the contribution of this research at Magnet.me strategy aligns with its goals to increase customer utility and decrease churning for both users and recruiters. It is noticeable that, the outcome of this research will create a core value where the company can benefit from it.

Chapter 2

Literature

In this section, a variety of machine learning models and recommendation systems that handle data sparsity are introduced. It is worth mentioning that some of the models described, have been used for job recommendations while others for movie and product recommendation. Although, this thesis focuses on job recommendation, RSs across different domains, often utilize similar types of feedback were mentioned in the beginning and adopt approaches that are introduced in the next paragraph. Therefore, it is reasonable to consider implementing models from other domains for job recommendations. All the machine learning models presented below are under evaluation as potential candidates for the RSs this thesis aims to build.

2.1 Different types of RSs

As stated earlier in this research, RSs filter a massive amount of data and provide to the users the most relevant items according to their preferences. Personalised recommendations can be achieved by multiple types of Recommendation Systems. The most well-known types are the followings: Collaborative Filtering (CF), Content-Based Filtering (CBF) approach, and Hybrid Based which is a combination of RSs.

2.1.1 Collaborative Filtering (CF)

This method examines the user-item interactions (e.g 1 for click and 0 for not click) or ratings (e.g., 1-5), from past behavioral data. It creates a rating matrix where all users are mapped against all items having the rating or interactions as the connecting element between them, as referred by Fayyaz et al. (2020).

The rating matrix R is defined as follows:

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{bmatrix}$$

where $R \in \mathbb{R}^{m \times n}$, and r_{ij} represents the rating given by user i to item j. If no rating is given by user i for item j, then r_{ij} may be zero (e.g., no interaction) or left blank (e.g., no rating).

Collaborative filtering methods are consisted of two approaches:

1. User-based: The first technique focuses on identifying similar users to the desired ones based on common preferences. A desired user is the one who is active on platform by the time that a recommendation take place. The second phase involves using the ratings from similar users to the desired (target) one in order to generate recommendations for the latter. This technique is adopted on this research.

2. Item-based: The second technique, as described by Sarwar et al. (2001), searches the items that the target user has interacted with or rated and compares their similarity to other items that the user not rated yet. Based on this contrast, the items that share the most similar characteristics with the already rated items are the ones to be recommended to the user.

It is worth mentioning that several algorithms as was noted by Sarwar et al. (2001), such as Pearson correlation, cosine similarity and embeddings, are commonly used to calculate these similarities either for user-based or item-based approach. Lastly, one major drawback of CF is the inability to handle the cold start problem as it needs interactions from users in order to generate recommendation to them. Without historical data on user behaviour and characteristics (e.g., interactions), the CF cannot provide relevant recommendations.

2.1.2 Content-Based Filtering(CBF)

This approach provide recommendations by estimating the similarity between a profile of a user (e.g.,resume, skills) and a profile of an item (e.g., job description), respectively, as noted by Yuan et al. (2016). More specifically, similarity refers to how similar is a user-item profile based on common characteristics and is measured either by using cosine similarity, pearson correlation or TF-IDF. This RS method is applicable in domains where textual data of users and items can be easily gathered and used.

Unlike CF, the content-based filtering does not have any cold start difficulties. New items are proposed to the users before the latter interact with the former. Undoubtedly,

CB has its own limitations as well. More specifically, if items do not have enough content information to distinguish it from each other, the recommendations will not be that relevant as stated by Fayyaz et al. (2020). In addition, if user and item profile have limited information, the suggestion will not be so personalized as the RS approach cannot match the characteristics from these two profiles.

2.1.3 Hybrid-Based(HB)

Hybrid approach is a combination of methods of RSs for more relevant recommendations. The main goal is to eliminate the disadvantages of the individual recommendation systems methods. Based on Fayyaz et al. (2020), HB work with weights where first aggregates the results of all combined RSs and then a score is assigned to the recommended item. In addition, it can work as a switching method, for different user, another technique can be implemented based on a criterion which defines the choice of method. Lastly, the last well-know HB strategy is the mixed one, where RSs are being used interchangeably.

2.2 Techniques for Measuring Similarities of Users-Items

As was mentioned earlier, there are several ways of measuring the similarities between items, users or item-user. Beginning with Cosine Similarity, as was referred by Sarwar et al. (2001) it measures how similar two data points are by calculating the angle of their vectors. If the angle is small, the vectors are on them same direction indicating data similarity and vice versa. Also, according to Yuan et al. (2016), Pearson Correlation calculates how related are two data points. It shows how strongly (e.g., score ranges from -1 to 1) and in which direction (positive or negative) these two data points are heading to. In addition, TF-IDF is a statistical method used to understand the importance of words between a document and other related documents, making it ideal for comparing and searching for similarities within texts.

2.3 Machine Learning Models Addressing Sparsity

Throughout the years, various machine learning (ML) models have been adopted in order to alleviate data sparsity and improve personalized recommendations. In this subsection, several ML models (including their recommendation system type) are introduced as potential candidates for this thesis. It is worth mentioning that some models have been used for job recommendations, while others only for movie recommendations. However, a model tested on movies can still work for jobs because both rely on matching users

with items (e.g., movies or jobs). Whether using explicit feedback (e.g., ratings) or implicit feedback (e.g., clicks or views) the model can be adapted to work with data for job recommendations. Lastly, it is noticeable that each model has its own advantages and disadvantages, which must be taken under consideration before deciding the optimal one for this research.

2.3.1 Graph-Based (HB)

The graph-based method, as noted by Shalaby et al. (2017), takes advantage of Neural Network of Deep Learning family for hybrid recommendations by combining behavioral information from CF and job embedding from the CBF approach to alleviate data sparsity. This method creates a unified graph where jobs are represented as nodes, connected by edges that correspond to aggregated similarity scores. The latter is derived from user behavior (e.g.,, co-clicks, co-applications) and job embeddings generated from job descriptions using a Deep Learning Matcher. With this way the model reassures that even jobs without interactions but including content information, can be incorporated into the recommendation system. Utilizing the content-based filtering, the data sparsity in that article is handled,by creating scores from job embeddings.

The advantages of this approach, is the way that the graph is structured as it allows the system to handle successfully millions of jobs and users. It is worth mentioning that it has been implemented on a real case study of CareerBuilder (recruitment platform) outperforming the performance of traditional CF models, such as Matrix Factorization (MF), based on user interaction. However, a concern that arises from this approach is the manual way of tuning the parameters instead of well-known validation trial, setting room for improvement in future development of this model. Lastly, similar to Neural Graph Collaborative Filtering (NGCF) that is presented on the next paragraph, this method relies on a graph structure for large datasets. However, while NGCF focuses only on user interaction information (CF), Graph-Based (HB) combines both content and interactions for recommendations, making it a hybrid recommendation system.

2.3.2 Neural Graph Collaborative Filtering (CF)

The NGCF, introduced by Wang et al. (2019), is another approach from the Deep Learning family which aims to present relevant recommendations to users making use of both first-order and high-order connectivity in the user-item graph. In simple terms, first-order connectivity is the direct interaction between a user and an item while high-order is the interactions from several deep layers (e.g., friend of a friend). The high-order connectivity, reinforces the model to provide accurate suggestions by incorporating collaborative signal from a broaden network of user interactions addressing at the same time the data sparsity issue. The NGCF takes advantage of the embedding propagation

layers which spread information from farther neighbors assisting users or items with less interactions to receive valuable information from their nearby neighbors.

In addition, as this model captures better deeper user-item interactions in contrast to traditional models such as MF or the Graph-Based (HB), it has the capability to recommend more relevant suggestions. In the article, NGCF was tested successfully in three large datasets (e.g., Amazon-Book - product recommendation) making a suitable method to handle scalability problems. Nevertheless, it should be referred that deep learning models suffer from overfitting especially when many layers are stacked together on the tuning phase and even if there are techniques to tackle overfitting (e.g., node dropout), they are quite challenging.

2.3.3 Matrix Factorization with Side Information (CF)

A recent study by Behera1 et al. (2024) presented that Matrix Factorization technique can integrate auxiliary(extra) data as side information to address sparsity problem of collaborative filtering. More specifically, the model aims to forecast user ratings by incorporating demographic attributes (e.g., age, gender, and occupation) that used as bias terms and vectors. In general, the traditional model "Matrix Factorization" functions by decomposing the user-item rating matrix into lower-dimensional matrices, learning latent feature factors for both users and items. The objective of this process is to approximately re-structure the original matrix by multiplying the matrix of users with the one of items, predicting missing ratings by capturing hidden relationships between them.

The additional user-side information significantly enhances the traditional model performance, cope data sparsity and defeat quite a few models such as Singular Value Decomposition (SVD) and Probabilistic Matrix Factorization (PMF). Nevertheless, there is no doubt that adding side information into Matrix Factorization , it increases its complexity and the need for more computational resources as highlighted by the hardware requirements in Behera1 et al. (2024). In contrast to MF, which requires matrix decomposition, the methods of Graph-Based (HB) and NGCF use graph structures or content embeddings to seek hidden patterns (e.g.,latent features), making it easier and more effective to handle large datasets and complex relationships.

2.3.4 Bayesian Personalized Ranking (CF)

The article of "Bayesian Personalized Ranking (BPR)" by Rendle et al. (2009) depicts an interesting approach of BPR criterion, derived from Bayesian analysis, for improving personalized recommendations. In contrast to the already mentioned recommendation models, the main difference of it is that BPR aims to rank the items rather than forecasting the potential ratings or interactions. More specifically, BPR generates ranking pairs of items for each user, having as an objective to increase the likelihood that the

models recommend the observed items (e.g., interacted) higher in the recommendation list of a user than the unobserved ones (e.g., non-interacted), alleviating simultaneously the sparsity issue. The process is engaged by sampling pair of items and adjusting the parameters of the model through stochastic gradient descent (SGD). In this article, the model was implemented for movie recommendations and outperformed a series of models such as , singular value decomposition(SVD) with MF and k-nearest neighbours (kNN). In contrast to NGCF, which focuses on embedding propagation, and Graph-Based models, which utilizes content analysis for predicting ratings of items, BPR try to rank the rated items.

Thus, the first advantage of this model is its design for specifically personalized ranking approach which is quite valuable for many real cases recommendation tasks. Secondly, it is flexible to be integrated with various models, such as MF and k-Nearest Neighbours (k-NN), and well-structured for handling implicit feedback data (e.g., clicks, views etc). However, careful tuning of the regularization parameter is a major concern to avoid overfitting which in cases of presence raise the model complexity whereas creating pair of times for every user individually can be computationally expensive, particularly for large datasets. Lastly, the current thesis proceeds with the implementation of BPR model, and the supporting arguments for this choice are presented in the methodology section.

2.3.5 Word2Vec and K-means Clustering (CBF)

Another approach for personalized recommendations was introduced by Mhamdi et al. (2020) using a model that leverages text clustering and user engagement. This method classify job opportunities based on common attributes such as job titles, skills and description. More precisely, the model uses Word2Vec (Neural Network model) to capture semantic(content) similarities between job opportunities and k-mean clustering to group them into insightful clusters based on shared characteristics. The recommendations are generated by analyzing user interactions (e.g., application, likes) and matching this information with the clustered jobs. Finally, when a user engage positively with a job from a cluster, other jobs within that clusters are suggested to the user, aiming to approach the user preferences.

The model handles sparsity by combining the current two methods, providing recommendations even when users have constraint engagement. Similar to Latent Dirichlet Allocation (LDA) that is introduced in the next paragraph, the current method also utilizes content-based filtering (CBF) to handle sparse data, but while LDA relies on probabilistic modeling (probability distributions to represent how possible certain words or topics appear in a document), Word2Vec is effective at processing unstructured data, such as job descriptions, using word embeddings. Both LDA and Word2Vec focus on content rather than user engagement, in contrast to collaborative filtering (CF) methods

such as NGCF and BPR. However, as mentioned earlier, a threat to this approach is when information is limited, not being capable to observe similarities between documents.

2.3.6 Latent Dirichlet Allocation (CBF)

In 2017, Bansal et al. (2017) deployed another strategy for job recommendations by leveraging the content-based filtering recommendation of Latent Dirichlet Allocation(LDA). This method takes advantage of natural language processing(NLP) to gather features such as skills and job descriptions, and uses word patterns along (n-graphs) to understand word relationships (e.g., Data Scientist). LDA produce topics models, which are groups of related words to create themes within the text which represent both job descriptions and user profiles. Furthermore, it matches jobs to users based on similarity scores derived from common characteristics within job and user profiles.

By leveraging topic modeling, LDA can handle data sparsity and generate suggestions even if a user has less engagement with an item unlike BPR and NGCF (CF) which rely on user engagement. However, it should be taken into account that the computational complexity due to natural language processing can be challenging, making it less powerful to handle large amounts of data as Graph-Based model and NGCF.

Chapter 3

Data

The current chapter includes the data collection, description and exploration as well as some challenges that were faced preparing the final dataset for analysis. A deep understanding of these components is necessary for the preparation of our dataset and interpretation of the results.

3.1 Collection and Description

The datasets used for this research were provided by the Magnet.me database, and the author was granted permission to use the data for the academic purpose of this thesis. The datasets span from January 1, 2022, to December 31, 2023. The first dataset, called "opportunity interactions," comprises 2,623,832 observations and 3 columns, while the second dataset, named "opportunity attributes," consists of 174,008 observations and 5 columns, respectively. The former includes information about user (job seeker) interactions with items (job opportunities) on the platform, including explicit feedback in the form of user ratings, which represent evaluations of users on job opportunities. These explicit ratings are used as a direct relationship of user engagement with the jop opportunities. The second dataset, "opportunity attributes" contains information about the job opportunities themselves. Before the columns are described, it is worth mentioning how the data was gathered based on user engagement on the Magnet.me platform for a better understanding of the dataset.

When users create their profiles and log into the platform, they see job cards (job opportunities) on the user interface generated by the current recommendation system of Magnet.me. At this stage, they can engage with the job cards by choosing either the "ignore" or "like" buttons to skip or like the jobs(adding it on favorites), respectively,. After that interaction, the following job card is displayed. In case a user selects to explore additional information on the job card, this is recorded as a "view." At last, an "application" interaction is tracked when users apply for a job, whether directly from the view phase or from their favorites. Therefore, the first dataset "opportunity interactions"

is consisted by the following columns:

Table 3.1: Description of Variables in the "Opportunity Interaction" Dataset

Column	Description
job_seeker_id	A six-digit integer representing the user ID.
opportunity_id	A six-digit integer representing the job ID.
interaction	An integer value representing the type of interaction: 1 (ignored), 3 (view), 4 (like), 5 (application).

The Table 3.1 describes the columns of first dataset which includes user-item interactions. The first and second column includes the user and item IDs, respectively, whereas the third column describes the type of interaction a user had with an item. In detail, when a user (job seeker id) ignores an item (opportunity id), it receives a rating of 1 (interaction). Viewing the job details gives a rating of 3, while liking or applying for the job opportunity obtains ratings of 4 and 5, respectively.

The company adopted this 1-5 rating system, originally used for movie reviews, and adapted it for job recommendations. In general, a 5-star review represents the best rating for a movie, indicating that the viewer found it highly relevant and interesting. Adapting it to job recommendations, a 5-star rating represents an application submission by job seekers to job opportunities that are relevant based on the user preferences. On the other hand, a 1-star rating in movie recommendations reveals that the film does not reflects the preferences of a viewer. Similarly, in job recommendations, a 1-star rating indicates that the job is irrelevant to what the user is interested to. Heading further, a 4-start rating corresponds to a liked items by users without submitting any application for it. However, the fact that users liked the items suggests that they are quite related to user preferences. Also, a rated item with the number of 3 (e.g., viewed) indicates that the job seeker devoted few time searching more information about the job opportunity, even if it did not match with his interests. Thus, the ratings 1-5 illustrate how relevant is a job to a user. Lastly, rating 2 is absent from the current rating system as the Magnet.me company decided to make a strong distinction between ignored job opportunities and the ones that the user viewed, spending time on them.

Heading further, as was mentioned above the second dataset "opportunity attributes" contains information about the job opportunities the users see and is comprised of the below columns:

Table 3.2: Description of Columns in the "Opportunity Attributes" Dataset

Column	Description
opportunity_id	Unique ID for each job opportunity.
tag	Type of job opportunity (e.g., job, internship).
detected_language	Language in which the job opportunity is posted (e.g., English, Dutch).
city	City where the job is provided.
job_function	Job category (e.g., consulting, finance, IT).

On Table 3.2 we examine the characteristics of each posted job available on the Magnet.me platform. Every job has a unique ID, a job type (e.g., internship), a posting language, a city location in the Netherlands, and a job function (e.g., IT, finance etc). For the purpose of our analysis, as will be explained in the following paragraphs, we use only the first two columns of this dataset.

3.2 Exploration

In this section, we explore our datasets and proceed with some minor pre-processing steps for better investigation of them. The examination of the data took place on jupyter lab environment by Python on 3.11.7 version. More details about the computational resources were used are given on the methodology chapter.

3.2.1 Opportunity Interactions Dataset

Exploring the first dataset, we examine 61.332 unique users and 174.008 job opportunities with total interactions 2.623.832 as the number of rows. Nevertheless, it should be mentioned that the almost 2.5 million observations includes duplicates interactions as a user can have multiple interactions recorded with a single job. For instance, a job seeker could first view a job opportunity (rating-3) and then ignore it (rating-1). Also, a user may have liked a job (rating-4) and then apply for that job (rating-5).

Although, in order to be more precise on our exploration and analysis, we aim to keep only the higher ratings of users interacting with unique jobs and that because a high rating 4 would reflect to a more relevant job for a user than a single view on it with a rating of 3. As a result, we decide to keep the greatest numbers on every occasion in contrast to the lowest ones. After removing the duplicates ratings-interactions of job seekers, the first dataset "opportunity interactions" ends up with 1.934.734 observations or in other terms unique ratings - interactions of users to items.

Exploring the first dataset, Figure 3.1 exhibits the distribution of the unique ratings - interactions of users to items. Undoubtedly, the "ignored" interaction captures the greatest percentage of interactions, rising up to around 57.8%. That proportion symbolizes that users ignored the job opportunities they were recommended, as they found them irrelevant according their interests. Moving further, the "view" interaction reaches 25.7% , indicating that almost one-fourth of the job seekers read the job description details but decided to churn the job opportunity. The "like" interaction gathers nearly 11.2% where users liked an item but, for some reason, decided not to apply for the job opportunity. Lastly, 5.3% involved users applying for a job opportunity, reflecting to meaningful recommendations of job opportunities to users based on their interests.

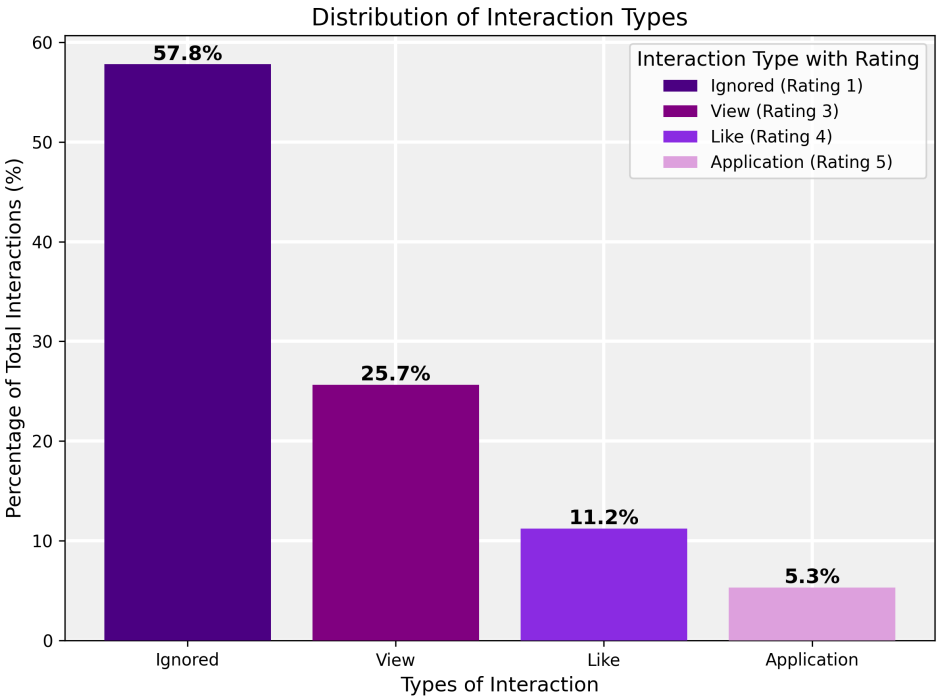

Figure 3.1: Distribution of Interaction Types

Continuing the exploration of the dataset, it is worth focusing on the number of unique job opportunities each user has interacted with in order to have an insight on how job seekers behave in the platform. The Table 3.3 provides a summary of the total unique ratings on job opportunities across 61,332 unique users. These users ,as was mentioned earlier, have been engaged with 1.934.734 job opportunities, which corresponds to the number of rows in the first dataset. On average, a user interacted with approximately 31.55 jobs while their navigation on the platform. Also, the standard deviation, is quite high at 137.02, representing the variability of interactions as some users interacted with more jobs opportunities than others.

The minimum number of interactions recorded for any user is 1, indicating that even the least active user explored at least one job opportunity. At the 25th percentile, users

had 2 or fewer interactions, while the median user (50th percentile) had 6 interactions. This reflects to a skewed distribution, as half of the users engaged with 6 or less job opportunities, but the mean is significantly higher than the median. This imbalance indicates that a small subset of users interacted with a disproportionately large number of job opportunities, as a result to increase the mean. Furthermore, at the 75th percentile, 75% of users interacted with 21 or fewer job opportunities. Surprisingly, the most active user interacted with 9,314 opportunities, showcasing an unusual level of engagement.

Table 3.3: Summary Statistics of Unique User-Job Opportunity Interactions.

Statistic	Interactions
Total Interactions	1.934.734
Mean	31.55
Standard Deviation	137.02
Min	1
25th Percentile	2
Median (50th Percentile)	6
75th Percentile	21
Max	9,314

Having seen, the total interaction summary statistics and the distribution of interaction types (e.g, ignore , view etc.), it is worth examining the distribution of unique interactions for each user on job opportunities. In other terms, we will study how many types unique users interact with job opportunities. The Figure 3.2 reveals that about one-third of people interacted with 1 to 2 items, while the largest proportion, 43.2%, engaged with 3 to 20 job opportunities. In other terms, the 74.8% of users interacted with 20 or fewer items, indicating that the majority of users do not interact that much with job opportunities.

Various factors may drive negatively user engagement. One possible reason might be that users experience the navigation and user interface challenging, which could prevent them from being interactive with items. Also, Another reason may be connected to the performance of the current recommendation system. In case the RS of the company suggests irrelevant jobs to users, the latter interact with fewer items , resulting in lower satisfaction and churning.

Heading further, 13.0% of users interacted with 20 to 50 job opportunities, and 6.1% engaged with 51 to 100 jobs. These percentages reveal that a portion of users spends more time on the Magnet.me platform and are quite interactive. In addition, 5.4% of users engaged with over 100 jobs but fewer than 500, indicating a small group of highly engaged users.

Additionally, we notice that only 0.7% of users interacted with more than 500 job

opportunities. This depicts that quite a few users are very active, and they are related with a small proportion of the total interactions.

Lastly, in the current plot, the bin ranges were selected to strongly distinguish different levels of user behavior based on the summary statistics. The (0, 2] bin captures users with very low engagement, matching the 25th percentile, while the (2, 20] bin includes most users, as it includes the median (6 interactions) and increases up to the 75th percentile (21 interactions). Larger bins such as (50, 100] and (100, 500] are used to represent the smaller group of users who engage more frequently. Finally, the (500, 10000] bin captures outliers, such as users with extremely high activity, like the maximum of 9,314. This choice of bin ranges demonstrates a balanced and clear picture for both regular and extreme users engagement.

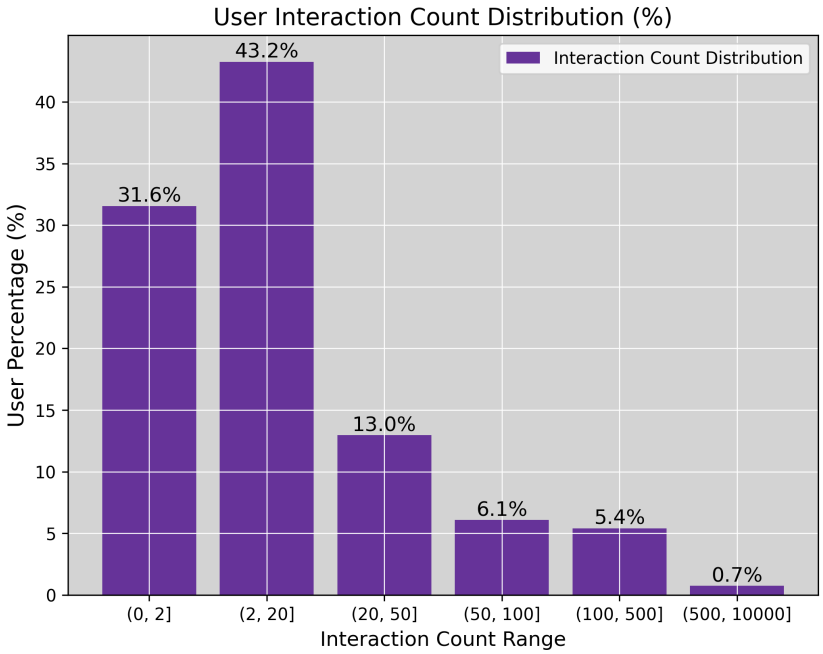

Figure 3.2: Distribution of Interaction Count

Before proceeding with the exploration of the second dataset, it is important to mention one more time the issue of data sparsity in the opportunity interaction dataset and explain how it can be calculated. As was stated by Yuan et al. (2016) in the major challenges of RSs in the introduction chapter, the inadequate interactions of job seekers to job opportunities results in a sparse rating matrix and can be seen as missing values.

To discover the data sparsity problem in our dataset, we first calculate the total number of possible interactions by multiplying the number of unique users by the number of unique items. Thus, multiplying 61.332 unique users by 174.008 unique job offers generates a total of 10.672.258.656 potential interactions.

In the meantime, the actual interactions which corresponds to the total number of

rows in the opportunity interaction dataset is 1.934.734. Following this, the density of the rating matrix is calculated by dividing the number of actual interactions by the total number of possible interactions. However, due to our interest in sparsity, we deduct the density from one. The formula below illustrates the calculation of sparsity in the rating matrix, where all users are mapped against all items. The matrix contains both zero and non-zero elements, representing the absence and presence of interactions, respectively.

$$\begin{aligned} \text{Sparsity} &= 1 - \frac{\text{Number of Non-Zero Interactions}}{\text{Total Possible Interactions}} \\ &= 1 - \frac{1,934,734}{61,332 \times 174,008} \\ &= 1 - \frac{1,934,734}{10,672,258,656} \\ &\approx 0.9998 \,\text{or}\, 99.98\% \end{aligned} \tag{3.1}$$

Undoubtedly, the rating matrix is extremely sparse, as the density or in other terms the actual interactions captures only the 0.02% percent of all possible interactions. Once more, the model that is presented on this thesis aims to cope the 99.98% sparsity and increase the personalized job recommendations on Magnet.me

3.2.2 Opportunity Attributes Dataset

In this part, we explore the opportunity attributes dataset provided by Magnet.me, which contains information about job opportunities. The insights gained from this exploration will provide a better understanding of the recruitment market from the perspective of recruiters and businesses.

The Figure 3.3 displays the distribution of job opportunity types, showing that almost fourth fifth are defined as job, identifying it as the most popular type of opportunity offered followed by internships with a 13.2% of all job opportunities. It can be seen that a significant percentage of available job postings includes temporary or entry-level roles. In addition, Traineeships at 3.2% and graduate internships at 2.0% are smaller groups, reflecting to a minor proportion of jobs falling under this job types. Lastly, 0.8% of jobs belongs to the "Other" category, which contains sub-categories such as business courses, in-house days, and events. Events are typically one-day networking activities, while in-house days focus on introducing job seekers to the roles of a company. Business courses are events designed to attract users by offering deeper engagement with the company. The last job type is the least frequent among all the other available job types.

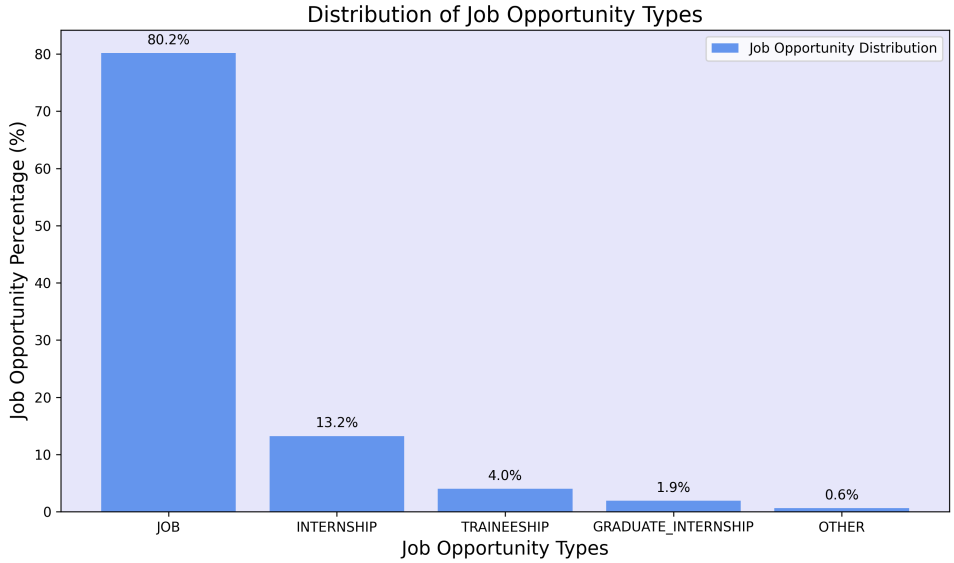

Figure 3.3: Distribution of Job Opportunity Types

Heading further, the remaining columns of this dataset is the ”detected language”, ”city” and ”job function”. Those columns are not taken into account in our analysis but are detailed explained in Appendix 7, to gain significant insights of the recruitment market in The Netherlands.

Chapter 4

Methodology

In this chapter, we present the criteria for the model selection as well as the features of the model and how it handles data sparsity. Additionally, we discuss the development and evaluation of the model, focusing on ranking metrics and introducing the current recommendation system of Magnet.me which served as the baseline to compare it with the thesis model. Finally, we will mention the computational environment used for implementation.

4.1 Criteria for Model Selection

Taking under consideration the available computational resources and time constraints of this research as well as the features of the presented models we decided to proceed with the BPR model to improve personalized job recommendations at Magnet.me facing the data sparsity issue. The key reasons to support our choice are presented below.

4.1.1 Focus on Ranking

To begin with, as was stated by Rendle et al. (2009), BPR emphasize on comparing pairs of items for each users and ranking more relevant items higher and less relevant ones lower, instead of predicting absolute ratings as Matrix Factorization does. This ranking approach of BPR is ideal for job recommendations where the primary goal is to suggest the most relevant jobs to users based on their preferences as early as possible within the recommendation list. The pairwise comparison of BPR aim to place the most relevant items at the top of the recommendation list, making it accessible as users interact with the recommendation system, indicating with their engagement what they find relevant and not relevant.

4.1.2 Effectiveness in Sparse Data Environments

Another key point of BPR is its ability to deal with sparse datasets in contrast to traditional approaches such as Matrix Factorization, which struggles with limited interactions. Noted by Rendle et al. (2009), BPR addresses this problem by prioritizing the observed interactions (e.g. non-zero elements) over unobserved ones (e.g. zero elements) making the recommendations more relevant by focusing on user interactions that represents valuable interest in the items they have interacted with. Therefore, it is a suitable tool to handle data sparsity and can be applied in our research where a limited amount of interactions has created a 99.98% sparse dataset.

4.1.3 Computational Efficiency

Additionally, a major factor that we thoroughly considered in selecting BPR is its computational efficiency. This method leverages a simpler approach whereas more complex models like NGCF or Word2Vec methods demand significant computational power due to their deep learning structures which leads to complex computations. In fact, the optimization algorithm - stochastic gradient descent (SGD) is the key feature of BPR making it a valuable tool for handling sparse data without without needing large computational resources. This advantage makes BPR an idea solution for this research as it performs well even on systems with limited computational power.

4.1.4 Comparison of BPR with the rest candidate models

Summarizing, BPR seems to be the best choice due to its simplicity and effectiveness in handling sparse data without requiring complex computations and additional information. As it has been already mentioned earlier, Matrix Factorization with Side Information adds complexity, while NGCF is prone to overfitting and is computationally demanding, similar to graph-based approaches. In addition, Word2Vec and K-means Clustering require strong NLP processing, and LDA contains complex probabilistic modeling. In contrast, the pairwise ranking method of BPR is simpler to implement given the computational resource constraints, and time limitations of this research.

4.2 Benchmark Models at Magnet.me

One of the models that Magnet.me had implemented for job recommendations was the Non-Negative Matrix Factorization (NMF). As was mentioned by Lee et al. (2024), NFM is collaborative-filtering method that predicts user preferences (e.g.,ratings) by identifying hidden patterns in a user-item interaction. It works in a similar way with the previously mentioned Matrix Factorization with the only difference to be that this model does not

accept negative values making results more interpretable. The goal of the company in the past years was the accuracy of predictions on recommending jobs to job seekers.

In recent years, the company implemented another collaborative filtering approach, item based, for job recommendations, which outperformed the previous NMF recommendation system. The new recommendation system called "NameSimilarity", searches for jobs with titles similar to those the user has interacted with. When a job seeker engages with a specific job title, the system seeks and suggests other job opportunities with similar title characteristics. This approach takes advantage of the TF-IDF (Term Frequency-Inverse Document Frequency) and uses it to distinguish the most important words in job titles by focusing on terms that appear often in a specific title but less frequently across other titles. In addition to this approach, the "Cosine Similarity" measure compares job titles by measuring how similar their TF-IDF representations (e.g., numerical vectors) are. The higher cosine similarity score is the more similar the titles are.

However, this approach apparently faces a major limitation where it only takes into consideration the job titles for text matching, ignoring other important features such as job descriptions, user skills and experience. This often leads to less relevant suggestions in case two jobs have similar titles but different characteristics on their main content. Since this thesis aims to improve job recommendations at Magnet.me while handling the issue of data sparsity, the chosen model, BPR, will be compared to the "NameSimilarity", serving as a baseline model.

4.3 BPR: Model Overview

4.3.1 Implementation of BPR on Matrix Factorization

Up to this stage of the research, BPR has been referred to as a model. However, BPR is more properly defined as an optimization criterion which is integrated into the classic Matrix Factorization (MF) model to improve personalized job recommendations and address data sparsity. As mentioned by Rendle et al. (2009), MF forecasts interactions of users on items by decomposing the interaction matrix into lower-dimensional user and item factor matrices. Nevertheless, this traditional approach focuses on predicting interaction scores (e.g., absolute ratings) rather than optimizing item rankings.

The integration of BPR ranking criterion on the top of MF leads the model focus on pairwise item comparisons. BPR ranks items that users have interacted with higher than those they have not, relying on implicit feedback. This method not only predicts whether items match with user preferences but also ranks them higher in the recommendation list based on those preferences. In contrast to models that utilize explicit feedback (e.g.,ratings), BPR originally leverages implicit feedback (e.g., user clicks, purchases) which is often easier to gather automatically throughout the user engagement

with the platform. The assumption of BPR is that unobserved interactions (e.g.,items not clicked or purchased) could either considered items the user is not interested on or items a user has not seen yet. Lastly, the model focuses on what the user has already interacted with. The combination of BPR criterion and MF model enhances the relevant recommendations.

4.3.2 Optimization Criterion (BPR-opt)

BPR optimization criterion, known as BPR-Opt, is derived from Bayesian analysis and its goal is to identify the best model parameters Θ that maximize the possibility that items are ranked correctly (e.g., high for the relevant and low for the non-relevant items) within the recommendation list based on each user interests. This approach, known as the maximum posterior estimator, combines well the model fits the data with past information (e.g., past interactions) to find the ideal parameters.

The BPR-Opt formula is the following:

$$\text{BPR-Opt} := \ln p(\Theta| >_u) = \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2$$

In this formula, $p(\Theta| >_u)$ represents the probability that the model settings Θ are accurate based on the ranking preferences of users $>_u$. The term $\hat{x}_{uij} = \hat{x}_{ui} - \hat{x}_{uj}$ indicates the difference in predicted scores for two items i and j for a single user u. This difference implies how much the model believes the user u chooses item i over item j. The sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$, translates this difference into a probability between 0 and 1, showing the possibility that the user prefers item i over j.

The term $\ln \sigma(\hat{x}_{uij})$ is the log-likelihood of the user to desire item i over j. The total sum adds this probability acroos all pairs of items that the model is learning from. The regularization term $\lambda_\Theta ||\Theta||^2$ helps to avoid overfitting by keeping the model simple and reassuring that the parameter values exists within a reasonable range.

The main idea of BPR is to use pairwise comparisons instead of predicting absolute scores for items. The model learns by comparing pairs of items for each user. For instance, if a user has engaged with item i but not item j, BPR assumes the user likes i over j. Following this way, BPR focuses on ordering the items correctly within the recommendation list, placing the most relevant ones higher and the least relevant ones lower, instead of predicting exact ratings. This approach makes the model better at ranking items by understanding which ones users are more possible to like. By learning these preferences, the model can generate more accurate and personalized ranking suggestions for each user.

4.3.3 Learning Algorithm (LearnBPR)

LearnBPR is the algorithm that trains BPR by improving how well it ranks items for each user and updates the model so that it learns from data. More specifically, LearnBPR adjusts the parameters of the model (Θ) through a technique called stochastic gradient descent(SGD). Instead of investigating the whole dataset, it proceeds with minor updates on the model using bootstrap sampling, a technique that selects small, random samples of data (a user and two items). With this way, it allows the algorithm to learn effectively and quickly by processing these smaller subsets rather than the full dataset at once. Each sample, also know as triple, consists of a user, an item they have interacted with, and an item they have not. LearnBPR adjusts the model so that the items users have interacted with are ranked higher in the recommendation list than the ones they have not interacted with.

The above process is repeated multiple of times, continuously improving the power of the model to capture user preferences and rank items better. Using the bootstrap method, LearnBPR manages to handle large datasets and avoid overfitting, making the model a promising tool to perform well on unseen data. Summarizing, LearnBPR is the algorithm that optimally tunes the BPR model, assisting it to predict which items the users are possible to prefer based on their past interactions.

4.3.4 Handling Data Sparsity

Bayesian Personalized Ranking (BPR) handles data sparsity or in other terms missing data by focusing on pairwise item comparisons and leveraging implicit feedback , where only positive interactions (e.g.,clicks) are observed. Furthermore, BPR handles the overall negative interactions (e.g., zero elements of the rating matrix) as a combination of no interactions and those that users responded to negatively.

The model considers that if a user has interacted with item i but no with item j, the interacted item is the preferred one. This comparison between observed (positive) and unobserved (negative) interactions allows BPR to effectively manage missing data. Lastly, the model ranks items the user interacted with higher, without needing to identify unobserved items as negative.

4.4 Model Development

4.4.1 BPR Leveraging Explicit Feedback

As Rendle et al. (2009) notes in the paper of BPR, the model is designed for implicit feedback, learning from actions like clicks. However, the pairwise ranking method of BPR can be easily adapted on explicit feedback, such as ratings or interactions (e.g., likes and

dislikes). As previously stated, BPR considers that users prefer items they have interacted with over those they have not. The same explanation can be applied to explicit feedback, where items with high ratings are ranked higher (e.g., positive interaction), and the ones with low ratings are ranked lower (e.g., negative interaction), in the recommendation list. Additionally, LearnBPR, the stochastic gradient descent (SGD) algorithm, can successfully handle explicit ratings by creating rankings based on strong user preferences like ratings. Despite the fact that BPR is originally built for implicit feedback, its structure makes it flexible for handling explicit feedback as well, which was used in the current research.

4.4.2 Data Preparation

In this part, we refer to the steps were taken to prepare the dataset "opportunity interactions" before we incorporated it into the chosen model. Having 2.623.832 rows and three columns (job seeker id, opportunity id and interaction) we split the data into 90% - 10% for training and test set , respectively. This allows us to use a large part of the data to train the model while keeping a separate portion to compare the performance of the BPR model with the baseline model on the same consistent data, ensuring both models were tested under the same conditions.

As was mentioned on the data chapter, the "opportunity interactions" dataset contains duplicated ratings. After splitting the dataset, the first pre-process step we proceed is to keep only the higher ratings of users interacting with unique jobs for reasons were explained on the data exploration section. After removing the duplicates interaction - ratings of job seekers, the training set "opportunity interactions" ends up with 1.775.111 observations or in other terms unique ratings - interactions of users to items.

To begin with, on the second dataset "opportunity attributes" the smallest percentage of job type of "others" (Figure 3.2) was excluded from "tag" column because the subcategories that were belonging to that type were not a normal job but rather engagement with companies. Also, after having excluded the "other" type of job from "opportunity attributes" dataset we exclude the corresponding "opportunity id" of job opportunities from "opportunity interaction", training dataset.

Secondly, from the "opportunity interactions" dataset, we selected only the unique users who had between 3 and 500 interactions. As shown in Figure 3.2, almost one-third of users have just 1-2 interactions. The reason for excluding users with fewer than 3 interactions is that the focus of this thesis is on handling data sparsity and not the cold-start problem. We assume that users with 1-2 interactions are cold start users, and we support this decision based on how the chosen model operates. The model needs interactions to identify patterns, and when a user has very few interactions, the algorithm is not able to use the limited data to generate trends or recommend relevant job opportunities.

Therefore, including such users would only introduce noise into the data.

In addition, we constraint the interactions to a maximum number of 500 because users with more than 500 interactions have engaged with so many jobs that it becomes difficult to shape clear preferences. This intense activity complicates the ability of the model to generate patterns and relevant suggestions. Based on Figure 3.2, we decided to focus on the majority of users, who rises up 67.7% of the dataset. In addition, the thresholds were chosen as a trade off between noise reduction and verifying we have enough data points for the implementation of the model. However, we recognize that the cut-off of the interactions could have taken place in another range reflecting to a different model performance and this is referred as a limitation of our research on the discussion chapter.

Furthermore, regarding the "interaction" column in the "opportunities interaction" training dataset, we converted ratings of 4 and 5 to 1, and ratings of 1 and 3 to 0. Specifically, we treated jobs that were liked or applied by users as relevant jobs (positive interactions), while jobs that were removed or viewed were considered irrelevant (negative interactions). Converting 1-5 ratings to binary (1-0) interactions is strong supported by the design of Bayesian Personalized Ranking (BPR) which is created specifically for systems that track positive interactions instead of relying on ratings.

As was stated by Rendle et al. (2009), the model treats interactions as positive (1) and negative (0). Negative interaction is a combination of negative feedback and missing values (items the user has not interacted with yet), which supports our decision on altering ratings into binary values. Since the model works on pairwise ranking, prioritizing user interests between raking items rather than absolute ratings, converting ratings to binary values aligns it with the main goal of BPR, improving personalized ranking accuracy. Lastly, after the preparation of our data, the training dataset is consisted of 1.252.156 observations and 48.891 unique users.

4.4.3 Cornac Framework

In this research, we developed the recommendation model using an open-source Python framework, called "Cornac" (Salah et al. (2020) specifically for building and evaluating recommendation systems. Although Cornac offers an extensive range of advanced tools (e.g., Deep Learning Models) for leveraging auxiliary data such as text, we decided to focus on simple models due to the time constraints of the research and the available computational resources. Specifically, we implemented the Bayesian Personalized Ranking (BPR) model, which we considered to be the ideal solution in our study.

4.4.4 Hyperparameters Tuning

In this part, we introduce the process and rationale of tuning the hyperparameters of our BPR Model which allows us to control how a model learns and obtain the best model

performance.

BPR consists of the following parameters: k latent factors, learning rate, max iterations and regularization. As was refered by Galuzzi et al. (2020), the "k latent factors" define the hidden dimensions shaping user-item relationships. These factors help decompose the rating matrix into two smaller matrices, capturing the underlying characteristics that drive user preferences while simplifying the data. A high number of k assists the model to capture more complex patterns but also raise the risk of overfitting, while a low number tends to generalize the model. For that reason "regularization parameter" (lambda) is applied and controls model complexity by penalizing large weights to prevent overfitting (good performance on training data, and not good on unseen data).

Heading further, as noted by Irani et al. (2022) the parameter "learning rate" manages how much the parameters of the model adjust with each step during the training process. A high number of it may cause the model to converge to quickly, missing the best values of it while a low number might make the process quite slow increasing not only the execution time but also the necessary computational resources. Lastly, the "max iterations" parameter indicates the number of times the model read the data to update its parameters. More iterations improve the way the model learns but it can lead to overfitting while a low number of might lead to underfitting.

Beginning the hyperparameters tuning process, we utilized "Grid Search" method which as was mentioned by Hossain et al. (2021) it tests all combinations within a given set of hyperparameters values training and evaluating the model for each one of them. The best combination of hyperparameters will be the one which raise the best performance of the model. Inspired by Galuzzi et al. (2020), we started our first hyperparameters tuning experiment with values in the range of 10,100 for K latent factors and number of iterations , while selecting values between 0.001,0.1 for the learning rate and regularization parameter. After completing the first experiment, the model selected the optimal hyperparameters as follows: 85 for k, 85 for max iterations, 0.08 for the learning rate, and 0.001 for lambda.

It is important to note that we only used the Normalized Discounted Cumulative Gain at rank 20 (NDCG@20) metric to assess and tune the hyperparameters of the BPR model. Optimizing hyperparameters using NDCG@20 in job recommendations ensures that the most relevant job opportunities are ranked higher, improving user experience. Since most users typically engage with fewer than 21 job listings (as shown by the 75th percentile on Table 3.3), focusing on the top 20 job opportunities aligns well with their behavior, making the recommendations more effective and meaningful for them. This metric as well as the rest ones will be thoroughly explained in the "Ranking Metrics" section.

As we can see the higher values of hyperparameters are chosen as the best one in the majority of them. Thus we are gonna run more experiments with a greater range and higher values on google cloud for execution time reasons. However, as noted by Hossain et al. (2021), when we aim to test different values for multiple hyperparameters, the number

of combinations increases multiplicatively. For example, testing 5 values for each of 4 hyperparameters results in 5*5*5*5 = 625 combinations. This approach is not practical due to both time constraints (e.g., if each combination takes 5 minutes, the total runtime would be 3125 minutes or nearly 52 hours) and limited computational resources. These concerns are supported by Wistuba et al. (2015) who argue that "Grid Search" can lead to high computational costs. Additionally, another hyperparameters tuning method called "Random Search", is a more flexible approach that selects hyperparameters randomly. Yet, it can still waste time on poor configurations and does not adapt its hyperparameters values based on previous trials.

Having seen some major limitations of hyperparameters tuning processes, we decided to proceed with a manual tuning inspired by the sequential optimization process. As was mentioned by Wistuba et al. (2015), in simple terms this method tunes step by step the hyperparameters allowing it to learn from previous trials. Instead of testing all possible combinations ("grid search") or random ones ("random search"), it begins by evaluating an initial set of hyperparameters, tracking the model performance, and then adjusting the next set of options based on model performance. This slow and steady process continues, targeting on the most promising areas where the best hyperparameter values are possibly to be found more quickly. An important advantage of this approach is that it decreases the need for unnecessary trials, saving both time and computational resources. By narrowing down the search space for the best values, it avoids testing hyperparameter combinations that are impossible to improve model performance.

Thus, we began by testing a few initial hyperparameter values and continuously updating our search based on model performance. This approach allowed us to focus on the most promising areas with the higher posibility of improving the model. Specifically, we tuned one hyperparameter at a time while keeping the other three constant. Once the best value for the adjusted hyperparameter was found, we kept it steady and moved on to tuning the next one, again keeping the remaining three steady.

For instance, we started by tuning the k latent factors. After identifying the value for k that yielded good results, we shifted our focus to the other hyperparameters. Once the model converged on a specific learning rate, we fixed it and proceeded with the tuning of the remaining parameters. After completing the first round of testing all the hyperparameters, we started again from the beginning, this time trying new values for the number of latent factors (k). We then kept the rest of the hyperparameters the same as in the first round. Overall, we went through every hyperparameter three times, adjusting each one in turn. It is worth noting that Cross-validation was not implemented on purpose to avoid the high computational cost and execution time from multiple parameter combinations. However, on the next paragraph we present the actions were taken to validate our model. Therefore, this gradual process helped us minimize computational resources, save time, and ensure that we have approach the optimal set of hyperparameters.

4.4.5 Evaluating BPR Robustness

After defining the best hyperparameters that lead to the best model performance, we need to validate our results and prove that the model is robust. Having mentioned that BPR did not run within cross validation, we decided to run BPR with the best hyperparameters on the training set of the "opportunity interactions" dataset to prove that the performance of our model remains constant across all runs. Observing that the standard deviation across all runs is minimal, and the metric consistency does not change dramatically, we would assume that our model is robust and generalizes well. However, the poor performance of BPR on the training set of "opportunity interaction" dataset in contrast to 3-4 times better performance on the test set, raise some concerns about the validity of the model. For that reason, to further strengthen the validity and generalization abilities of our model and in order to prove that our model does not overfit, we compare the performance of BPR and the baseline model(NameSimilarity) on a new dataset, named "dataset2024," which contains the same columns as the "opportunity interactions" dataset. Once again, as it is presented, our model performs well. Further details about the rationale behind this process will be provided on the next chapter.

4.5 Ranking Metrics

Having introduced BPR model, in this section we present the ranking metrics that were chosen for the evaluation. As previously mentioned, one reason to choose BPR for this research is because it leverages a ranking based approach, making it ideal for job recommendations where the goal is to rank the most relevant jobs higher and earlier in the list. To thoroughly evaluate the performance of the BPR model against the baseline, appropriate ranking metrics must be used. In recommendation systems, as was mentioned by Valcarce et al. (2020), ranking accuracy is crucial since users usually interact with only the top-N recommendations. As a result, ranking metrics are more crucial than rating accuracy, as the ability of the system to place relevant items at the top ranking significantly guides user satisfaction and improves the ability of recommendation systems to provide relevant suggestions.

In this study, we chose four ranking metrics to assess the recommendation system, each ranging from 0 to 1, with higher scores indicating better performance. These metrics are calculated for individual users and then averaged across all users to determine the overall score. The evaluation is conducted up to a specific depth, also known as the "cut-off" (e.g., N-recommendations) where the rankings are examined. As mentioned in the hyperparameters tuning subsection, we set the cut-off at 20 aligning with the 75nth percentile in the user behavior summary statistics table.

4.5.1 Precision

The Precision@N metric depicts the percentage of relevant items within the top-N recommendations, without considering their order. It is calculated by dividing the number of relevant items in the recommended list by the total items that were recommended. The Precision@N formula is as follows:

$$P_u@n = \frac{|L_u^n \cap R_u|}{n}$$

L_u^n is the group of n items recommended to user u, R_u is the set of relevant items for user u, and $|L_u^n \cap R_u|$ indicates the number of meaningful items in the recommended list. In addition, n is the total number of items that were recommended in the top-N list. This formula allows us to understand how well the recommendation system stores relevant items within the top-N list, emphasizing on the proportion of relevant items retrieved without taking their ranking into account.

4.5.2 Recall

The Recall@N metric illustrates how many relevant items are included in the recommendation list, compared to the total number of relevant items that are available for a user. It measures how many relevant items are successfully obtained by the system. Recall formula is defined as:

$$Recall_u@n = \frac{|L_u^n \cap R_u|}{|R_u|}$$

L_u^n is the set of n items recommended to user u (e.g., in our case 20), R_u is the set of relevant items for user u, and $|L_u^n \cap R_u|$ represents the number of relevant items in the recommended list. Finally, $|R_u|$ is the total number of relevant items available for user u. Recall assist us to assess how effectively the recommendation system captures all meaningful items within the top-N list, aiming to obtain as many items as possible, without considering their order.

4.5.3 NDCG

Normalized Discounted Cumulative Gain (NDCG) is a ranking metric that measures how successfully relevant items are ranked in a recommendation list, giving more weight to items that appear higher in the list. It uses a discount function to assign less importance to items as their rank goes down, since items at the top of the list are more likely to be seen by the user. NDCG is calculated as:

$$NDCG_u@n = \frac{\sum_{k=1}^{n} G(u,n,k)D(k)}{\sum_{k=1}^{n} G^*(u,n,k)D(k)}$$

$G(u, n, k)$ is the gain (relevance) of the recommended item at position k, for user u on top n items and $G^*(u, n, k)$ is the gain in the ideal (best possible) ranking. The discount function $D(k) = \log_2(k + 1)$ reduces the weight of items that appear lower in the list. NDCG compares the actual ranking of items to the ideal ranking, helping evaluate how well the system places relevant items near the top.

4.5.4 MRR

Reciprocal Rank (RR) is a metric that exhibits how quickly the first relevant item is presented in a recommendation list. It is calculated by finding the position of the first relevant item and taking the inverse. This means that the earlier the relevant item appears, the higher the RR score. The formula for RR is:

$$RR_u = \frac{1}{\min_k(L_u^n[k] \in R_u)}$$

In this formula, u represents the user, $L_u^n[k]$ is the item at position k in the list for user u, and R_u is the set of relative items for that user. The metric seeks for the position of the first relevant item ($\min_k(L_u^n[k] \in R_u)$) and then calculates the reciprocal (inverse) of that position. The lower the position (closer to the top of the list), the higher the RR score will be. When this metric is averaged across multiple users, it becomes Mean Reciprocal Rank (MRR), which shows us how fast relevant items are found on average for all users.

4.6 Computational Environment

The BPR implementation was developed and evaluated using various models and platforms. Data manipulation and development were done in a JupyterLab environment using Python version 3.11.7. The initial work took place on a MacBook Pro with an Apple M1 chip, featuring 4 CPU cores and 16 GB of memory. Early hyperparameter tuning was conducted on this machine, while more intensive final tuning, was performed on Google Cloud. Specifically, the n1-standard-4 machine type, equipped with 4 CPU cores and 16 GB of RAM, was used. Model performance was tracked using the Neptune.ai platform, a tool designed for monitoring and analyzing machine learning experiments.

Chapter 5

Results

This chapter provides a detailed overview of the results obtained from the evaluation of our model. It begins by describing the optimal hyperparameters, which were selected based on the performance observed on the test set. Next, we compare the BPR model with the baseline model to indicate the performance improvements of RS at Magnet.me. Finally, we validate the robustness of the BPR model with the optimal chosen hyperparameters through 10-fold cross-validation on the training set and by testing the model on a new, unseen dataset. This confirms the model ability to generalize effectively, addressing any concerns about overfitting or data leakage from the original training-test set.

5.1 Best Hyperparameters

5.1.1 Hyperparameters Plots and Optimal Selections

The plots in Figure 5.1 depict the hyperparameter tuning process, where for each plot, one hyperparameter is given different values while the other three remain constant at their optimal values. This approach allows us to observe the individual effect of each hyperparameter on the NDCG@20 metric for the model performance. After this tuning process, the optimal hyperparameters are the followings: k=200, max iter=1100, learning rate=0.1, and lambda reg=0.0075. The green line in each plot represents the NDCG@20 performance under the chosen hyperparameters values, keeping the other three unchanged, while the red line indicates the chosen value for each specific hyperparameter on every plot. By gradually adjusting one parameter at a time, we get a better understanding on how it influence the performance of the model.

The NDCG@20 Across Different k Values plot (top-left) depicts a progressively performance improvement as the number of latent factors (k) rises, especially up to k=200. After this point, the curve reaches a plateau, indicating diminishing returns while increasing k values does not have any important impact on the performance of the model. The smoothness of the curve and the "flattening" beyond k=200 helped us choose this value

as the ideal balance between model complexity, performance and generalization.

In addition, the NDCG@20 Across Different max iteration Values plot (top-right) describes a significant enhancement in performance up to around max iterations =1100, after which the curve starts showing signs of flattening. This means that that increasing iterations beyond this point would need more computational resources while the performance improvements would be minimal. Thus, we decided to proceed with this "balanced" value as we wanted to achieve good model performance and avoid long processes, heavy computation, and overfitting.

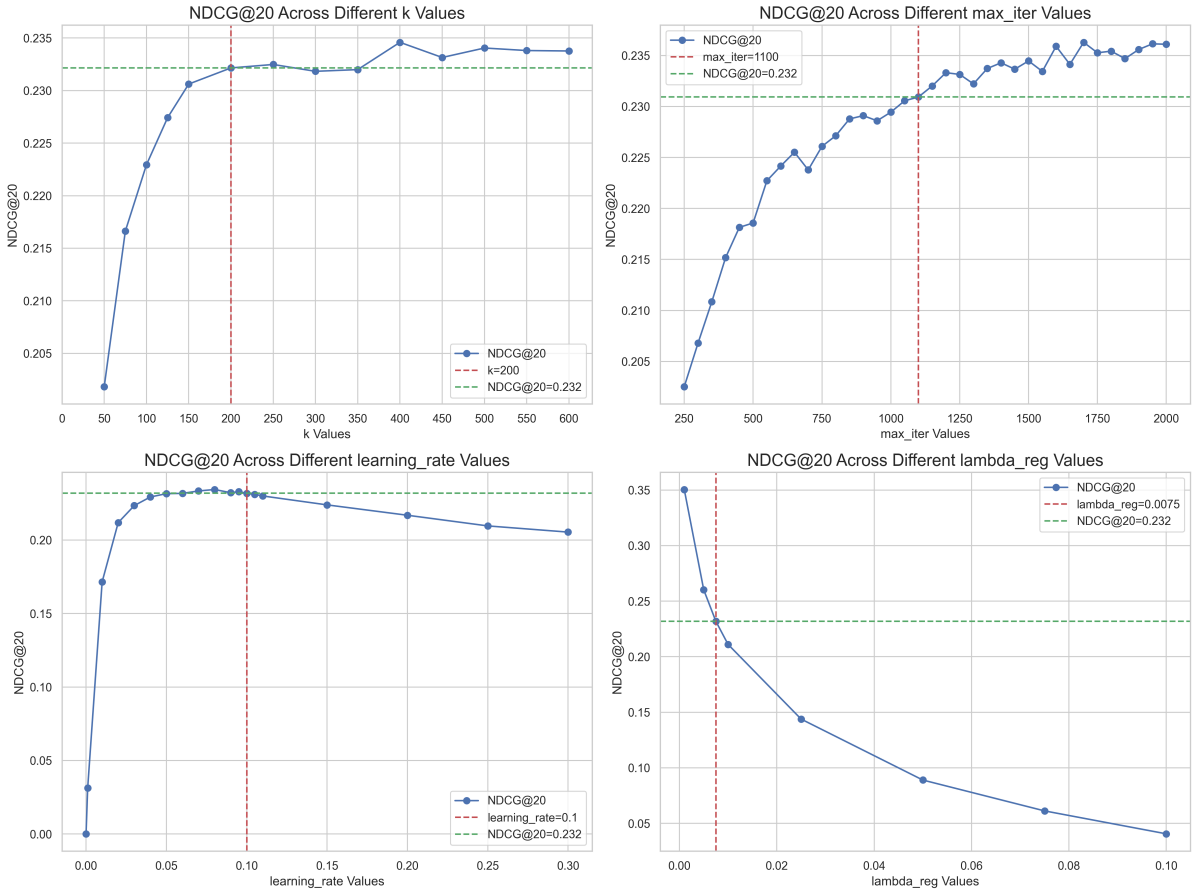

Figure 5.1: Hyperparameters Tuning

Heading further, in the NDCG@20 Across Different learning rate Values plot (bottom-left), despite the fact that lr = 0.08 shows slightly better performance than lr = 0.1, it was not significantly better. Learning rate=0.1 was chosen as it allows for faster convergence, making it more efficient while maintaining nearly the same performance as lr=0.8. The balance between speed and performance made 0.1 a better solution for our case.

Lastly, in the NDCG@20 Across Different lambda regularization Values plot (bottom-right), it is noticeable that boosting regularization negatively impacts performance, with the selected value to be lambda - reg=0.0075. Lower values (e.g., 0.001) could possibly lead to overfitting, while higher values could constrain the model too mutch, leading

to underfitting and minimizing its ability to capture important patterns. Therefore, we selected the regularization parameter at 0.0075 as a safe and balanced choice to avoid both underfitting and overfitting while still showing good model results.

5.2 Comparison of BPR with Baseline model

In this section, on Table 5.1 we compare the performance of two models, NameSimilarity (baseline model) and BPR, on the 10% of the "opportunity interaction" dataset (test set), using the following metrics: Recall@20, Precision@20, NDCG@20, and MRR. These metrics provide a detailed view of each model performance in recommending relevant job opportunities to job seekers. The BPR model reveals a significant performance across all metrics outperforming the NameSimilarity model of Magnet.me.

Beginning with NDCG@20, our model scores 0.232, demonstrating the quality of ranking within the top 20 recommendations. This shows that BPR retrieves and ranks relevant items quite effective, prioritizing the most important job opportunities at the top of the list. In contrast, NameSimilarity scores 0.123, which indicates that it performs less effectively in ranking relevant jobs. The 88% performance increase shows how much better BPR is at ranking and prioritizing items in the top 20 recommendations.

In advance, the BPR score of 0.0576 on Precision@20 reveals that around 5.76% of the job opportunities in the top 20 recommendations were relevant. On the other hand, NameSimilarity, with a lower score of 0.0185, provides less accurate recommendations than BPR does. The 211% improvement from BPR indicates its strong advantage in providing more relevant recommendations within the top 20. However, it should be noted that the score is still low, leaving a lot of room for improvement.

Furthermore, the Recall@20 score of 0.325 for BPR meants that approximately 32.5% of the relevant items were successfully acquired within the top 20 recommendations. This suggests that out of the 20 job opportunities recommended, around 6-7 of them were relevant to the job seekers. Oppositely, NameSimilarity receives a lower score of 0.195, suggesting only about 19.5% of relevant items. The performance of BPR model in contrast to NameSimilarity increased by 66% which presents the former model strength to track relevant job opportunities within the top 20 recommendations.

Finally, BPR scores 0.294 on MRR, explaining that the first relevant item usually appears earlier in the top 20 recommendation list. More specifically, users can find relevant job opportunities relatively faster but not at the very top of the list. This is a significant improvement in comparison with NameSimilarity score of 0.134, where relevant job opportunities tend to appear later in the list. The 119% increase in performance depicts that BPR ranks relevant items early, enhancing user experience and satisfaction.

In summary, across all metrics BPR outperforms NameSimilarity, establishing it as more effective model to obtain, rank and recommend relevant job opportunities to job

seekers within the top 20 recommendations. Nevertheless, we were quite surprised by the tremendous performance improvements achieved by the BPR model. Taking upon a closer investigation of how BPR and recommendation systems work, it becomes clear that our evaluation was conducted using offline tests on predetermined data (e.g., user interactions). This means that the interactions were already established based on the previous user behavior on the platform. Ideally, we would have conducted an online experiment to monitor the performance of BPR on real time interactions, which would show us a more representative evaluation. The online testing can reveals us how job seekers interact to recommendations in real time, providing useful insight into how well the model adapts to changing behaviors and preferences. Unquestionably, we assume that our BPR model outperform the NameSimilarity model, but on an online experiment we would expect different and maybe smaller performance improvement.

Table 5.1: Performance Metrics for NameSimilarity and BPR Models on Test Set (10%)

Model	Recall@20	Precision@20	NDCG@20	MRR
NameSimilarity	0.195	0.0185	0.123	0.134
BPR	0.325	0.0576	0.232	0.294
% Performance Increased By	66%	211%	88%	119%

5.3 Robustness of BPR

5.3.1 Cross Validation and Threat To Validity

As was mentioned earlier on this research, the hyperparameters for the BPR model were tuned separately before cross-validation, avoiding both the high cost of running both processes together and the demanding execution time. Thus, in order to validate the results and our model, we used BPR model with the best-performing hyperparameters on a 10-fold cross-validation on the training dataset. This process involved splitting the data into 10 parts, training the model on nine of them, and validating it on the remaining one, changing the parts until all are used as validation sets.

The results, shown in Table 5.2, indicate consistent performance of BPR across all folds, with small standard deviation across all metrics. The low standard deviation values means that the model predictions were stable and not vulnerable to specific data splits, revealing a robust and generalized model. However, the significantly lower performance of the BPR model across all metrics on the training set compared to the test set surprises us and raise concerns about the validity of the model. Specifically, selecting hyperparameters based on the performance of BPR on the test set before cross-validation may have led to overfitting. Knowing this potential threat to validity of our model and results, we aim to prove its robustness and generalizability by testing it on a completely new, unseen

dataset, in the next subsection.

Table 5.2: Performance Metrics Across 10 Folds

Fold	MRR	NDCG@20	Precision@20	Recall@20
Fold 0	0.0703	0.0730	0.0129	0.1528
Fold 1	0.0676	0.0711	0.0128	0.1509
Fold 2	0.0672	0.0710	0.0127	0.1500
Fold 3	0.0670	0.0707	0.0124	0.1507
Fold 4	0.0693	0.0731	0.0129	0.1550
Fold 5	0.0679	0.0722	0.0129	0.1540
Fold 6	0.0672	0.0714	0.0128	0.1519
Fold 7	0.0685	0.0716	0.0127	0.1506
Fold 8	0.0686	0.0738	0.0130	0.1568
Fold 9	0.0657	0.0701	0.0126	0.1510
Mean	0.0679	0.0718	0.0128	0.1524
Std	0.0012	0.0011	0.0002	0.0021

5.3.2 Evaluating BPR Model Performance on Unseen Dataset: A Robustness Test

To validate our model, we further tested BPR model on a new dataset called "Dataset2024", which spans from January 1st, 2024, to June 30th, 2024. This dataset inclides the same columns as the "opportunity interactions" dataset and consists of 1,528,070 rows. We split Dataset2024 into 90% for the training set and 10% for the test set, providing 1,375,263 rows for training and 152,807 rows for testing. As a next step, we applied the same pre-processing approach on the training set, training our BPR model on 758,946 rows and testing it on the test set.

The Table 5.3, compares the performance of NameSimilarity and BPR models based on new data called "Dataset2024" as well as depicts the increase in performance of BPR in contrast to the baseline model. Starting with Recall@20, BPR scores 0.370, a 41% improvement over NameSimilarity score at 0.217, showing that the former model finds relevant job opportunities more often within the top 20. Similarly, Precision@20 for BPR is 0.067, reflecting a 69% increase, explaining that BPR returns more accurate recommendations. Also, the NDCG@20 score of 0.249 for BPR, compared to 0.132 for NameSimilarity, shows a 47% improvement, depicting BPR ability to rank relevant items higher in the recommendation list. Lastly, for MRR, BPR achieves a score of 0.307, a 53% increase, explaining that users find the most relevant jobs more quickly.

Table 5.3: Performance Metrics for NameSimilarity and BPR Models on "Dataset2024"

Model	Recall@20	Precision@20	NDCG@20	MRR
NameSimilarity	0.217	0.021	0.132	0.144
BPR	0.370	0.067	0.249	0.307
% Performance Increased By	41%	69%	47%	53%

At first, we suspected potential data leakage and overfitting due to hyperparameters being tuned on the test set. However, the strong performance of the BPR model on the new "Dataset2024" showed its robustness and ability to generalize well on unseen data. This suggests that the good performance of BPR is not just an outcome of overfitting on the original test set, but represents its extended ability to provide relevant recommendations across different datasets.

Chapter 6

Conclusion

The goal of this research was to investigate "How can state-of-art recommendation systems improve personalized job recommendations at Magnet.me by handling the data sparsity problem". Having studied various machine learning models, we chose to proceed with the implementation of Bayesian Personalized Ranking (BPR) as the most suitable approach for handling the data sparsity problem taking also into consideration the time constraints to meet the deadline of the research, as well as the limited computational resources. BPR model with its pairwise ranking approach, placing relevant items higher in the recommendation list, showed important improvements, outperforming the baseline model of Magnet.me across key metrics such as Recall, Precision, NDCG, and MRR. The selected model demonstrated its power to improve personalized recommendations within the recruitment industry where the sparsity problem is quite intense due to limited user interactions. Therefore, Magnet.me company can take advantage of BPR model to improve personalized recommendations, leading to better user experience, increasing satisfaction and decreasing churning.

In addition, this research proved that BPR can be applied also for job recommendations apart from its original use for movie recommendations setting it as a flexible and adaptable model. Furthermore, after achieving satisfying results from BPR model, we believe that the model that would be even more successful when applied to a less sparse dataset with more user-item interactions. In summary, this thesis demonstrates that BPR is an adaptable and flexible approach, making it a valuable tool for different fields that use recommendation systems.

Chapter 7

Discussion

This study reveals the power of Bayesian Personalized Ranking (BPR) to improve personalized job recommendations at Magnet.me, handling the data sparsity issue and outperforming the baseline recommendation system the company currently uses. Using a pairwise ranking approach, BPR efficiently recommends relevant items based on user interactions, depicting its strong performance across metrics such as Recall, Precision, NDCG, and MRR. These results constitute BPR a suitable tool to handle the limited user interactions within the recruitment industry. BPR was successfully adapted from movie to job recommendations indicating its flexibility to other domains that also leverage recommendation systems. In contrast to the company model (NameSimilarity), BPR managed to cope better data sparsity, utilizing the available interactions and generating more relevant job opportunities to job seekers. From business perspective, Magnet.me can take advantage of BPR to improve job recommendations assisting it to boost the job seeker satisfaction, and reduce churning.

The current research faced some limitations due to time and computational restrictions. Specifically, it constrained the values of the hyperparameter that were tested and made it challenging to proceed with the appropriate cross validation trials. Additionally, the thresholds that were established for minimum and maximum user interactions could possibly be different increasing even more the performance of the model and the relevancy of the job recommendations. However, taking different thresholds could harm the general ability of the model if cold start users (e.g., 1-2 interactions) were included or high engaged users were excluded.

In addition, is worth mentioning that for future development , BPR could be combined with a content based approach to improve relevant recommendations with sparse data. This could be possibly applied to cold start users, where there are not interactions yet, building a hybrid recommendation system, that performs good with and without interactions. Lastly, a further development step that should be done is to test BPR at Magnet.me in production in order to receive representative insights based on real time interactions.

In summary, this thesis validates that BPR is an effective model for job recommendations on Magnet.me, creating room for future research to implement its features to other domains that face data sparsity challenges.

References

Bansal, S., Srivastava, A. & Arora, A. (2017). Topic modeling driven content based jobs recommendation engine for recruitment industry. *ScienceDirect*, *122*. doi: https://doi.org/10.1016/j.procs.2017.11.448

Behera1, G., Nain, N. & Soni, R. K. (2024). Integrating user-side information into matrix factorization to address data sparsity of collaborative filtering. *Multimedia Systems*, *30*. doi: https://doi.org/10.1007/s00530-024-01261-8

Das, D., Sahoo, L. & Datta, S. (2017). A survey on recommendation system. *International Journal of Computer Applications*, *160*. doi: https://dx.doi.org/10.5120/ijca2017913081

Fayyaz, Z., Ebrahimian, M., Nawara, D., Ibrahim, A. & Kashef, R. (2020). Recommendation systems: Algorithms, challenges, metrics, and business opportunities. *Applied Sciences (Switzerland)*, *10*. doi: 10.3390/app10217748

Galuzzi, B., Giordani, I., Candelieri, A., Perego, R. & Archetti, F. (2020). Hyperparameter optimization for recommender systems through bayesian optimization. *Computational Management Science*, *17*. doi: 10.1007/s10287-020-00376-3

Hossain, M. R., Timmer, D. & Moya, H. (2021). Machine learning model optimization with hyper-parameter tuning approach. *2021 International Conference on Advanced Engineering, Technology and Applications (ICAETA*. doi: http://jmlr.org/papers/v21/19-805.html

Irani, H., Elahi, F., Fazlali, M., Shahsavari, M. & Farahani, B. (2022). Auto-tuning hyperparameters of sgd matrix factorization-based recommender systems using genetic algorithm. *2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*. doi: 10.1109/COINS54846.2022.9854956

Lee, H.-C., Kim, Y.-S. & Kim, S.-W. (2024). Real-time movie recommendation: Integrating persona-based user modeling with nmf and deep neural networks. *Applied Sciences*, *14*. doi: 10.3390/app14031014

Mhamdi, D., Moulouki, R., Ghoumari, M., Azzouazi, M. & Moussaid, L. (2020). Job recommendation based on job profile clustering and job seeker behavior. *Procedia Computer Science*, *175*. doi: 10.1016/j.procs.2020.07.102

Panteli, A. & Boutsinas, B. (2023). Addressing the cold-start problem in recommender systems based on frequent patterns. *Algorithms*, *16*. doi: 10.3390/a16040182

Rendle, S., Freudenthaler, C., Gantner, Z. & Schmidt-Thieme, L. (2009). Bpr: Bayesian personalized ranking from implicit feedback. *Cornell University*. doi: https://doi.org/10.48550/arXiv.1205.2618

Ricci, F., Rokach, L. & Shapira, B. (2010). Introduction to recommender systems handbook. *Springer Ebook*. doi: https://doi.org/10.1007/978-0-387-85820-3_1

Roy, D. & Dutta, M. (2022). A systematic review and research perspective on recommender systems. *Journal of Big Data*, *9*. Retrieved from `https://api.semanticscholar.org/CorpusID:248508374`

Salah, A., Truong, Q.-T. & Lauw, H. (2020). Cornac: A comparative framework for multimodal recommender systems. *Journal of Machine Learning Research*, *21*. doi: http://jmlr.org/papers/v21/19-805.html

Sarwar, B., Karypis, G., Konstan, J. & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. *Proceedings of ACM World Wide Web Conference*. doi: 10.1145/371920.372071

Shalaby, W., Alaila, B., Korayem, M., Pournajaf, L., Aljadda, K., Quinn, S. & Zadrozny, W. (2017). Help me find a job: A graph-based approach for job recommendation at scale. *Proceedings - 2017 IEEE International Conference on Big Data*, *January*. doi: 10.1109/BigData.2017.8258088

Valcarce, D., Bellogín, A., Parapar, J. & Castells, P. (2020). Assessing ranking metrics in top-n recommendation. *Information Retrieval Journal*, *23*. doi: https://doi.org/10.1007/s10791-020-09377-x

Wang, X., He, X., Wang, M., Feng, F. & Chua, T.-S. (2019). Neural graph collaborative filtering. *arXiv*. doi: 10.1145/3331184.3331267

Wistuba, M., Schilling, N. & Schmidt-Thieme, L. (2015). Hyperparameter search space pruning - a new component for sequential model-based hyperparameter optimization. *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part II*. doi: 10.1007/978-3-319-23525-7_7

Yuan, J., Shalaby, W., Korayem, M., Lin, D., AlJadda, K. & Luo, J. (2016). Solving cold-start problem in large-scale recommendation engines: A deep learning approach. *Cornell University*. doi: https://doi.org/10.48550/arXiv.1611.05480

Appendix

Columns Description of "Opportunity Attributes" Dataset

Exploring the remaining columns of the "opportunity attributes" dataset,in Figure 7.1, we discover that 79.4% of job opportunities are in Dutch ("nl"), followed by 20.5% in English ("en"). Additionally, a small proportion, 0.1%, is showed as "other." Clearly, Dutch language dominates the job marketplace in terms of language.

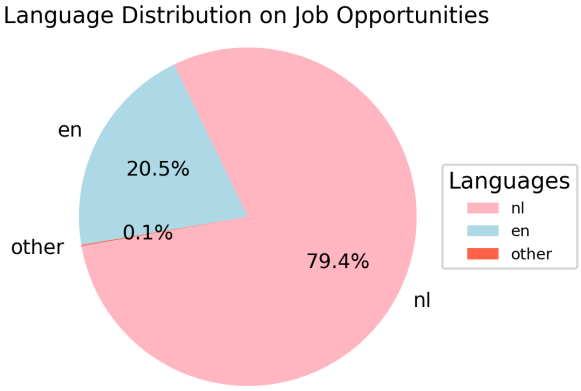

Figure 7.1: Language Distribution on Job Opportunity

In addition, we inspect Figure 7.2 which reveals the top 10 job functions with the most job opportunities. Unquestionably, IT function leads the job market reaching a bit more than one fifth of the total job opportunities, nearly double than that of Engineering at 11.6%. The later is followed by Consulting and Management which come quite close with 10.8% and 10.3% of job opportunities, respectively.

With Finance and Marketing at 9.4% and 8.4%, respectively, they show similar demand, following the top categories. Jobs in legal sector reach 8.0%, just below Marketing, while Human Resources and Project Management are almost equal, with 6.9% and 6.8%, respectively. Finally, General Business receives a 5.4%, being the least favorable out of top 10 job functions. Overall, while IT clearly shows its dominance in the job market, the other job functions reveals a more balanced demand across sectors, offering a wide range of opportunities for job seekers in different industries.

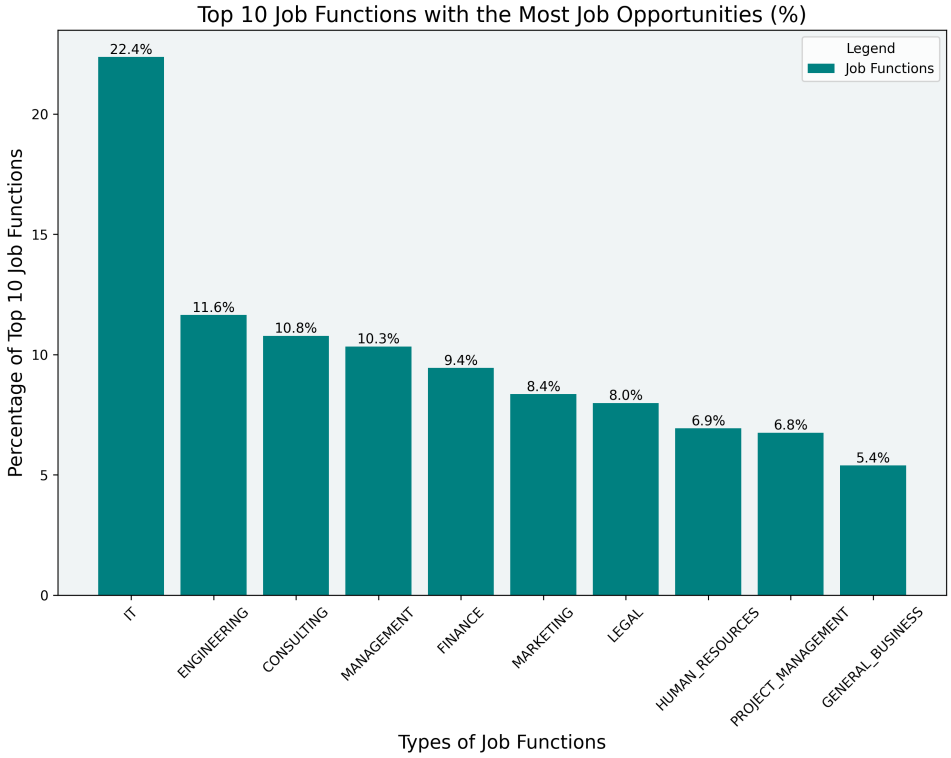

Figure 7.2: Language Distribution on Job Opportunity

As a last column of the second dataset "opportunity attributes" be the "city", we investigate the top 5 cities with the highest percentage of job possibilities in Figure 7.3 Amsterdam has the highest percentage of job opportunities at 20.8% followed by Utrecht which comes in second place with 11.4% of available jobs. Furthermore, third place is taken by Rotterdam at a percentage of 9.0% for the opportunities, followed by The Hague with 7.5%. Lastly, Eindhoven, includes a 3.1% of the total available job opportunities in the market. There is no doubt that one-fifth of the job market is dominated by the capital of the Netherlands, Amsterdam, which offers the most job opportunities within the job market.

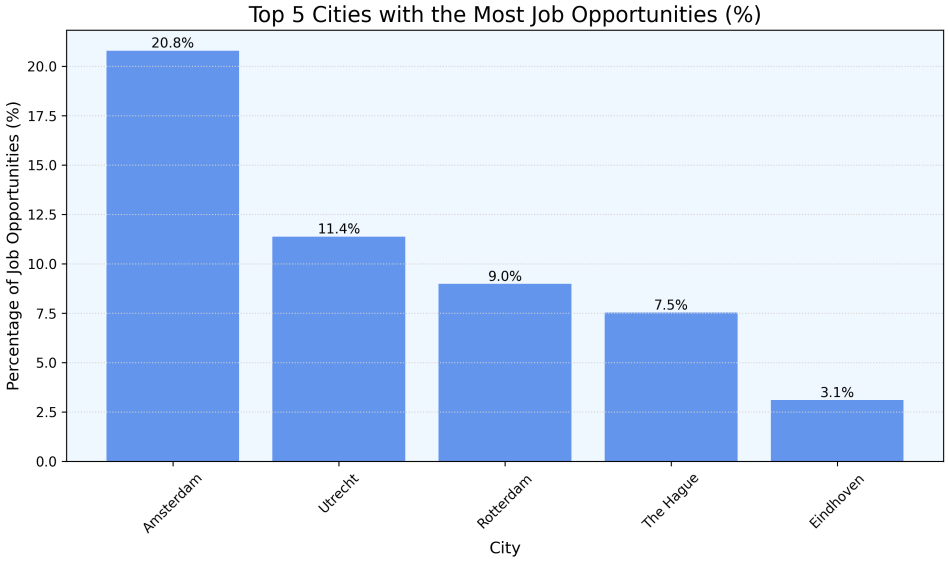

Figure 7.3: Language Distribution on Job Opportunity