Universiteit Leiden

A Personalized Job Ranking System for the Recruitment Industry

MSc Computer Science Thesis

Name:	Stan Eveleens
Student ID:	s3282716
Date:	August 20, 2024
Specialisation:	Data Science
1st supervisor:	Zhaochun Ren
2nd supervisor:	Guus Ramackers

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Contents

1 Acknowledgements

First and foremost, I would like to express my sincere gratitude to **ManpowerGroup** for providing me with the opportunity to conduct this research within their organization. Their support and resources were instrumental in the completion of this thesis.
I would particularly like to thank Eelco Staphorst for giving me this opportunity and connecting me with the Data & Analytics team, as well as answering all my questions and introducing me to the Manpower organization. Special thanks to Jacco van Rijsbergen, who assisted me in integrating into the Data & Analytics team and connecting me with the right people within the organization. Furthermore, I want to extend my thanks to everyone in the Data & Analytics team for making me feel welcome and part of the team. The last few months have been a pleasant experience, and your openness to questions greatly facilitated my thesis progress. I also want to thank Niels Ruitenberg for being the link between me and the recruiters at Manpower. Your excellent and swift communication was invaluable. Thank you all!

Furthermore, I want to extend my gratitude to my supervisor, Dr. Ren, for guiding me throughout this thesis process. I am very grateful for your support and for taking the time to help me. I also want to thank Zihan Wang for his assistance, providing swift responses whenever I needed help. Lastly, I would like to thank Dr. Ramackers for stepping in and assisting me during critical moments. Your help was invaluable, and I deeply appreciate it.

Abstract

The rise of Large Language Models (LLMs) has profoundly transformed the field of recommendation systems, encouraging the development of innovative methods for integrating advanced models. This trend is also evident in the recruitment industry, where new strategies are being pursued to optimize processes and enhance efficiency. Leveraging this advancement, this thesis aims to develop a job prioritization model for the ManpowerGroup organization using the capabilities of a state-of-the-art LLM, Llama3. By utilizing historical placement data from recruiters, this model seeks to identify the most suitable jobs in the pipeline, thereby assisting recruiters in prioritizing their efforts to increase efficiency in their daily activities. Following the TALLrec recommendation framework, my research creates a personalized top 10 ranking for recruiters at Manpower. Through the implementation of three distinct prompt configurations, this study determined that a simplified prompt version was most effective for Manpower's use case. Although recruiters recognized the model's potential, the final results lacked the accuracy required for immediate practical use. Further refinement and data enhancement are necessary to fully realize the model's practical applications.

2 Introduction

In the current recruitment landscape, the advent of artificial intelligence (AI) is becoming increasingly crucial. Its integration into recruitment strategies is not just a trend but a necessity for companies striving to maintain a competitive edge. This urgency is highlighted by the fact that many competitors are already leveraging AI to enhance their recruitment processes [1]. The advantages of employing AI in recruitment are multifaceted, including heightened efficiency, more accurate candidate matching, and enhanced consistency [2]. Furthermore, a growing body of research underscores the positive impact AI has had in this domain, with studies [3–5] indicating substantial improvements in recruitment outcomes due to AI implementation. Conversely, the integration of artificial intelligence into organizational structures can present significant challenges, as evidenced by a Deloitte study from 2018. [6], 72% of organizations recognized the importance of AI, but only 31% were ready to adopt it. This gap highlights the challenge of transitioning to AI-based recruitment. This paper will contribute to this transition, specifically in the field of recruitment.

There is substantial research done about AI in the recruitment field [7, 8] and what kind of impact it has [9]. And this research is also contributing in the same field. However, most existing research primarily focuses on either the job seekers' quest for employment or the employers' efforts to find the perfect candidate for the available job vacancies. In contrast, this study concentrates on the recruiter's perspective, particularly their effort to match the most appropriate job position with their expertise in finding a suitable candidate, focusing on aligning recruiters with job positions based on their personal experience rather than matching candidates to jobs. In the realm of recruitment, a recruiter's core responsibility lies in maximizing the number of successfully filled job vacancies for their organization. The selection of a candidate not only meets the organization's staffing needs but also contributes to its revenue, as the candidate's working hours generate income. The importance of successfully placing candidates in jobs is paramount for recruitment agencies, highlighting the significance of this research, which centers on the subject of job

prioritization.

This thesis will be conducted in cooperation with ManpowerGroup. ManpowerGroup is an outsourcing organization which derives its profitability from undertaking outsourcing projects from clients. In the Netherlands, this organization operates under three distinct labels: Experis, Jefferson Wells, and Manpower. The focus of this research will be on the Manpower label, which specializes in 'blue-collar' roles, typically associated with entry-level positions. Currently, the organization is in the process of modernizing its operations and integrating AI into its daily workflow. Amidst this transition, several aspects of their processes could benefit from enhancement, with job prioritization being a good example. Currently, the organization has no job prioritization system in place, leading to inefficient recruiting. The proposed model will deliver a personalized job prioritization system that helps the organization in creating a more efficient recruitment workflow.
The primary objective of the research in this thesis is to develop and implement an innovative job prioritization system called JobRank: Dynamic Job Prioritization, within the recruitment domain of Manpower. This system aims to revolutionize the existing recruitment processes by introducing a more efficient and data-driven approach to job prioritization, tailored specifically for recruiters. The goal is to enable recruiters to identify and focus on the most critical job vacancies quickly and effectively, thereby optimizing their recruitment efforts.

JobRank will be utilizing historical recruiting information as input. Specifically, this thesis considers every job vacancy where a recruiter has previously successfully matched a candidate and subsequently filled the position. Building upon this historical behavior, JobRank will generate a prioritized ranking of jobs as output. At the top of this ranking, we will find the job opportunity that demonstrates the closest alignment with a recruiter's previous successful placements, as determined by the highest similarity score. Previous research has shown that historical data of a user can reflect on their behavior in the future [10]. In the context of this research, a recruiter's "track record" is suggested to reflect their recruiting competence. Accordingly, the integration of JobRank would prioritize job opportunities within the same category as the recruiter's previous successful placements. When establishing JobRank we face 4 challenges.

- Challenge 1: One of the primary challenges we face is the need for a significant volume of dependable data. Dependable data is clean, accurate, and contains sufficient relevant information without unnecessary high dimensions, ensuring that the created rankings by JobRank, are reliable and effective for accurate analysis. Both the quality and quantity of this data are crucial - it needs to be thorough and precise. If the data is of poor quality or contains too much irrelevant information, it might result in unreliable job rankings. This would compromise the effectiveness of JobRank and, consequently, could negatively impact the credibility and broader impact of our research. Therefore, maintaining the integrity of our data is essential for the success of our study. It plays a direct role in determining the accuracy and practicality of our job prioritization model in an actual work environment. Ultimately, this data must undergo a process of refinement to transform it into a cleaned and readily usable dataset.

- Challenge 2: Another vital aspect of this research involves determining the most

effective features to incorporate. To achieve this objective, we will leverage a substantial dataset comprising numerous job listings. Each of these job listings will possess its unique set of features, some of which will hold greater significance than others. The precise identification of these key features will play a pivotal role in the success of our research and subsequent experiments.

- Challenge 3: An integral component of JobRank efficacy is the reliance on reliable feedback or accurate evaluation. For the algorithm to function efficiently and adaptively, it must receive consistent and accurate feedback from its users. This feedback loop is crucial for the system to learn and evolve, fine-tuning its prioritization capabilities over time. Without accurate feedback, the results may not show the preferred personalized recommendation per recruiter.

- Challenge 4: Leveraging a state-of-the-art LLM like Llama3 also presents prompt creation challenges. Identifying the most suitable prompts to query the LLM and obtaining sufficient results requires extensive testing. This process is crucial for achieving optimal performance in our systems. Without well-crafted prompts, our LLM may underperform.

Addressing the first challenge entails constructing a well-structured data overview. This overview serves as a valuable tool for gaining insights into the data and determining which datasets are pivotal and should be incorporated. Additionally, data validation is imperative, and this can be achieved by engaging with the appropriate individuals within the organization. Once the data has been validated, and the relevant datasets have been selected, the next step involves merging the pertinent data while preserving the integrity of their respective information. The research can proceed with its experiments only when the data meets the required quality standards. Solving the second challenge involves the use of a feature selection method. Two different methods will be conducted, checking for correlations in the dataset and high variations. In addressing the third challenge, a validation concept will be developed. This study will provide a personalized ranking for every recruiter at Manpower. The quality of this ranking will be assessed using a survey, where recruiters will provide feedback on their personalized rankings. This feedback will serve as a validation stage and ensure reliable evaluation. The design of this model is such that it provides precise and actionable feedback for the rankings generated by JobRank. This autonomous refinement process is pivotal, as it enables the system to continually evolve and improve its performance. Lastly, to address the fourth challenge, this thesis will make extensive efforts to generate prompts tailored to Manpower's data, ensuring stable and accurate responses. We will develop and test three different prompt configurations to evaluate their effectiveness and identify the optimal configuration for our use case.
Our proposed framework exhibits notable parallels with contemporary sequential recommender systems, which have gained prominence in recent years, as evidenced by prior research [11, 12]. This research builds upon the foundations laid by the TALLrec framework, originally developed by Keqin Bao and colleagues in 2023 [13]. Our proposed framework is a tailored adaptation of the TALLrec model, incorporating specific modifications to align with our objectives. Central to our approach is the utilization of individual recruiters' historical data concerning their previous job placements. We plan to analyze the characteristics of these successfully placed positions and compare them against all the successfully placed jobs from Manpower. This comparative analysis of features forms the basis for

generating candidate recommendations. Subsequently, these recommendations are placed in a natural language prompt, which is then fed into Llama 3 to create a personal ranking.

To assess the efficacy of our proposed framework, a series of comprehensive experiments will be conducted. The cornerstone of this validation process involves evaluating the rankings generated by our framework through surveys administered to recruiters based on their personalized rankings. These surveys will gather feedback to test the quality and accuracy of the personalized rankings.
To test the importance of prompt creation, we will evaluate three different configurations, each utilizing distinct prompts to generate personalized rankings for each recruiter. These prompts will then be assessed by the recruiters themselves to validate their quality and determine which prompt yields the most accurate personalized ranking.
This research endeavors to contribute to two scientific fields: the application of artificial intelligence in recruitment processes and user behavior-driven recommender systems. It proposes the development of a job prioritization system tailored for an outsourcing company, designed to enhance recruiter efficiency and placement quality. Central to this system is its reliance on the historical performance data of recruiters, enabling the formulation of a personalized job ranking algorithm. Such a personalized approach in leveraging AI for recruitment with the recruiter as a user has not been extensively explored in current literature, marking this study a unique contribution to the field. JobRank is at the forefront of research in investigating the task of Job Prioritization within the context of a recruitment company, employing Language Model (LLM) prompts to effectively establish a ranking system.

3 Related Work

3.1 Artificial Intelligence in recruitment

In the expanding field of artificial intelligence (AI) within recruitment, a significant amount of research has been undertaken. As we've discussed in the introduction section, AI is increasingly becoming a part of many organizations. With substantial growth in the enterprise value of AI applications in recruitment, projected to increase from $107.3 billion in 2025 to a massive $15.7 trillion by 2030 [14]. However, there is a variation in how ready they are to adopt it [5]. For AI to be truly effective, it's crucial that it's user-friendly and that it enhances both efficiency and cost-effectiveness in labor [3].
In the recruitment sector, this particularly affects the hiring process, from job application to candidate placement, significantly impacting the competitive dynamics within recruitment organizations [3]. In the field of recruitment, AI's integration is increasingly widespread and already exerting a significant impact. Its role is seen as an enhancement to recruitment processes, offering potential benefits to many organizations [9]. A 2020 study highlights AI's substantial impact on recruitment but also points out the challenges and risks involved in adopting AI-driven strategies [7]. It raises concerns about employees potentially being targeted by rivals using AI techniques. It also notes privacy issues, especially as public awareness about data use increases, which might reduce the availability of essential data for AI-driven systems. Because AI can fundamentally change the way organizations operate, implementing it comes with limitations [**aibuildingblocks**]. The

implementation of AI comes together with a lot of time and cost. Also, the data being used for these AI systems should be complete and unbiased.
However, AI has a lot of value to add to an organization as well. AI technologies are set to improve access to talent, heightening competitive pressures and compelling organizations and HR managers to adapt by integrating AI into their recruitment strategies [1]. With its primary objective being the reduction of time spend when placing a new candidate [8]. Integrating AI into recruitment processes serves various purposes, including the ability to suggest suitable candidates to an organization. This involves a machine-generated concept known as the Person-Job fit, tailored specifically for each organization. These recommender systems play a pivotal role within the job-seeking domain, wherein the pursuit of talent acquisition is confronted with multiple complexities. The job market has evolved to present formidable challenges in identifying and matching suitable candidates with job openings, necessitating the development of a highly accurate job-matching system. Consequently, there is an increasing interest in research endeavors centered on these systems. This heightened research activity is further complicated by the exponential growth of data availability in the current era [15].
Moreover, Deep Neural Networks (DNNs) have gained a lot of attention in the field of matching people with suitable jobs. DNNs have proven to be very good at understanding and matching natural language, which is important for finding the right job for a person [15]. A specific instance of a Deep Neural Network (DNN), as advanced by Qin et al. in their research in 2021 [16], is formulated with the intent of enhancing the efficiency of matching job candidates with job requirements while mitigating the requirement for extensive human-annotated data. Their model leverages extensive historical job application data, utilizing a Recurrent Neural Network (RNN) to create a word-level semantic representation of both job requirements and applicant experiences. It incorporates four hierarchical ability-aware attention strategies to assess the relevance of job requirements and the contribution of each job experience towards specific ability requirements.
The exploration of the connection between job requirements and applicant experiences is a common theme in research. Examining a user's past behavior can contribute to a more lucid understanding of effective recommendations. As a result, many studies in the field of job recommendation have explored this correlation. Salehi, in his 2013 paper, articulated that the direct comparison of two similar user ratings may not always be the preferred approach [17]. This stems from the fact that different users may harbor distinct intentions and emphasize disparate facets of a product of interest. The paper introduced an innovative personalized recommender system designed to unearth patterns in user preferences, thus enhancing the quality of recommendations.
A recent survey conducted in 2023 highlighted a deficiency in explicit user feedback [18]. In the absence of sufficient feedback, recommendation systems may not operate efficiently. However, by integrating Neural Networks in conjunction with User Behavior Modeling (UBMs), it is possible to compensate for this feedback gap and enhance recommendation systems. The model presented in the aforementioned paper illustrates a system that harnesses sequential patterns derived from users' past behavior. Another paper, from 2022 [19] shows a model that uses both the perspective from a job seeker and an employer (dual-perspective). The paper suggests an effective optimization algorithm with a quadruple-based loss which shows effectiveness in the successful Person-job matching.

3.2 Recommender Systems

Classic Recommender Systems In contemporary research, classic recommender systems can be categorized into four distinct classes: Content-based Recommender Systems (CBR), Collaborative Filtering (CF), Knowledge-based Recommender Systems (KB), and Hybrid Recommender Systems (HRS), as outlined in a review by Smith et al [20]. Content-based recommender systems function by analyzing a user's profile alongside the attributes of the items being considered [21–23]. This methodology ensures that recommendations are specifically tailored to an individual's past preferences, focusing on item characteristics rather than interactions with other users.
Conversely, collaborative filtering is based solely on behavioral data within a matrix [24–27]. This data includes actions such as website clicks or movie ratings, and, in job recommendation scenarios, it might involve recruiters' previous job placements. By examining the collective behaviors of users, collaborative filtering identifies patterns and preferences that may not be apparent from individual profiles alone, thus enabling predictions based on similar users' activities.
Knowledge-based recommender systems aim to gather detailed information about both job requirements and candidate attributes [28–30]. This information is then used to create a predefined job ontology, which aligns job positions with suitable candidates. This approach is especially useful in areas where explicit user preferences are absent, relying on structured knowledge and expert rules to guide recommendations.
Hybrid recommender systems integrate multiple recommendation models to enhance the overall effectiveness of the system [31–34]. By combining different methods, hybrid systems overcome the limitations of individual approaches, providing more robust and accurate recommendations through a diverse array of information sources.
These systems are widely used and there has been extensive research, done. Despite their success, the classical recommender systems did have their limitations. With content-based methods sometimes suffering from overspecialization and where recommendations are also limited to items similar to the user, new items to the user were hard for these systems to rank, the cold-start problem.
However, in the 2010s the research began to shift in a direction of working with deep neural networks (DNNs). This shift was confirmed with the research from Google in 2016 about YouTube recommendations [35]. This paper describes the use of deep neural networks in recommender systems. DNNs often had the ability to learn more complex patterns and representations from a substantial amount of data. With the world evolving with more and more data this was a necessary improvement.

Neural Recommender Systems DNNs accounted for a substantial proportion, approximately 50%, of research contributions in both 2019 and 2020 from a Job Recommender systems review of 2021 [20]. DNNs have demonstrated their potential to augment recommender systems by significantly enhancing scalability and adaptability, thereby facilitating more comprehensive insights into data patterns. Moreover, the advancement of research on DNNs has been notably accelerated by the presence of competition platforms like Kaggle. These platforms catalyze research progress by providing an inclusive environment for individuals to engage in competitive problem-solving and address real-world challenges. This platform achieves this by granting access to diverse datasets, orchestrating machine learning competitions, and nurturing a vibrant community of data-driven

practitioners [36].

The classic techniques mentioned in the previous paragraph were enhanced with DNNs. An article from Ullah et al. applied Deep Neural Collaborative Filtering (DNCF) in educational services [37]. Deep Edu tackles the limitations of collaborative filtering by applying a three-part Deep Neural Network model which consists of an input layer, a multilayered perceptron, and an output layer. Furthermore, an article by Sangeetha et al. combines the advantages of content-based as collaborative filtering approaches in a hybrid approach. The research further extends its model by applying a deep neural network. Also to improve the ability of the model to tackle complexity and to create a privacy-preserving system.

Moreover, the advent of Deep Neural Networks led to further advancements. Building on the foundation of DNNs, Recurrent Neural Networks (RNNs) were developed to handle sequential data, enabling the capture of higher-order dependencies within user sequences [38]. Following RNNs, Graph Neural Networks (GNNs) emerged as advanced techniques for representation learning, designed to learn representations of users and items [39].

DNNs seemed the solution for a long time improving a lot of recommendation systems. While DNNs addressed many limitations of traditional recommender systems, LLMs appear to overcome some of the challenges associated with DNNs. DNNs are often criticized for being "black box" systems, where inputs are processed in a way that is not explained to the end user. This lack of transparency makes it difficult to understand how decisions are made, and DNNs still struggle with fully grasping the nuances of the data.

Recommender Systems with LLM Although neural networks feel as they are still a recent development, they have actually been around for over a decade. Recently, however, the field of recommender systems is witnessing a shift towards a new technology: Large Language Models (LLMs) [40]. This topic will be further explored in the section on Large Language Models within the related work. LLMs offer improvements in these areas by providing better interpretability and a more nuanced understanding of the data. LLMs have the ability to understand language and interpretation and have impressive generalization and reasoning skills. With the rise of ChatGPT [41], there has been extensive research into utilizing Large Language Models (LLMs) for recommender systems.

Recent research has demonstrated that LLMs possess strong language understanding and generalization abilities, making them well-suited for handling diverse data inputs. This versatility is particularly advantageous for recommender systems, where LLMs can generate explainable recommendations. For instance, research by Gao et al. in 2023 introduced Chat-REC, a recommender system enhanced by ChatGPT that leverages LLMs to build conversational recommender systems. Chat-REC converts user-profiles and historical interactions into prompts for more personalized recommendations. Additional recommender systems will be discussed in the following section.

3.3 Feature Selection

Feature selection is a common practice within data science, particularly when dealing with large datasets that can be computationally intensive and detrimental to explainability. Utilizing feature selection techniques allows for the retention of the most important features, thereby telling the same story with the data but in a more concise manner. This approach reduces the dimensionality of the data, improving both efficiency and explainability. There are three primary methods for performing feature selection: unsupervised, supervised, and semi-supervised [42]. Each method presents its own set of advantages and challenges. Supervised methods depend on labeled data, enabling feature selection algorithms to effectively identify discriminative and relevant features that distinguish between different classes [43–45]. Semi-supervised methods, on the other hand, utilize both labeled and unlabeled data. Typically, the model is initially trained using the labeled data and subsequently used to infer labels for the unlabeled data [43, 46]. Unsupervised feature selection, however, does not rely on ground truth labels, which presents a significant challenge [47, 48]. In this approach, features must be evaluated based on various criteria to determine their utility for the dataset. This approach also offers several advantages, such as being unbiased and performing well in scenarios where no prior knowledge is available. They will also reduce the risk of overfitting since it has no real truth, in contrast to supervised feature selection.

Within feature selection, three distinct methods exist: filter methods, wrapper methods, and embedded methods [.] Filter methods rank features based on specific criteria and then select the highest or lowest-ranked features. Examples of filter methods include variation analysis, correlation analysis, and clustering [49]. Firstly, Variation analysis is an important element in feature selection [50]. Variance analysis evaluates the variability of features within the dataset. It identifies features with higher variation under the assumption that features with low variance contribute less to the discriminative power of the model. This method is particularly effective in scenarios where features with minimal variability are unlikely to significantly influence the outcome. For example, in job application systems, variance analysis can help prioritize features that exhibit meaningful variations in job attributes. Correlation analysis on the other hand measures the linear relationship between pairs of features [51]. Filtering out highly correlated features can mitigate multicollinearity issues in certain models, thereby enhancing interpretability and reducing computational complexity. In the context of job application ranking, correlation analysis assists in selecting independent features that collectively provide a comprehensive view of a recruiter's suitability without redundancy. Wrapper methods utilize the intended learning algorithm to evaluate the features, with Recursive Feature Elimination (RFE) serving as a notable example. In RFE, the model is built iteratively, removing the weakest features until a predefined threshold is reached [52]. Embedded methods perform feature selection as an integral part of the model construction process. A prominent example is random forest-based feature selection, which calculates feature importance by measuring the reduction in impurity (Gini importance) each feature contributes across all trees. Features are then ranked based on these scores, identifying the most useful features [53].

Filter methods are often regarded as straightforward and efficient solutions for feature selection. These methods identify features based on metrics such as variation or correlation, offering simplicity and ease of interpretation. In contrast, embedded and wrapper methods are more suitable for complex datasets with substantial amounts of data. While

filter methods excel in mitigating overfitting, embedded and wrapper methods are more susceptible to this issue due to their integration with specific models and extensive search processes [54].
Another way of selecting features is the use of clustering techniques. Clustering algorithms group similar data points into clusters based on their feature similarity. An efficient and widely used clustering technique is k-means [55]. Despite being over 50 years old, its foundational principles are still prevalent in contemporary data analysis. An algorithm built on this principle is the k-prototype algorithm [**kprototype2**, 56]. This algorithm has the ability to cluster mixed datatypes. For example, job features like salary, location, or job title can be clustered together using this algorithm. And to visualize this data certain methods are available. To effectively visualize clustering algorithms, various techniques can be employed. Given that datasets often contain numerous features and clusters, it is essential to use visualization methods that reduce dimensionality for practical presentation. One prominent technique that focuses on transforming high-dimensional data into smaller principal components is Principal Component Analysis (PCA) [57]. This algorithm is efficient and applicable to numerical data, preserving the global structure effectively. The components are ordered by the amount of variance they explain, with the first principal component accounting for the most variation in the dataset. Consequently, changes in this component have the most significant impact on the dataset.
Another useful algorithm is t-SNE, introduced by van der Maaten and Hinton in 2008 [58]. Similar to PCA, t-SNE maps high-dimensional data to a lower-dimensional space. However, t-SNE minimizes the divergence between the similarities of data points in the high-dimensional space and the lower-dimensional embedding, with a focus on preserving local structures rather than global ones. This approach is particularly effective in visualizing clusters, revealing relationships and groups of similar data points. t-SNE is suitable for exploring complex structures and patterns but can be computationally expensive. A more scalable algorithm is UMAP [59]. This method is a nonlinear dimensionality reduction technique also specialized in mapping high to low-dimensional data. UMAP also has a focus on preserving local and global structures of the data. It is specialized in working with both numeric and categorical data types which makes it more versatile than the other two methods.

3.4 Large Language Models

The emergence of ChatGPT has brought Large Language Models (LLMs) into the spotlight, sparking a lot of interest in the academic world [41]. This interest isn't completely new, though. The groundwork was laid about five years ago, with the first paper on GPT (Generative pre-trained Transformer) from 2018 [60]. This landmark publication had a profound impact on the domain of Natural Language Processing (NLP) by introducing innovative concepts. A pivotal contribution was the transition from conventional Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) to the adoption of transformer-based architectures, signaling a paradigm shift in NLP methodology.
Another key player in the LLM field is BERT (Bidirectional Encoder Representations from Transformers), developed by Google [61]. BERT is often seen as the first "real" LLM used in NLP. What made BERT stand out was its ability to understand the context of words in both directions - forward and backward - using a transformer model. This was a big leap forward. Also, BERT is adaptable and can be fine-tuned easily to become a

top-notch model for a wide range of NLP tasks, which is why it's considered so important in the ongoing development of LLMs.

In the domain of job ranking and recommendations, the application of Large Language Models (LLMs) has been groundbreaking. Among the prominent research in this field, Zheng et al.'s 2023 paper [62] stands out with its innovative model GIRL (GeneratIve job Recommendation based on Large language models), leveraging LLMs to revolutionize job matchmaking. Distinguishing itself from traditional systems, GIRL generates personalized job descriptions directly from job seekers' CVs, shifting from the norm of using pre-set candidate lists. This approach enhances the system's transparency and user-centricity.
GIRL employs a sophisticated three-step training process involving Supervised Fine-Tuning, Reward Model Training, and Reinforcement Learning from Recruiter Feedback using Proximal Policy Optimization. These steps enable the model to align job descriptions with both job seeker preferences and market needs, significantly improving the adaptability and effectiveness of job recommendations. This method represents a significant leap in making job recommendations more personalized and comprehensive.
Another sophisticated model is TallRec [13]. The paper proposes a fine-tuned LLM with as input a historical sequence of a user's behavior and as output a simple "yes" or "no" recommendation. This paper addresses the challenge of aligning Large Language Models (LLMs) with recommendation tasks, identifying a critical gap between conventional LLM training and the specific needs of recommendation systems. TALLRec introduces an innovative framework for tuning LLMs efficiently with recommendation data. It stands out for its ability to enhance LLMs' recommendation capabilities in various domains like movies and books, even with minimal datasets. The framework is notable for its computational efficiency, requiring fewer resources for fine-tuning, and demonstrates robust cross-domain generalization, a vital attribute for dynamic recommendation environments.
The PALR framework, as discussed in its respective paper, is tailored to integrate user history behavior with Large Language Models (LLMs) to craft user-specific recommendations [63]. It employs a two-pronged strategy: utilizing user/item interactions for candidate retrieval, followed by an LLM-based model for generating recommendations. This approach involves translating user behavior into natural language prompts to better fit the recommendation scenario.
Another algorithm, LlamaRec, introduces a two-stage recommendation model using LLMs for ranking. It addresses the limitations of traditional recommendation methods, which often depend on pre-trained knowledge and are hindered by slow inference from autoregressive generation [64]. LlamaRec employs sequential recommenders for efficient candidate retrieval, informed by user interaction history, thereby enhancing the recommendation process's efficiency and accuracy.
To establish certain models, papers are using already pre-trained datasets to discover new ways to implement LLMs into recommendation systems. Such a model is M6, a pre-trained multi model language dataset. This dataset consists of 1.9TB images and 292GB texts on a wide variety of domains. A paper by Cui et al. utilizes this model and creates a model called M6-Rec [65]. This model leverages on the fact that M6 is already pre-trained for sample-efficient downstream adaptation. Applying state-of-the-art techniques like late interaction, early exiting and parameter sharing to reduce inference and model size. This technique, of leveraging an already pre-trained model, is called transfer learning. Also applied in the T5 paper [66]. This paper introduces a unified framework that converts all

text-based language problems into a text-to-text format.
In the context of Large Language Models (LLMs), certain limitations have been observed, such as the tendency to exhibit unintended behaviors or fabricate facts, contradicting their intended supportive role for humans [67]. These models should prioritize being helpful, harmless, and honest. Another limitation is the deficiency of human feedback in certain scenarios. Addressing this problem, RankGPT offers a noteworthy solution [68]. This study investigates the use of LLMs, specifically GPT-4 and ChatGPT, for relevance ranking in Information Retrieval, effectively bridging the gap between the models' pre-training objectives and their application in direct passage ranking. The findings underscore the feasibility of employing well-instructed LLMs as effective ranking agents.

However, RankGPT does have its potential issues since it does not leverage labeled data when available and they could also be inefficient due to their lack of parallel scoring and slow multi-pass decoding. Tackling this issue is RankLLaMA [69]. RankLLaMA is advocating for the refinement of large language models (LLMs) as retrievers and rerankers to surpass these limitations, thereby optimizing their integration within multi-stage processing pipelines. Furthermore, the use of a 'black-box' system like GPT-4 in the RankGPT paper can also raise its own challenges. The use of this system is not ideal for academic researches because of the cost constraints and access limitations it has [70].
This thesis aims to integrate and enhance the methodologies outlined in two seminal papers, creating a novel and dynamic framework tailored for recruitment processes. The cornerstone of this endeavor is the TALLrec framework, as delineated in [13]. This framework will form the foundational architecture for a sophisticated ranking system, specifically engineered to augment the efficiency and precision of recruitment professionals at Manpower.
A key innovation of this research lies in its shift of focus. Unlike traditional approaches that concentrate on identifying the most suitable candidates for a position, this study emphasizes the strategic selection of job placements. By empowering recruiters at Manpower with the ability to prioritize certain job openings, the thesis posits a significant improvement in their operational efficiency. This, in turn, is anticipated to yield substantial benefits for the organization at large. The synergy between advanced data science techniques and recruitment processes explored in this thesis represents a novel approach in the field, offering a unique perspective on the dynamics between recruiters and job allocation.

4 Preliminaries

In this section, we introduce the problem formulation of the job prioritization system. We first introduce our problem formulation and at the end discuss the preliminary knowledge. All the equations that are used are summarized in the glossary table 11 shown in the appendix.

4.1 Problem Formulation

Our research is based on two sets, i.e., a set of users and a set of items. In our research, we will refer to the users as "*recruiters*" and the items as "*jobs*". The recruiter's set represents the past successful placements by a recruiter and the jobs set represents all the successfully

placed jobs in the Manpower organization. Let R and J denote the recruiters and jobs dictionaries respectively. Let R be a collection of dictionaries, one for each recruiter u, shown in equation 1. For a given recruiter u, their dictionary R_u contains jobs they have successfully placed, denoted as r_t for the t-th item. Each job $r_t \in R_u$ is characterized by a set of features $\{F_{t,1}, F_{t,2}, \ldots, F_{t,m_t}\}$, where m_t is the number of features for the t-th item.

$$\begin{aligned} R_u = \{&r_1 : \{F_{1,1}, F_{1,2}, \ldots, F_{1,m_1}\}, \\ &r_2 : \{F_{2,1}, F_{2,2}, \ldots, F_{2,m_2}\}, \\ &\ldots, \\ &r_n : \{F_{n,1}, F_{n,2}, \ldots, F_{n,m_n}\}\} \end{aligned} \tag{1}$$

Similarly, let J be a dictionary representing all the successfully placed jobs at Manpower, shown in equation 2, where each element j_i is a job characterized by a set of features $\{F_{i,1}, F_{i,2}, \ldots, F_{i,m_i}\}$, and m_i denotes the number of features for the i-th job.

$$\begin{aligned} J = \{&j_1 : \{F_{1,1}, F_{1,2}, \ldots, F_{1,m_1}\}, \\ &j_2 : \{F_{2,1}, F_{2,2}, \ldots, F_{2,m_2}\}, \\ &\ldots, \\ &j_n : \{F_{n,1}, F_{n,2}, \ldots, F_{n,m_n}\}\} \end{aligned} \tag{2}$$

In both R_u and J, n represents the total number of jobs, and m_i represents the number of features for each specific job i. This notation clarifies the association between jobs, their features, and the respective dictionaries for recruiters and all the successfully placed jobs. Using the available datasets J and R a natural language prompt, or Query Q, is created. This prompt combines the two datasets into text input where the historical sequence of successfully placed jobs and the total amount of placed jobs will be put into a Query and their relevance will be ranked. The instruction input for the prompt can be formally written as,

$$\max_{\phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{l=1}^{|y|} \log(P_\phi(y_l|x, y_{<l})) \tag{3}$$

where x and y represent the Instruction Input and Instruction Output respectively. y_t is the $t-$th token of the y, $y < t$ represents the tokens before y_t, ϕ is the original parameters of M, and $\mathcal{Z}$ is the training set [13]. The output of the TALLrec prompt consists of a list of recommended jobs to the recruiter. This list I will further be used for ranking.

List I contains a job l, which includes the JDCO code [71]. JDCO, or Job Digger Code, is generated by Job Digger, a third-party scraping tool utilized by Manpower for vacancy texts. This code serves as a collective identifier for various job roles. For instance, a position at the PostNL sorting center would be categorized under the collective name "postsorteerder," encompassing jobs for individuals working both evening and day shifts. This collective naming convention facilitates effective job categorization.
In addition to this code, the corporation associated with the job is also specified. For example, a "Administratief medewerker" at KLM may represent a different role than a "Administratief medewerker" at bol.com. In this paper, each job is identified by its JDCO code and the corporation together. Furthermore, each job comprises multiple features,

which are also included in list I. List I will then be processed by a ranking prompt. Resulting in a final top 10 ranking per recruiter.

4.2 Recruitment at Manpower

Within the organization of Manpower there is a pipeline of successfully placed job positions stored in J. These jobs $j_1, j_2, ..., j_n$ are job vacancies that a recruiter successfully placed and where Manpower successfully found a candidate for and generated revenue from. A recruiter spends most of his/her time trying to fill these positions with a suitable candidate. He/she does this by contacting already known candidates or tries to recruit them via other channels like LinkedIn [72]. A successful job placement by a recruiter $r_1, r_2, ..., r_{L_r}$ is defined as the process where a recruiter contacts a candidate, who then engages with a client and completes the recruitment process. This process is considered successful when the client decides to hire the candidate, leading to a contract being signed. Once the candidate signs the contract, it is officially recognized as a successful placement. These placements are stored in a database R, serving as a track record for the recruiter. Recruiters often specialize in certain types of jobs. For instance, a recruiter might focus on positions that require minimal recruitment effort, such as a "Magazijnmedewerker at KLM," where a candidate might be selected after just one interview. In theory, does J consist of all the placed jobs within the Manpower organization and R_u is all the successfully placed jobs by recruiter u. However, the allocation of both these datasets is different which could result in a mismatch between jobs. This could be a potential problem when trying to rank the jobs from dataset J.

4.3 TALLrec

The TALLRec framework enhances a LLM, denoted as $\mathcal{M}$, for recommendation systems [13], incorporating two main tuning stages: alpaca tuning, which uses Alpaca's self-instruct data ([73]) to improve LLM generalization, and rec-tuning, tailored for recommendation tasks. Tuning is a crucial technique to train LLMs with human annotated instructions and responses. It generally consists of four steps, the first task is the instruction which is created using natural language, detailing the task's objectives and specific solutions. Second, the task's input and output are formulated in natural language. Third, the task instruction is combined with the input to create the "Instruction Input", and the output is used as the "Instruction Output" for each tuning sample. Lastly, the instruction tuning is conducted on LLMs using the formatted pairs of "Instruction Input" and "Instruction Output". Formally,

$$\max_{\phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{l=1}^{|y|} \log(P_\phi(y_l|x, y_{<l})) \tag{4}$$

where x and y represent the Instruction Input and Instruction Output respectively. y_t is the $t-$th token of the y, $y < t$ represents the tokens before y_t, ϕ is the original parameters of $\mathcal{M}$, and $\mathcal{Z}$ is the training set.

Furthermore, TALLrec capitalizes on the capabilities of Large Language Models (LLMs) to construct a Large Recommendation Language Model (LRLM). The primary function

of this LRLM is to ascertain the suitability of new items for a user, effectively predicting optimal recommendations. This is achieved through the process of rec-tuning, which adeptly transforms recommendation data into a format compatible with instruction tuning methodologies.
Subsequent to the completion of the Alpaca tuning stage, the LLM transitions into the Rec-tuning phase. This sequential approach to tuning is designed to incrementally adapt the LLM to our specific research needs, ensuring a high degree of task-specific performance enhancement. The outcome of this stage will be a LLM that is better suitable to handle the nuances of a recommendation scenario. The goal is to predict the optimal recommendations based on a users history and item features. For the task instruction TALLrec will formulate a natural language prompt instructing the LLM to categorize the new item by answering "Yes" or "No".

Because directly tuning an LLM is very computationally intensive and time consuming TALLrec uses Lightweight tuning. This method is called LoRA [74]. This method "freezes" the original parameters of the model and adds some smaller components that are easier to adjust. These components are special matrices added to each layer of the model's architecture that can be updated to improve the model's performance. By doing this, the model is enhanced with new information efficiently, without having to overhaul the entire set of parameters. Shown in equation 5 where θ are the LoRA parameters and we only update these.

$$\max_{\theta} \sum_{(x,y)\in Z} \sum_{t=1}^{|y|} \log(P_{\theta+e}(y_t|x, y_{<t})) \tag{5}$$

Where θ is the LoRa parameters and we only update these during the training process. Furthermore, x and y represent the Instruction Input and Instruction Output respectively. y_t is the $t-$th token of the y, $y < t$ represents the tokens before y_t and $\mathcal{Z}$ is the training set. Moreover, TALLRec utilizes LLMs-LLaMA ([75]) for its experiments, addressing the complexities and data security concerns in using different LLMs and third-party APIs in real-world recommendation environments.

4.4 RankGPT

RankGPT is a ranking agent made possible by ChatGPT [41, 68, 76]. In their paper, Sun et al discusses how ChatGPT and GPT-4 can be used for relevance ranking in Information Retrieval. Their experiments reveal that, if properly instructed, the LLMs can deliver superior results compared to state-of-the-art supervised methods. The ranking done by RankGPT, ranks a passage based on the relevance to the query. Leveraging ChatGPT and GPT-4's text comprehension capabilities, a re-ranking of passages can be created, with the foremost emphasis placed on raising the most "relevant" passage to the top. This process is executed by generating a query that incorporates the log-probability of the query itself. Formally, given query q and passage p_i, their relevance score is s_i is calculated as $s_i = f_\theta(q, p_i)$ where θ are the parameters to calculate the relevance score s_i. For example, a passage is presented alongside a corresponding question or query. Subsequently, a relevance score is computed, essentially addressing the question: "To what extent does

the passage address the query?" This systematic framework is then extended to multiple passages, where ChatGPT or GPT-4 are tasked with ranking them in accordance with their relevance to the given query.
Due to token limitations of LLM, RankGPT uses a sliding window strategy. In this strategy, the LLMs first rank passages from $(M - w)$ to M, then slide the window in steps of step size s and re-rank passages from $(M - w - s)$ to $(M - s)$. This process continues until all passages have been re-ranked, allowing us to work around token limitations and efficiently rank a larger set of passages.
Furthermore, because ChatGPT and GPT-4 are both very expensive to deploy, RankGPT uses a novel permutation distillation method to distill the passage re-ranking capability into a specialized model. It directly uses the model-generated permutation as the target without introducing bias.
The training objective of RankGPT can formally be denoted: suppose RankGPT has a query q and M passages $(p_1, \ldots, p_M)$ retrieved by BM25 ($M = 20$ in RankGPTs implementation). By leveraging ChatGPT for instructional permutation generation, RankGPT ranks these passages as $D = (d_1, \ldots, d_M)$, where d_i signifies the ranking of passage p_i among the entirety. To determine the relevance score s_i for each query-passage pair (q, p_i), RankGPT utilize a cross-encoder model f_θ, which calculates s_i. Employing the rankings generated by ChatGPT, RankGPT adopts the RankNet loss function to refine their model by considering pairwise comparisons among the passages, encapsulated by the equation:

$$\mathcal{L}_{\text{RankNet}} = \sum_{i=1}^{M} \sum_{j=1}^{M} \mathbb{1}_{d_i < d_j} \log(1 + \exp(s_i - s_j)) \tag{6}$$

This approach, grounded in the RankNet pairwise loss methodology, enables an efficient evaluation of the model's capability to accurately sequence the passages, taking into account $M(M-1)/2$ potential pairs derived from the rankings provided by ChatGPT.
Furthermore, RankGPT considers two model structures. The first is BERT-like model which utilizes a cross-encoder model [77] based on DeBERTa-large. It concatenates the query and apssage with a [SEP] token and estimates relevance using the representation of the [CLS] token. The other is a GPT-like model which utilizes the LLaMA-7b [75] with a zero-shot relevance generation instruction. It classifies the query and passage as relevance or irrelevance by generating a relevance token. The relevance score is then defined as the generation probability of the relevance token.

5 Data

Firstly, we will discuss the data that is used for this research. Sourced from Manpowergroup, it consists of two main sources. The primary and more extensive segment is obtained from the Bullhorn platform, while the secondary portion originates from MySolution. These platforms are specialized software applications tailored for the recruitment and outsourcing domains, facilitating the storage of relevant data.

5.0.1 Bullhorn

The data from Bullhorn is used by Manpowergroup as first stage of the recruitment. This data saves the jobs that are outgoing and still need to be filled upon the point where

there is an actual candidate available for the job. So mostly the recruitment process itself is registered. There are multiple datasets in Bullhorn but for this research, we will only make use of these: Joborder, Placement, Jobspecialties, Specialties, Jobsubmission and Corporation. All these datasets are linked together with their own unique IDs.

Joborder contains a row for each job that is either currently an open vacancy or has already been filled. This dataset will serve as our baseline, as we aim to create a ranking based on the jobs at Manpower. Each column represents a feature of a job, such as the location of the job or the start date for the applicant.
Corporation contains all the information about the corporations associated with the job vacancies. Each row represents a unique corporation, with columns detailing the features of the corporation. These features include the number of employees, location, and number of offices.
Placement contains the jobs that have actually been filled. These jobs are linked to an applicant who is ready to start working. The rows in this dataset also represent jobs, with columns detailing various features. These features include, for example, the type of employment (temporary or permanent) and the number of hours per week.
Jobspecialties is a linking dataset that maps the specialties dataset to the Joborder dataset. It contains the Specialty ID and the Joborder ID.
Specialties is a dataset that contains information on job specialties, comprising 4,566 different categories that represent various job types. For example, all jobs related to creating packages are grouped under the "Packaging" category. This category is identified by a JDCO code, that helps organize and categorize different types of jobs within the dataset.
Jobsubmission contains the IDs of the jobs and the linked candidates who have been placed. This dataset is used to verify if a job has actually been filled by examining the "status" column. Since our research focuses solely on placed jobs, we use this dataset to filter them accordingly.

The datasets are cleaned and eventually merged and stored into the dataset "Bullhorn". The cleaning procedure is summarized in the "Cleaning" section.

5.0.2 MySolution

In Mysolution, the data of applicants who are actually working on a placed job is recorded. This includes their registered hours and the revenue generated for Manpower. In this dataset, the applicants are referred to as "Resources" and the actual jobs are called "Projects".

Project resource is a merged dataset containing information about both a project and a resource. Each row represents a link between a project and a resource, with columns detailing the features of this link. For example, the columns include the number of hours the resource works on the project, the type of contract, and the start and end dates.
HLRE stands for Hour Ledger Revenue Entry and contains the recorded hours a resource has worked on a project. Each row represents a link between a project and a resource, along with a quantity. This quantity can represent hours worked or the number of days of sickness. To simplify, we will focus solely on the number of hours worked. For each

project, the dataset stores the total hours spent on the project and the revenue generated for Manpower.
Project has all the details from a project similar to that of the Joborder dataset in Bullhorn. The rows are the different projects and the columns contain features such as job title, description of the job and address.

These datasets are eventually merged into one big dataset called "Mysolution".

5.0.3 Historical Interaction

Recruiters History
For our research, we will also analyze previously placed jobs by recruiters. This "track record" or history of placements is essential for our personalized job ranking system. We gather this data from the data warehouse of Manpower. This data is stored in a dashboard and comes from the Bullhorn system. However, this dashboard needs to be updated manually which could result in data being outdated or not present. Each row in the dataset contains a placement made by a specific recruiter, along with the candidate's name and the job ID. This ID can then be linked with the job order dataset to retrieve the corresponding job features. Ultimately, we will have a dataset of jobs placed by recruiters along with their features. The cleaning process for these features is the same as for the other datasets and is explained further in the Cleaning section.
Together with the JDCO code, we also retrieved the ISCO code for each job [78]. The ISCO code is a standardized classification system developed by the International Labour Organization to categorize occupations globally. It facilitates the comparison and analysis of labor market data by providing a common framework for different jobs and skills. It has four levels: major groups, sub-major groups, minor groups, and unit groups, which in our dataset are called Isco niveau 1-4. These levels categorize the different jobs, enabling us to compare similar jobs, which is vital for our research. For example, a shop employee is categorized the same as an employee in a medical shop under the category "Dienstverlenend personeel en verkopers." This categorization allows us to distinguish between similar jobs and categorize them accordingly in our research. The ISCO code is linked with the JDCO code and ultimately linked to each job to include them in the overall dataset.

5.0.4 Cleaning

Before processing the raw data, we will undertake a thorough data-cleaning procedure. We will import the aforementioned datasets and initiate the cleaning process. Given the large size of these datasets, each containing on average more than 100 features, our first step will be to reduce the amount of noisy data by removing features with more than 10% missing values (NaN) or features that consist solely of a single unique value. Additionally, we will leverage domain knowledge to eliminate any other irrelevant features. For instance, the job order dataset contained ID features that were not relevant to our analysis, and the Project dataset included duplicate features already present in the job order dataset. Utilizing this approach, we reduced the datasets to approximately 30-40 features per row. During this process, we also addressed inconsistencies within the datasets. For example, in the job order dataset, we standardized the capitalization of province names and generalized the education level to address inconsistencies.

Furthermore, we enriched the job order dataset by creating additional features. We added the number of open job vacancies in each province to assess job demand and the scarcity of job vacancies in different areas. Additionally, we included age group data for each province to estimate the number of people suitable for employment. This allows us to identify regions where the supply of potential candidates is high, which can impact the difficulty of filling job vacancies.
Upon completing the cleaning processes, we attempted to match the datasets. This involved initially combining all of the Bullhorn datasets with each other and with the Mysolution dataset. This process presented some challenges due to certain mismatches. For example, some placed jobs found in the Job Order dataset were not present in the Project dataset. Another issue was that the HLRE dataset did not record the hours worked for certain placed candidates. Unfortunately, these mismatches could not be resolved, leading to some data loss. Ultimately, we combined the datasets into one comprehensive dataset containing both job (J) and recruiter (R) information. We then filtered out the rows with the recruiters' placements to generate the R dataset and divide them into two separate sets.

6 Methods

In this section, we will discuss the different methods used in this research. We will discuss which methods are used to conduct the experiments and how we will achieve the results. Furthermore, we will discuss the proposed framework of our model and the different methods we will use to test and validate the model.

6.1 Model Overview

We propose a user-specific recommendation model designed to rank jobs within the Manpower organization. The primary objective of this model is to accurately create a ranking per recruiter based on their previous interactions, producing a prioritized list where the most relevant job occupies the top position. The model is structured into four distinct stages, each contributing to the final ranking. The framework is shown in Figure 1.

- **Stage 1: Feature engineering** This stage serves as the data preparation stage. The primary goal is to create a final dataset that is optimally suited for the subsequent stages of the research. The input of this stage will be the raw dataset of Manpower. During this stage, we will analyze the data and clean it by removing outliers and inconsistencies. Ultimately, from this stage, we will develop two datasets: one, labeled J, which includes all the placed jobs in the Manpower organization, and a second, personalized dataset, R, which contains all the jobs placed by an individual recruiter u.

- **Stage 2: Recommendation** In this stage, the two datasets generated in the preceding stage are employed as inputs. From these inputs, three distinct prompts are formulated (C1, C2, C3). Utilizing the default TALLrec model, these prompts undergo tuning and testing on the open-source model Llama3. The result is three separate lists of recommended jobs, each personalized for individual recruiters. These prompts, serving as distinct configurations, will subsequently be evaluated in the validation stage to ascertain the most effective configuration.

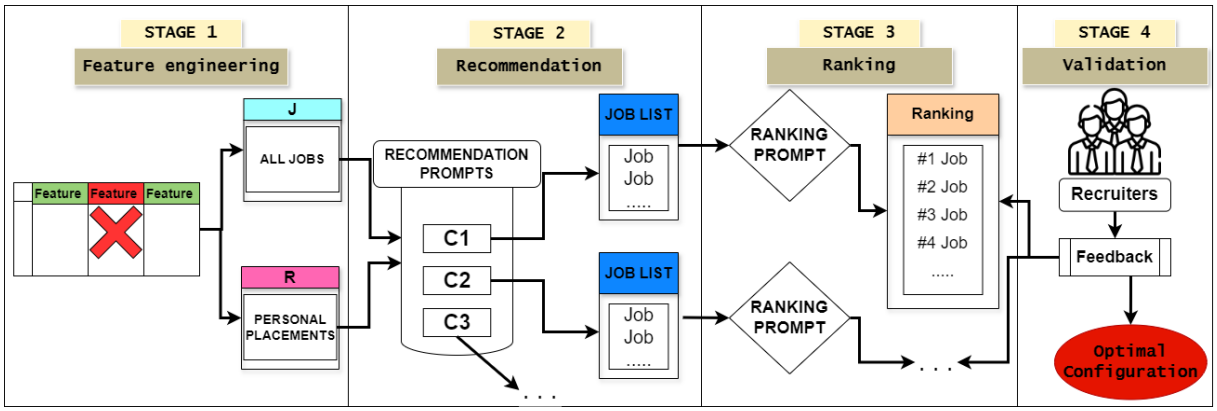


Figure 1: Overview of the proposed framework. Where J is the dataset of all placed jobs, and R the set of all successfully placed jobs by the recruiter.

- **Stage 3: Ranking** Upon creation of the recommended lists for each configuration, they will be ranked by again utilizing Llama3. A ranking prompt is created and the jobs in the job list are given a similarity score from 0 to 100. This process is designed to ensure that the most relevant jobs are given the highest score. The outcome of this stage is a top 10 ranking for each configuration for each recruiter, which will be subject to validation in the next phase of the study.

- **Stage 4: Validation** In this final stage, the effectiveness of the rankings will be evaluated through direct feedback. Recruiters will review the personal rankings generated in the previous stage based on their own job placement history. Each recruiter will assess three different configurations and choose the most effective one. This feedback process is crucial to validate the results and determine if our model improves their daily workflow, thereby directly impacting the overall assessment of the research project. This step ensures that the model's practical benefits are thoroughly evaluated by end-users.

6.2 Feature selection

To extract the features which would be most beneficial for our model we will be using several methods. The dataset we will be using has a lot of different features with both numerical and categorical categories. Due to the diverse range of features and their varying types, numerous options exist for selecting the most appropriate features. Working with large datasets can be computationally intensive, making it essential to focus on explainability and efficiency. Consequently, the aspect of dimensionality reduction is also considered to enhance the overall performance. Therefore, we choose a combination of methods to eventually select the most important features for our datasets.

Initially, we will eliminate features guided by experts or domain knowledge. Domain knowledge is obtained by consulting experts familiar with the datasets. Within the organization, several data experts are specialized in Bullhorn or MySolution data and posses comprehensive knowledge of all the features in the dataset. Given the extensive nature of the data and the possibility of multiple users manipulating it, maintaining a clear

understanding of each element is essential. By leveraging the expertise of these professionals, certain features are identified as unnecessary, such as IDs or internal metrics that are relevant only to the organization, or data that is incomplete and therefore unusable. This process entails consulting experts within the Manpower organization to assess the data and subsequently discarding redundant features or those lacking meaningful content. These features frequently contain invalid or improperly filled data, rendering them unsuitable for analysis. This phase involves the straightforward removal of features prior to proceeding with any subsequent feature analysis.

Our subsequent phase involves conducting an Exploratory Data Analysis (EDA). This critical step allows us to comprehensively comprehend the dataset at hand, recognizing certain patterns, outliers, and other anomalies. Although a substantial part of our EDA has been carried out during the data cleaning phase discussed in the previous section, we will continue to investigate further in this stage. Specifically, we will undertake correlation and variance analyses on the remaining features. A correlation analysis involves the creation of a correlation matrix, facilitating the understanding of linear relationships between pairs of continuous variables. High correlation coefficients indicate that variables move in the same direction and are nearly equivalent, suggesting redundancy. In parallel, the variance analysis aims to identify features with minimal variability, indicating that they are essentially constant and contribute negligible information. Removing such features will streamline the dataset, reducing its dimensionality while retaining meaningful features. This reduction not only enhances the performance of our model but also ensures the dataset is optimized for analysis.Further we will also analyse the explained variance components. We will measure how much of the variance in the data is explained by the remaining features. Potentially this will reduce our dimension even further by removing certain components that do not help with explaining the data. We do have to be careful with removing certain features. Since the LLM would benefit from as much information as possible to compare job applications. Where an algorithm might seem to neglect a feature, the LLM could potentially benefit. We have to find the sweet spot in the amount of features.

Our next stage involves conducting a cluster analysis, a method aimed at finding natural groupings within the data that were not immediately visible. This process proves particularly useful in feature engineering, especially when dealing with datasets of high dimensionality. Through this method, we spotted patterns and figure out which features contribute most significantly to these patterns. The tool we will employ for this task is the k-prototype algorithm [56], which builds upon the foundation of the well-known k-means algorithm, extending its applicability to categorical data. Similar to k-means, the objective of this algorithm is to partition the dataset into k clusters, with each observation assigned to the cluster whose centroid is closest. Initially, the algorithm initializes the k cluster centers or prototypes. Subsequently, each data point is assigned to one of these initialized clusters, with distance calculated using both Euclidean distance for numerical data and matching dissimilarity measures for categorical data. Following this assignment, the algorithm updates the cluster centers, computing the mean value for numerical features and the mode for categorical features. This iterative process continues until the clusters stabilize, indicating convergence. This hybridized approach enables the k-prototype algorithm to effectively manage mixed types of data. Before applying the k-

prototype algorithm, categorical data must be encoded. We will evaluate three encoding methods: binary, ordinal, and frequency encoding.

Binary Encoding: This method maps categorical variables onto binary numbers. Each category is assigned a unique integer, which is then converted into binary. This approach is particularly effective for data with high cardinality. Binary encoding begins by identifying all unique categories within a categorical feature and assigning each a unique integer code. For example, if there are 20 unique categories, they might be coded from 1 to 20. These integers are then converted to binary form, where, for instance, 1 becomes *01* and 2 becomes *10*. This conversion creates new columns based on the number of bits needed to represent the integer values, which can increase the number of columns in the dataset. However, binary encoding is more compact than one-hot encoding, which creates a separate column for each category. Unlike ordinal and frequency encoding, binary encoding does not imply any order among categories; each category is treated as having equal significance, avoiding unintended assumptions. However, binary encoding can be less interpretable than ordinal or frequency encoding due to the ambiguity of binary columns and has a risk of overfitting.

Ordinal Encoding: Ordinal encoding assigns a unique integer to each category based on a meaningful order or ranking, preserving the sequence of categories. For example, in the Dutch education system, "MBO" represents a lower education level than "HBO", so "MBO" would be assigned a lower integer value than "HBO". This encoding is effective when categories have an inherent order. Compared to Frequency and Binary encoding, ordinal encoding retains important ranking information without adding extra dimensions, making it both simple and compact. However, if used inappropriately, ordinal encoding may introduce assumptions that could negatively impact data quality.

Frequency Encoding: In this method, each category is assigned a numerical value based on its frequency in the dataset. This approach is useful when the frequency of a category's occurrence is significant, with higher numerical values assigned to more frequently occurring categories. For instance, if the education level "HBO" appears 130 times in the dataset, it would be assigned the value 130, reflecting its popularity. Frequency encoding is effective for datasets with high cardinality, as it can provide a unique value for each category based on its occurrence. However, it may struggle to distinguish categories that occur infrequently, potentially leading to issues with assigning unique values. Additionally, frequency encoding can introduce bias by favoring categories with higher numerical values, which represent higher frequencies.

We will run the k-prototype algorithm with the three encoded methods and compare their effectiveness by their average silhouette score. This score measures the quality of the clustering by calculating the average distance between a sample and all the other points in the same cluster. The range of the score lies between -1 and 1, where 1 indicates that the sample in the cluster is assigned to the right cluster. When the clustering is done we will compare the results. We hope to see a pattern in every cluster and the possibility for us to measure which features contribute the most to the clustering. The result of this phase will consist of two sets, J and R, each possessing optimal characteristics. These sets will serve as the foundation for subsequent stages of the study.
When selecting features, it is essential to ensure that important features are not elim-

inated. Solely relying on feature selection algorithms or domain knowledge is a critical mistake. Thus, it is necessary to frequently verify our results with domain experts, as some features hold significant business value that might not be immediately evident from an analytical perspective. However, reducing the size of the datasets is also crucial for the subsequent prompt creation stages, enabling the LLM to generate an adequate recommendation list and ranking. The challenge in our data lies in the combination of categorical and numerical data, which can complicate the feature engineering process. Finding the perfect balance in limiting the dimensions and retaining the most useful information in the data is key.

6.3 Recommendation

During this stage, we employ the TALLrec framework and adapt it to fit our specific requirements. Firstly, we will use Llama3 to respond to our prompts, followed by the selection of three distinct prompt configurations. This process aims to evaluate the suitability of LLMs for our intended purpose and determine the most effective prompts. Crafting tailored prompts is vital for achieving the desired outcomes [79, 80].
Numerous LLMs are accessible, such as GPT-4, LLama3, and PaLM 2 [76, 81, 82]. For our research, we prioritize utilizing an open-source LLM openly accessible to the public. We have opted to use the LLM: *Meta-Llama-3-8B-Instruct*, this LLM is created by Meta-AI and is easy to access and use. We have decided to choose LLama3 after careful consideration, this LLM is well-known and one of the best-performing open-source LLMs at this moment. For our research, we opted to use the default (Not fine-tuned) version of the Llama-3-8B-Instruct LLM. This decision was primarily due to the lack of a ground truth in our use case, which complicates the fine-tuning process. Additionally, we chose to focus on other aspects of the study and made compromises to accommodate the limited time frame available for completing this thesis.

To initiate the prompt we first need to format the input. This will be achieved by formatting every row in datasets J and R_u to a format which is accepted by the prompts. This means that the format needs to be readable for the LLM, avoiding a single block of text without separation between the different sections. Proper indentation and line breaks are necessary where appropriate. Experimenting with different formatting options is essential to achieve the desired structure. Subsequently, this formatted information will be inserted into the prompt, and the resulting list will be saved for later reference. Standardizing this format throughout all prompts is pivotal for receiving stable responses from Llama3. Furthermore, the names of the recruiters are mapped onto a fake name to maintain their privacy. We will provide a detailed description on the final job format in the Experiments section.

We will create different prompt instructions for our research. All prompt instructions are influenced by the original instruction in the TALLrec paper. The instruction is structured to take as input the jobs successfully placed by the recruiter along with their respective features, essentially reflecting the historical sequence of recruiter actions. For instance, a successful placement like "Pakkerbezorger, PostNL" might include features such as location (Haarlem) and average salary (€2300 per month). On the other side, the input includes candidate jobs that share similar features with the placed jobs and can be com-

pared accordingly. Therefore, the query posed to the LLM prompts it to recommend candidate jobs based on the recruiter's historical interactions. In the prompt, we will focus on limiting the amount of jobs added to the recommended list. We do this by creating a prompt which is strict and only recommends jobs that are perfectly suitable for the recruiter, thereby limiting the output size. When the output of the prompt is "Yes" we add the job and the respective features to the recommended job list.

Given that the size of R_u varies significantly, with some encompassing up to 20 jobs, we must manage potentially large prompts. To ensure stable responses, we will employ the sliding window strategy inspired by RankGPT [68]. This strategy will help slice the recommendation prompts to a manageable size. Specifically, when the length R_u exceeds 15, we will segment them from $(15 - w)$ to 15, sliding the window in steps of 5, where w represents the total number of historical interactions. This approach allows for the regularization of prompts, ensuring more consistent and stable responses.
The prompt engineering component of this stage presents a significant challenge. It is crucial to construct a detailed and appropriately sized prompt to ensure the LLM understands it and provides a stable and accurate response. By employing the sliding window strategy, we limit both the prompt size and the computational resources required. Additionally, we rely entirely on the untrained Llama3 model's capability to generate a satisfactory recommendation list. This reliance confines us to the performance of this particular model, which has its inherent limitations.
We will conduct experiments with multiple task instruction settings to find out if a different wording of the prompt makes any difference in the results. We will configure three different prompts to serve as the basis of our recommendation stage. These prompts are designated as C1, C2, and C3:

- **C1**: This is the "regular" prompt, influenced by the TALLrec paper.
- **C2**: A simpler prompt, focusing on a small token count and straightforward language.
- **C3**: A more complex prompt, utilizing a higher token count and more sophisticated language.

For each configuration, we will generate an instruction input, as shown in equation 4. In this context, x represents the instruction input derived from our experiments, and y denotes the output resulting from this instruction. We will utilize the default parameters θ of the Llama3 model $\mathcal{M}$ on the training set $\mathcal{Z}$, which, in our case, comprises the dataset input from J and R_u.

This approach allows us to evaluate the effectiveness of different prompt configurations and determine the most effective strategy for generating accurate and useful job recommendations. We will utilize GPT-4 [76] to assist in creating and experimenting with different prompts. We will evaluate the prompts by analyzing the consistency of the "yes" or "no" responses and assessing the relevance and coherence of the results in the context of our research. For validation, we will compare the results of the three configurations by conducting a survey. The three different prompt configurations will be applied to the sets R and J. The objective is to generate a recommended list: a 'job list' comprising items

pertinent to the user. These configurations aim to personalize a list of jobs that align with user relevance. For instance, consider a user whose historical data includes a successful placement with the job position titled "Logistiek medewerker, KLM". The LLM should ideally recommend similar positions such as "Bagageband medewerker, Transavia", reflecting comparable attributes like salary, location, and hiring duration. Conversely, a position such as "Klantenservice Medewerker GGD", which diverges significantly from logistics-related roles, should not appear as relevant. The effectiveness of the implemented prompts is measured by their ability to discern and categorize these similarities and discrepancies based on the specified features, thus demonstrating the tailored capability of the LLM in job recommendation scenarios. For every configuration, a job list is generated per recruiter. This list is generated by looping through all the candidate jobs and effectively instructing the LLM on the given configuration prompt. These three different recommended lists will be taken to the next stage. Due to the lack of a labeled dataset and sufficient data points, we have opted to not tune Llama3 for our research.

6.4 Ranking

Following the generation of the recommended job list from the three different configurations, the next step involves ranking these jobs. For this purpose, it is essential to establish a tier list, prioritizing certain jobs over others. Given that Llama3 has recommended every job in the provided list, we must develop a methodology to rank these jobs based on specific criteria. We leave room for the LLM to make a decision on which criteria are more relevant than others.

For the optimal response it is necessary to minimize the prompt size whenever possible. With the recommended job list averaging around 200 jobs, directly querying this large list will result in unstable and inaccurate query responses [83]. Therefore, instead of submitting all 200 jobs at once and requesting the LLM to create a top 10, we have opted for the LLM to assign a similarity score between 0 and 100 to each job. The similarity score for each job is directly produced by Llama3, leveraging its internal processing capabilities, rather than being computed by an external algorithm. We have also chosen to utilize the sliding window strategy again for the recruiter's history placements [68]. When a prompt contains multiple windows, we will average the scores from all the windows to determine a final score for each job. The prompt is created by testing multiple configurations to eventually reach a stable and accurate response from the LLM, this allows for a more manageable and effective ranking process. Notice how we explicitly state the format of the answer from the LLM and we provide an example as well. Because our LLM model is not pre-trained, it is crucial to provide a detailed prompt to steer the response in the right direction.

The output of the Ranking Prompt is an integer score assigned to each recommended job in the job list. A top 10 ranking is generated by sorting the jobs according to their scores, with the highest score receiving the top position. This ranking process is conducted separately for each configuration per recruiter. Consequently, for a recruiter u, the outcome of this stage is three distinct rankings: one for Configuration 1 (C1), one for Configuration 2 (C2), and one for Configuration 3 (C3).

The lack of explainability in this stage poses a significant challenge. The similarity score is generated directly by Llama3, but the underlying rationale may be artificial and opaque. Another issue is the risk of overfitting or tunnel vision, where the LLM may overly focus on specific patterns or biases. To address these challenges, we have opted to initiate a new conversation with the LLM for each prompt. This approach helps mitigate tunnel vision by treating every job as a new instance, ensuring that each job is evaluated independently and objectively.

6.5 Validation

The final stage of our research involves validation. In this stage, we use the rankings generated for each configuration per recruiter and validate them through user feedback. Given that recruiters will be the end users of this model, it is crucial to understand their preferences and how well the model's outputs align with their day-to-day tasks. This feedback will help ensure the model's practical relevance and effectiveness in real-world applications.
The generated rankings will be sent to recruiters in the format of a Microsoft Forms survey. The recruiters selected to complete the survey are directly drawn from dataset R, which includes both active recruiters with a history of placements within the company. To obtain comprehensive validation output, we will conduct a survey with multiple questions. It includes a mix of quantitative and qualitative questions, such as Likert scale ratings for the perceived relevance of recommendations and open-ended questions for detailed feedback.
These questions will be formatted in a survey and sent to the recruiters personally. The aim is to get at least 20 responses from the 27 total recruiters to get a sufficient understanding of their feedback. Responses will be collected over one week with a follow-up reminder after 2 days. The responses will be stored securely to maintain confidentiality and specific answers will be presented anonymously.

The responses will be analyzed using different statistical metrics, comparing the results from each recruiter to ultimately identify the optimal configuration. We will look at the average similarity scores as well as the deviation in answers. We will compare the different survey results to get a sense on how our model performed. We anticipate that one prompt configuration will emerge as the most effective, serving as a reliable prompt for future use. Additionally, we will analyze the responses to questions regarding the value of JobRank for its work and its potential time-saving benefits. This is a crucial aspect of the research, as it is not enough to identify a sufficient configuration; the model must also be perceived as beneficial and efficient by the users for it to be deemed successful.

The outcome of this stage will determine whether our model can potentially enhance the workflow of recruiters at Manpower and identify the most optimal configuration for doing so. However, there are challenges, particularly regarding the response rate. It is crucial to collect as many responses as possible to ensure a valid validation. To achieve this, we can increase the response rate by minimizing the amount of text that respondents need to read, thereby retaining their attention longer. Additionally, it is essential to ensure that the rankings are distinct enough for recruiters to make clear comparisons between them.

7 Experiments

The aim of the experiments is to determine which state-of-the-art LLM methods are the most suitable to create personalized job rankings for the recruiters at Manpower. With extensive testing, we have tried multiple methods described in the previous section. We have investigated the impact of different feature selection techniques on model performance to identify the most effective features for job recommendation tasks. This includes evaluating methods such as filter-based, wrapper-based, and embedded feature selection techniques. Furthermore, we assessed the effectiveness of different prompt-generation methods used to improve the interaction between models and data. This includes analyzing how variations in prompt design can influence the accuracy and stability of the recommendations provided by the models. We also implement a human feedback evaluation method to assess the practical applicability and user satisfaction with the rankings. This involves collecting feedback from recruiters to assess the quality of our model.

7.1 Experimental Design

In our experiments, we employed a range of software and hardware configurations to draw our conclusions. These choices were guided by the model's size and the associated computational demands. We had the advantage of utilizing the Dutch National Supercomputer, Snellius, hosted by SURF. This resource significantly reduced our computation time and enabled the successful execution of this research. For our code, we made use of Python 3.11.5 with the necessary libraries. For the feature engineering phase, we made use of Jupyter Notebooks. This way we could easily get access to the database and run snippets of our code to finalize the datasets. Furthermore, for the prompt creation, we made use of a Python script using Pycharm.
For the feature engineering phase, we utilized our local HP Zbook, which runs Windows 10 and is equipped with 32GB of RAM. We obtained the required datasets from the Manpower data warehouse. Access to the Bullhorn and Mysolution datasets was achieved by connecting to a SQL Server database hosted on Azure. This connection was established through a Python script, with authentication managed via Azure Active Directory. We executed several SQL queries to retrieve the data and used the pandas library to format it, creating a suitable environment to data cleaning and preparation. It took considerable time to make this connection and retrieve the data. Therefore, to limit the amount of SQL connections, we have opted to store the datasets locally and use the stored datasets for the remainder of the thesis research. The initial stage of our feature engineering setup was to merge all the necessary columns from the Manpower data. This was done by the *pd.merge* function and created a significant challenge. Because of the different key variables merging had to be done carefully otherwise certain merge mismatches could occur which would compromise the quality of the data.
For prompt creation, we selected the LLM: *Meta_Llama-3-8B-Instruct*. This choice was driven by the model's state-of-the-art performance within the Llama 3 series. The instruction-tuned Llama 3 models are optimized for dialogue applications and exceed many open-source chat models on industry benchmarks. Although Meta also offers a larger 70B model, we opted for the smaller 8B version due to computational constraints.

To experiment with various prompts, we utilized an open-source platform designed to facil-

itate the execution of large language models (LLMs) on local machines, thereby enhancing the accessibility and customization of advanced AI technology. Due to the limitations of our local machine, it was necessary to use a more computational-friendly solution. This platform, known as Ollama [84, 85], enabled us to obtain prompt results more efficiently and allowed for iterative refinement of prompts to achieve the desired outcomes. However, it's important to note that while Ollama provides a quick and user-friendly solution, it is a simplified version of our core LLM. As a result, the outcomes may differ compared to using the original Llama3 Model. Ollama was accessed through a Python script and the prompts were inserted using two loops. The algorithms used to derive the prompt responses are shown in algorithms algorithms 1 and 2.

Algorithm 1 outlines the recommendation stage. In this algorithm, a recommended job list is generated for each recruiter. The input parameters include the prompt type, which can be either C1 (Regular), C2 (Simple), or C3 (Complex) as shown in Tables 2 and 3. Additionally, the algorithm receives the historical interactions for a given recruiter, denoted as R_u, and the dataset of all placed jobs, denoted as J. The input also includes the model M, which is the Llama3 model, along with a window size of 15 and a step size of 5. The algorithm initializes the job list as an empty list and tracks the number of times the response is "yes" for a given candidate job. This is based on the requirement that at least 50% of the windows must respond "yes," as described in the prompt creation section. The algorithm then loops through the windows, generating a prompt for each window to elicit a yes or no response. When a job receives a "yes" in more than 50% of the windows, it is added to the recommended list.

Algorithm 2 outlines the ranking stage. In addition to the same inputs used in the recommendation algorithm, this algorithm also takes as input the recommended job list generated by Algorithm 1. The initial steps are similar, where the historical interactions are segmented into windows and the ranking prompt is provided as the instruction input for the LLM. A response is generated, and this response is checked to ensure it is an integer. If this condition is met, the score is saved and appended to the dictionary of job scores.

These algorithms are implemented in a Python file named *Main.py*. The size of the candidate list, denoted as J, is 1536, and the average size of the recruiters' historical interactions is 8. Each prompt is generated separately within its own conversation to ensure that responses are not influenced by previous prompts.

To execute the entire code of our LLM utilization, we made use the Dutch National supercomputer, Snellius [86]. This supercomputer is equipped with significant processing power and multiple GPUs. For our research, we used the partition *gpu_h100*, which consists of 64 cores per node and offers 720 GiB of available memory. According to SURF, "Multi-Instance GPU (MIG) is NVIDIA technology that allows partitioning a single GPU into multiple instances, each fully isolated with its own high-bandwidth memory, cache, and compute cores. On Snellius, GPUs using MIG technology are divided into 2 independent instances per GPU, resulting in a total of 8 GPU instances per node." From the available processing speed we utilized 128GB of Ram and 1 Nvidia H100 GPU and 16 CPU cores.

The chosen model, *Meta_Llama-3-8B-Instruct*, is accessed via Hugging Face [87] using

the transformers and vLLM libraries. vLLM, which stands for "Virtual Large Language Model," is a "fast and easy-to-use library for LLM inference and serving" introduced in 2023 [88]. It optimizes memory allocation for using an LLM by implementing an attention algorithm called PagedAttention. This algorithm divides the attention process into smaller, more manageable 'pages,' thereby reducing memory usage. The library enhances the LLM's ability to handle multiple requests simultaneously and respond faster than traditional methods.
Using this library, we set the parameters θ to the default values, with a temperature $\tau = 0.7$ and top-p or nucleus sampling of 0.9. The temperature parameter controls the randomness of predictions in the language model, affecting its creativity. Nucleus sampling manages the cumulative probability of token choices, ensuring that only the most probable words are considered, which improves the model's accuracy and reduces memory usage. In our case, it considers the smallest set of words with a cumulative probability of at least 90%.

7.2 Implication Details

7.2.1 Feature Engineering

For feature selection, we have opted to use various straightforward methods instead of complex techniques to maintain explainability and increase efficiency. Given the initial dataset size of over 100 features, we opted for a simple approach to limit the complexity of the process. Simpler methods allow us to quickly identify and remove irrelevant or redundant features without sacrificing transparency. Additionally, these methods are less computationally intensive, which is beneficial given the scale of our dataset. By using these straightforward techniques, we can ensure that our feature selection process is both effective and interpretable. Before we can conduct any experiment regarding feature selection we have transformed all the categorical variables into numerical ones by using a label encoder. This is a function from scikit-learn that encodes target labels with values between 0 and n_classes-1.

Initially, we have used domain knowledge to remove a rough amount of features from the initial dataset. By communicating with experts on the data regarding both the Bullhorn data as the Mysolution data we have removed over 40 features in the dataset. These features included:

- Duplicate features
- Job descriptions with significant noise
- Features with only one unique value
- Irrelevant features, such as business codes or job status within the company
- Unnecessary key features

We have checked the removal of each feature with a domain expert and concluded that, to decrease dimensionality and thus improve efficiency, these features were unnecessary.

The resulting dataset contained 59 features.

The next step is a correlation analysis. The aim of this analysis is to identify redundant features with high correlation. For example, if a feature has a high correlation with another feature, it means that changes in one feature are mirrored by changes in the other. This redundancy indicates that one of the two features does not provide unique information and can therefore be removed without affecting the overall dataset's integrity.
By using statistical thresholds to define what constitutes a "high" correlation, we can systematically eliminate redundant features. This process not only simplifies our model but also helps in reducing overfitting and improving computational efficiency. Given that our data is processed by an LLM, careful consideration is necessary when removing features. For instance, during correlation analysis, it may be found that two features have a high correlation. However, both features might still be beneficial if included in the prompt, as they could provide valuable context or nuance that enhances the LLM's performance. For the analysis, we created a correlation matrix shown in Figure 2. Because of the amount of features in the dataset the matrix is filtered on correlation score greater than 0.7 to retain a clear overview.

The figure illustrates that 14 features have a correlation score meeting the established threshold. These features are suffixed to indicate their dataset of origin, such as "_job" for a feature coming from the JobOrder dataset. We do not immediately delete these 14 features, as careful consideration is required when removing features. For example, the feature "Quantity", which represents the number of hours an employee worked for a client, can be considered redundant due to its high correlation with "Revenue Amount" (total revenue generated by the employee). Despite this correlation, we have opted to retain the "Quantity" feature. It may be beneficial for the LLM to have information on both the worked hours and total revenue, as this context could enhance its performance. Given that we are working with an LLM rather than complex algorithms, preserving this information could prove advantageous. The features "Hour Days," which represent the number of working days in a week for a job, and "Rate," which indicates the wage per hour an employee generates for Manpower, are removed. This decision is based on retaining the "Quantity" and "Revenue Amount" features, as the high correlation makes the other features redundant. We also notice a correlation between "Gemeentenaam2023" and "City". It is obvious that this relationship exists since cities are part of "Gemeentes" or "Municipalities" in English. Since we are not looking into the scale of municipalities and we think that the city of the job is enough we have opted to remove the feature "Gemeentenaam2023". Furthermore, we decided to leave all ISCO codes because they are vital for our LLM to understand the characteristics of a certain job and we removed placement and product type since they are only valuable for Manpower itself.

Subsequently, we performed a variance analysis on the remaining columns. The objective of this analysis is to identify features with low variance. Features with low variance have values that do not change much across different rows, meaning they provide little information to distinguish between observations. By identifying and potentially removing these low-variance features, we can simplify our dataset without losing valuable information, ultimately improving the performance and interpretability of our models. The results of this analysis are shown in Figure 3. The data is filtered again due to the size and only the

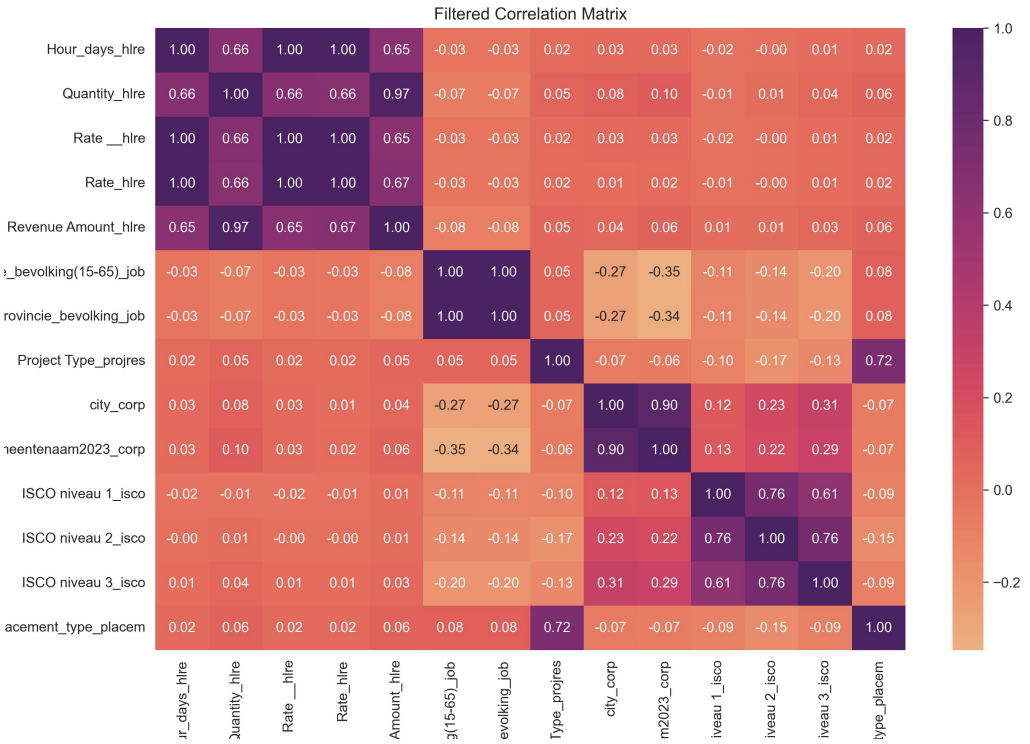


Figure 2: A Correlation Matrix from the Initial Dataset, Filtered by Feature Correlation Greater Than 0.7

features with a variance lower than 10 are plotted. There are 19 features with 7 features having a variance around 0. These features consist of just one unique value and can be removed from the dataset. Although the Education level and degree have a very low variance, they could still potentially be beneficial for our research. It is important to know what educational background a candidate needs in order for him/her to be placed. The common start day of a job has a low variance since there are only 5 workdays in a week. However, utilizing our Domain Knowledge we can also deem this feature to be irrelevant since the start day of a job has little to no influence on the ability of a recruiter to place a candidate on this job. Furthermore, the "Application Status", "Status" and "IsOpen" features can also be removed. These features are only relevant to the organization itself. Removing these features reduced the dimensionality of our dataset but retained the most important information.

For the correlation analysis, we opted to use Principal Component Analysis (PCA)[57]. This technique allows us to identify the most significant components within our dataset and assess their importance by transforming the original features into a set of linearly uncorrelated components. By analyzing the explained variance, we can focus on the most critical features that capture the most variance in the data. The results of our variance explained calculation are displayed in Figure 4. Our analysis indicates that the first three components, particularly the first two, account for a significant portion of the data, explaining 29.5% of the total variance. Specifically, the first component explains 15.3% of the variance, and the second component explains around 14.2%. This indicates that these

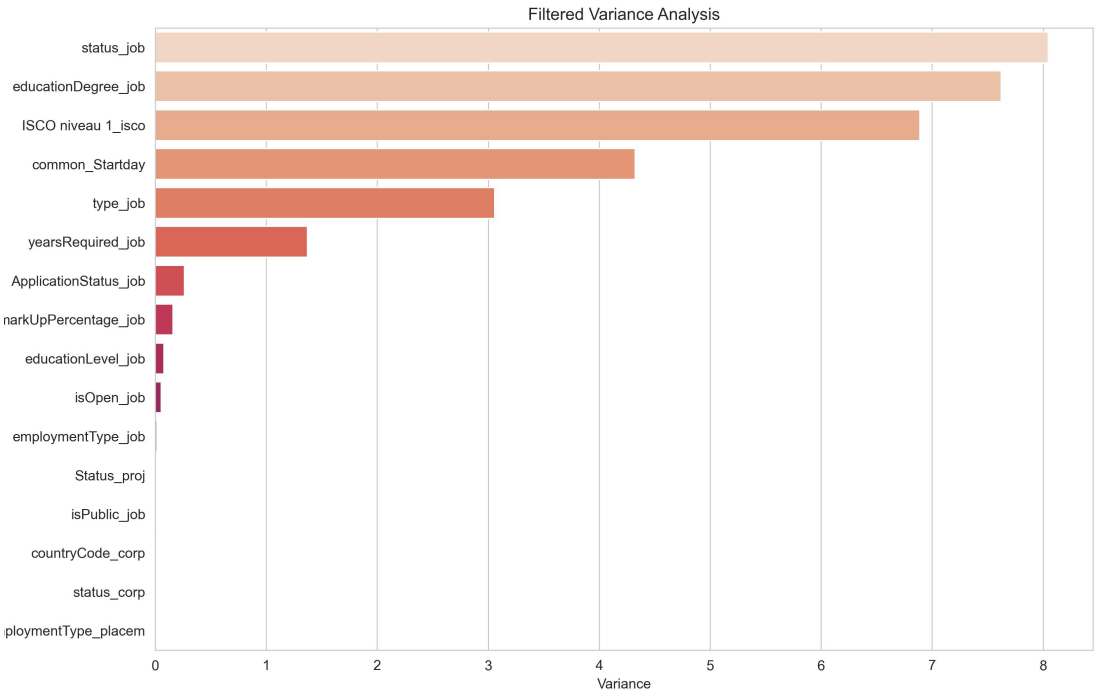


Figure 3: A Variance Analysis Bar Plot from the Initial Dataset, Filtered by Variance Value Lower Than 10

two components alone capture a substantial amount of the underlying structure of the data.

Further down the graph, the variance levels out around 5% per component after the third principal component. This pattern is typical in PCA, where the first few components capture the majority of the data's variance, and subsequent components contribute progressively less. Although the initial components explain most of the data, we aim to retain 85-90% of the variance to preserve the most important information. This goal is achieved by selecting approximately 12 to 14 components. By doing so, we ensure that we capture the majority of the variance while reducing the dimensionality of the data, which can help improve the performance and interpretability of our data and limit the size of the instruction input. When we dive deeper into the components, we derive that the most important feature for PC1 (Principal Component 1) is the Number of Employees a corporation has, with a loading of 0.47. A loading indicates how much a feature contributes to a principal component. This is followed closely by the Type of Job, which has a loading of 0.45. For PC2 (Principal Component 2), the most important feature is the Quantity of Hours Worked, with a loading of 0.59, followed by the Revenue Amount, which has a loading of 0.56. This means that for PC1, the Number of Employees and the Type of Job are the features that most strongly influence this principal component. Similarly, for PC2, the Quantity of Hours Worked and the Revenue Amount are the features with the strongest influence. The loadings indicate the strength and direction of these influences.

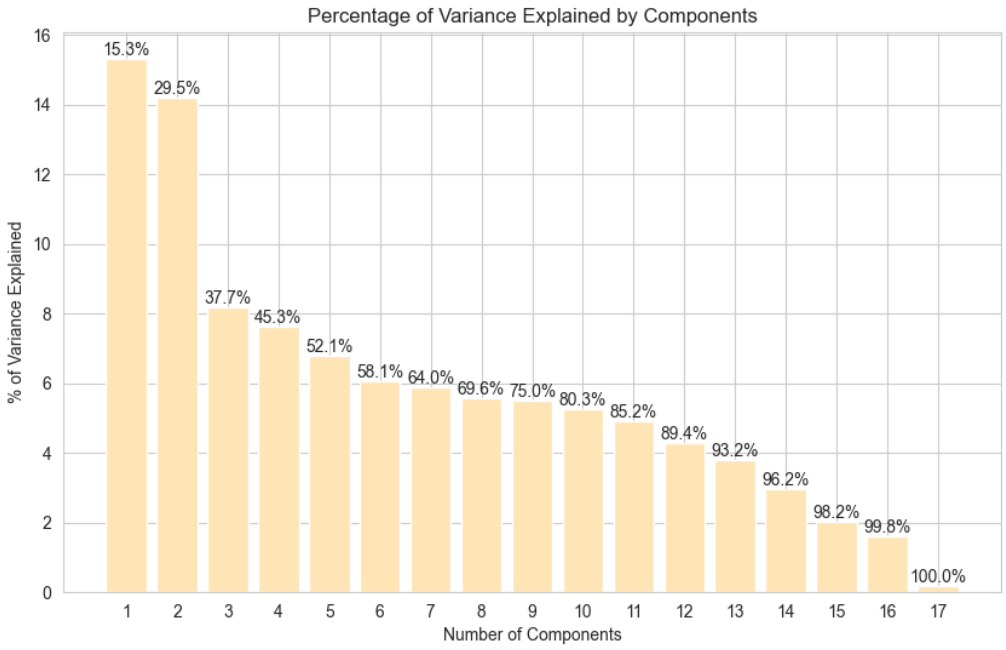


Figure 4: Explained variance plot sorted by components, with the cumulative values printed on top of the bars.

Our following step in our clustering analysis is the creating of the clusters itself. As mentioned in the methods section we test our data against three different encoders. We will choose the encode that best suits our data to retain the most information from our categorical data. When we apply our three methods, binary/ordinal/frequency encoding, to the data and we apply our chosen algorithm: k-prototype [56, 89] we can measure their silhouette score. We are running this experiment with 10 clusters and 5 initiations. The clustering done in the least amount of moves will be the chosen initiation. We then calculate the silhouette score and the results are shown in table 1. The silhouette scores indicate the ability of the algorithm to create distinct clusters. These scores range from -1 to 1, with higher scores indicating more well-defined clusters. Among the encoding methods, binary encoding yielded the highest silhouette score of 0.12, suggesting it forms clusters with the most well-defined boundaries compared to the frequency and ordinal encoding. However, a score of 0.12 is relatively weak, as evidenced when visualizing the clusters on a 3D PCA plot, which shows that the algorithm has some difficulties in creating distinct clusters. We determined that binary encoding is the most suitable method among the three options.

Encoding Method	**Silhouette Score**
Frequency Encoding	-0.26
Binary Encoding	0.12
Ordinal Encoding	-0.33

Table 1: Silhouette Scores for Different Encoding Methods

This analysis resulted in a PCA plot, shown in Figure 5, which illustrates the distribution

of the clusters within the dataset based on the first three principal components. Principal Component Analysis (PCA) reduces the dimensionality of the data while preserving as much variance as possible, with PC1, PC2, and PC3 capturing a significant portion of the total variance.
From the PCA plot, it is evident that there is a lack of clear, well-separated clusters. However, some distinctions can be observed:

- **Clusters 0, 1, 3 and 6**: These clusters are densely packed and show a good distinction from the rest. This could be indicating that these clusters characteristics are significantly different that the others. Especially cluster 0 has a clear separation from the rest of the pack.
- **Cluster 4**: This cluster appears to be arranged in a linear manner along PC3, showing an increasing contribution towards PC3.
- **Clusters 8, and 9**: These clusters exhibit some degree of separation within the middle of the big overlapping cluster.

Despite these observations, there is a considerable amount of overlap between many of the clusters. This overlap is consistent with the low silhouette score, indicating that the clusters are not well-defined and points are not distinctly closer to their own cluster compared to neighboring clusters. The overlap suggests that the categorical features encoded in the dataset may not capture sufficient variance to distinguish between clusters effectively. Additionally, it may indicate that the clustering algorithm did not perform optimally on this dataset with the given features and encodings. However, after optimizing the dataset as much as possible, we conclude that the data may inherently lack the distinction needed to create clear clusters as desired.

In table 12 in the appendix we show the characteristics per cluster. We have made this cluster analysis to get a deeper understanding of what is in the clusters and to determine if we want to keep these clusters in our data. Incorporating the cluster as an additional feature in our dataset could potentially simplify the LLM's task of categorizing certain jobs, thereby enhancing the quality of the recommendations.

Some clusters exhibit evident categories, such as clusters 0 and 1, which clearly correspond to two of ManpowerGroup's largest clients, KLM and PostNL. These clients account for over 90% of the total cluster population, indicating that these clusters are predominantly categorized by their respective clients. These clusters are therefore very distinct and this behavior is also visible in Figure 5, where the most distinct and densely packed clusters are clusters 0 and 1. Another notable cluster is cluster 6, which contains 94% of "Laders and Lossers." Although this cluster primarily consists of employees from CEVA Logistics, it includes a variety of other clients as well. Additionally, cluster 6 distinguishes itself by having a negative correlation with the mean pay rate and salary, indicating that the "Laders and Lossers" in this cluster tend to have lower average salaries. Furthermore, clusters 3 and 7 also contain many jobs with the "Laders and Lossers" ISCO code. However, they are distinct in that cluster 3 comprises jobs from clients with fewer employees, indicating smaller clients, whereas cluster 7 includes larger clients with many more offices. This distinction helps categorize jobs based on the size and distribution of the clients.

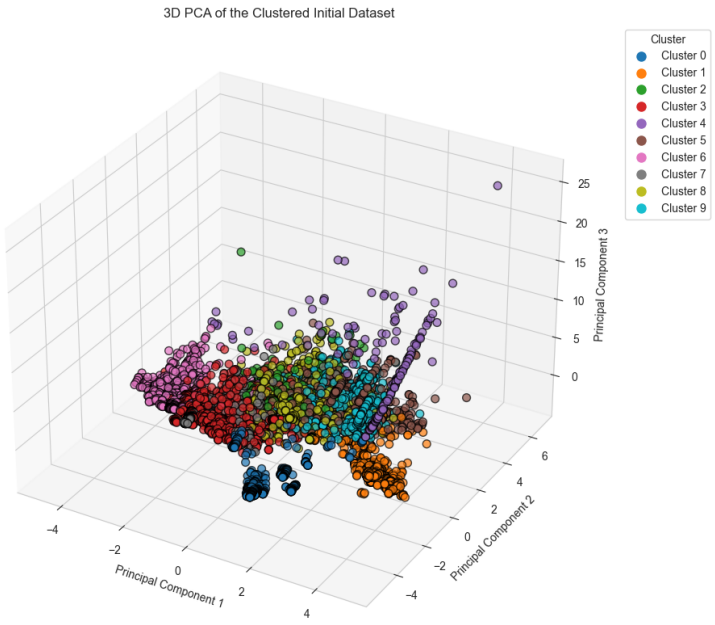


Figure 5: 3D Principal Component Analysis plot of the Clustering of the Initial Dataset

These clusters do exhibit significant overlap, as indicated in Figure 5.

Although there is some overlap among the clusters, they provide valuable additional information about the jobs. Grouping certain jobs together enhances the explainability of our data and could improve our LLM recommendations. Therefore, we have decided to include the clusters as an additional feature to enrich our dataset and potentially increase the accuracy of the LLM's ability to rank the jobs.

After the feature engineering stage, we have selected 30 features per job. An example of a job with its features is shown in the example below:

Job number: 0, **Job name:** Warehouse Operator Ceva Eindhoven AUGUST 2023, **Corporation name:** CEVA Logistics Netherlands B.V.

Features of the Job
Recruiter: Jack Nelson, **Number of placed candidates:** 4, **Time to hire (in days):** 30, **Revenue gained from job:** €2761.74, **Education Degree:** VMBO / MAVO, **Quarter:** Q3 2023, **State job is located in:** Noord-Brabant, **Cost Center:** 255 CEVA Logistics, **Application Status:** Do not publish, **Corporation Zip code:** 5301LJ, **Corporation City:** Zaltbommel, **Source:** Makelaar, **ISCO Level 1:** Intellectuele, wetenschappelijke en artistieke beroepen, **ISCO Level 2:** Specialisten op het gebied van informatie- en communicatietechnologie, **ISCO Level 3:** Software- en applicatieontwikkelaars en -analisten, **ISCO Level 4:** Systeemanalisten, **ISCO Code:** 2511, **Corporation Ad-**

dress: Hogeweg, **Job Address:** De Schakel, **Mark up percentage:** 1.0, **Job Title length:** 46, **Revenue Per Day:** 57.94, **Job type:** Full-time, **Years Required:** 0, **Corporation offices:** 37, **Quantity:** 123.94, **Provincie_bevolking(15-65):** 1685839, **Employees number:** 0, **Hours Per Week:** 40, **Salary:** 0, **Job openings:** 20, **Cluster Category:** 6

Every job will be input into the model using a specific format. We have chosen to define a job by three key features: Number, Name, and Corporation. This structure allows the LLM to clearly understand how to identify each job individually. After specifying these key features, we introduce some blank space before presenting the additional features of the job. This layout helps to distinguish the key identifying features from the other attributes of the job.

7.2.2 Recommendation

With the final job format completed, we can continue in creating the prompts. These prompts are essential for our model and need to be fine-tuned. As previously mentioned, having a sufficient prompt can be challenging but the results can be very positive for the outcome. [79, 80, 83]. We have fine-tuned the prompts by sampling our data and generating the responses for the prompts. When we got the desired results we continued increasing our samples until we reached a sufficient test size that we finalized our prompt creation.

Initially, we started with the example prompt from the TALLrec framework and customized it to fit our needs. It was crucial to provide clear instructions to the Llama3 prompt since it had not been fine-tuned on our dataset. The initial prompt, referred to as the 'regular' prompt, is shown in Table 2. This prompt was inspired by the TALLrec framework with some slight modifications. We selected this prompt as the regular version because it provided the clearest instructions for the task without adding unnecessary words.
One significant change we made to the TALLrec prompt was the addition of the word "perfect" and "extremely". Given the large size of our dataset, we wanted to create a more stringent prompt that would eliminate doubtful cases. By including these words we aimed to encourage the model to be more precise and selective in its recommendations. Another method we added to create a strict recommendation list is the introduction of a threshold on the window size. Whenever a recruiter has more than 15 historical interactions, R_u, we use a windowing strategy to divide the prompt load. We then apply a threshold that ensures the recommendation prompt answers "Yes" more than 50% of the time. For example, if a recruiter has an R_u of length 25, we create three overlapping windows: one from jobs 0-15, another from 5-20, and a third from 10-25. A job will be recommended only if the model's Task Output is "Yes" for at least 2 out of these 3 windows. This approach helps limit the size of the recommendation list, improving the model's runtime efficiency. Additionally, we added a sentence instructing Llama3 to limit its response to a single "Yes" or "No" without any additional feedback. During testing, we observed inconsistencies where the LLM not only responded with "Yes" or "No" but also provided reasoning. Since we will use these responses to create a recommended list later, it is crucial to establish a stable and consistent loop of answers to ensure reliable

processing in our code. Therefore we only accepted the response when this gave us the required answer otherwise we would run the prompt again.

Instruction Input	
Task Instruction C1: Regular	Based on the recruiter's historical placements, we aim to identify the most suitable job positions for this recruiter from a list of potential opportunities. For each candidate job presented, please respond with a clear "Yes" only if the job is a perfect recommendation for the recruiter and has extremely similar traits to the recruiter's historical placements, or "No" if it is not. Do not provide any additional feedback or comments. Limit your response to the single word 'Yes' or 'No'.
Task Input	Recruiter's historical interactions: $\{job1 : \{feature1, feature2, \ldots\}, job2 : \{feature1, feature2, \ldots\}, job3 : \{feature1, feature2, \ldots\}, \ldots\}$ Candidate Job: $\{job1 : \{feature1, feature2, \ldots\}$
Instruction Output	
Task Output	Yes / No

Table 2: Regular prompt example

After establishing the regular prompt (C1), we proceeded to create C2 and C3, as shown in Table 3. C2 is designed to be a simpler prompt, while C3 is intended to be more complex. We utilized GPT-4 [76] to rewrite our regular prompt into these new instructions. With these three different configurations, we aim to determine which setup is best suited for our dataset and produces the most effective rankings. The input of the tasks is in all three cases the same and consists of the recruiter's historical interactions and one candidate job to get evaluated. In the previous section, we have shown how a job gets formatted. To limit the size of the prompt, we utilized the window creation technique mentioned in RankGPT [68]. Specifically, we chose a window size of 15 with a step size of 5 for the historical interactions of a recruiter. This ensures that the prompt size never exceeds 16 jobs, including the candidate job. An example of a full prompt input is provided in Appendix A.4, where a recruiter with 4 historical interactions is illustrated. The outcome of all three configurations is a recommended list of jobs suited for the recruiter.

Task Instructions	
Task Instruction C2: Simple	Based on the recruiter's past placements, we want to find the best job matches for this recruiter. For each candidate job, reply with "Yes" only if the job is a perfect match and has extremely similar traits to the recruiter's past placements, or "No" if it is not. Only reply with 'Yes' or 'No'.
Task Instruction C3: Complex	Based on the recruiter's historical placements, we aim to meticulously identify the most fitting job opportunities for this recruiter from a curated list of potential positions. Each candidate job should be evaluated in comparison to the recruiter's historical placements. Please respond with a definitive "Yes" only if the job is an ideal recommendation for the recruiter and shares highly similar characteristics to the recruiter's past successful placements, or "No" if it is not suitable. Refrain from providing any additional comments or feedback. Limit your response strictly to the single word 'Yes' or 'No'.

Table 3: C1 and C2 prompt configurations

7.2.3 Ranking

Our next stage is creating a ranking prompt. In this stage, we need to format a prompt capable of ranking the jobs from the recommended list generated by the previous prompt. Through extensive testing, we concluded that creating a prompt to rank all jobs simultaneously was too challenging. We observed that, on average, around 30% of the total jobs were recommended to a recruiter, which, given a total of 1536 jobs, would result in an excessively large prompt. Also due to the fact that we still wanted to include the historical interactions by every recruiter in the ranking prompt since this was valuable information for the LLM to base a ranking on. Additionally, Llama3 encountered difficulties generating a ranking even with clear instructions to limit its response to a top 10 ranking. Therefore, we have opted for a similar approach as the recommendation prompts by looping through the candidate jobs in the recommended list one by one. Instead of asking Llama3 to create a ranking, we instruct it to assign a score to each job. This method simplifies the process and allows us to compile a ranked list based on these scores. The ranking prompt is shown in table 4. The prompt is focused on generating a similarity score. The score is derived from Llama3's interpretation, guided by instructions within the prompt to consider historical placements. When prompted for additional feedback on the thought process of the LLM behind a given score, Llama3 typically responded with trends such as: "The candidate job X had a salary of around Z and was located in D, showing similarities

with Job Y in the recruiter's historical placements." This indicates that Llama3 is capable of assigning a similarity score based on the job features. The effectiveness of these scores will be assessed in the validation stage.

Table 4: Example of Ranking prompt

Query input:	Identify the most suitable job positions for this recruiter based on their historical placements. Provide a similarity score between 0 and 100, where 100 indicates a perfect match and 0 indicates no similarity. Respond only with the score and format your answer as a single number. Example: 15
Document Input	Recruiter's historical interactions: $\{job1 : \{feature1, feature2, \ldots\}, job2 : \{feature1, feature2, \ldots\}, job3 : \{feature1, feature2, \ldots\}, \ldots\}$ Candidate Job: $\{job1 : \{feature1, feature2, \ldots\}$
Output:	Integer between 0 and 100

7.2.4 Validation

For the final stage, we conducted a validation experiment aimed at assessing our rankings. The objective was to evaluate the performance of these rankings and gauge recruiters' opinions on their potential to improve efficiency. We have decided to use Microsoft Forms for the survey due to its simplicity and the relatively small number of respondents, which totals 27 recruiters. A more complex tool like Qualtrics was deemed unnecessary. An example of the survey questions is shown in Table 5. The initial questions focus on selecting the best prompt configuration. We use a Likert scale to measure the quality of the C1, C2, and C3 configurations. Recruiters have been thoroughly instructed to respond honestly and avoid defaulting to the 'safe' option. They are encouraged to be critical and provide harsh ratings if a ranking is not satisfactory, ensuring we obtain a clear and accurate assessment of their opinions. To analyze the survey results, we will use simple statistical metrics such as average, median, and various plots. The configuration with the most votes from the fourth question will be considered the most successful. Additionally, we are interested in understanding if this configuration is genuinely beneficial for their work. Therefore, we have included two open-ended questions where recruiters can express whether they would use the system and if it would be efficient for their daily operations.

Input	Question	Output
C1 Job ranking 1-10 for recruiter r with job details.	Rank configuration C1 on a scale of 0 (Not suitable for me) to 5 (Very suitable for me)	Score from 0-5
C2 Job ranking 1-10 for recruiter r with job details.	Rank configuration C2 on a scale of 0 (Not suitable for me) to 5 (Very suitable for me)	Score from 0-5
C3 Job ranking 1-10 for recruiter r with job details.	Rank configuration C3 on a scale of 0 (Not suitable for me) to 5 (Very suitable for me)	Score from 0-5
Previous shown rankings	Which configuration represents the most suitable job ranking for you?	C1/C2/C3/None
	Would JobRank be valuable for you in your own work?	Yes/No
	Would you save time in using JobRank for your work?	Yes/No
	Do you have any other comments?	Open question

Table 5: Survey Questions for each recruiter

8 Results

The experiments detailed in the previous section were conducted over a runtime of approximately 45 hours. During this period, the code generated a .txt rankings file for each recruiter, along with a log file containing various statistics about the execution and outcomes. In total, there were 27 recruiters with historical interactions in the dataset, each receiving three personalized rankings generated by our experiments. This resulted in a total of 81 rankings with 810 jobs selected for inclusion in those rankings. In this section, we will discuss the outcomes of the experiments.

8.1 Optimal Prompt Engineering

The results of the prompt run for each recruiter are presented in Table 6. The table displays four metrics used to evaluate the outcome of our prompt: the time taken to complete a configuration run (in minutes), the number of historical interactions for each recruiter, the size of the recommended list generated by the prompt, and the average score of the jobs within the recommended list. The table includes the mean, median, highest value, and lowest value for each metric.
As previously mentioned, the average number of historical interactions is 8, with one recruiter having 56 placements. The median number of historical interactions is 5, indicating that more than half of the recruiters have 5 placements or fewer. This highlights that the number 56 is a significant outlier and that the dataset generally consists of a relatively small number of placements per recruiter. Furthermore, the average size of the recommended list is 473, which is 30.79% of the total 1536 jobs. This indicates that, on average, the prompts recommend approximately one-third of the jobs to any given recruiter. The average score of these recommended jobs is 73.26, which is the mean score across all different configurations. This number is quite high since the range asked by the prompt is 0

to 100. However, this is quite understandable since the jobs are from a recommended list which are occupied by jobs already deemed to be relevant for the recruiter. If the average was lower, then questions could arise on why the job was recommended in the first place.

Table 6: Prompt Column Metrics

Metric	**Mean**	**Median**	**High**	**Low**
Time (Minutes)	35	21	303	8
Historical Interactions	8	5	56	1
Size	473	415	1144	30
Average Score	73.26	73.21	83.18	67.57

Upon further examination of the data, we generated a box plot of the average scores to analyze the distribution. As shown in Figure 6, the data includes several outliers, contributing to the right skew of the box plot. However, the scores range between 67 and 83, indicating a narrow spread. This plot supports our previous assertion that the average scores are closely clustered and do not exhibit any anomalous values.

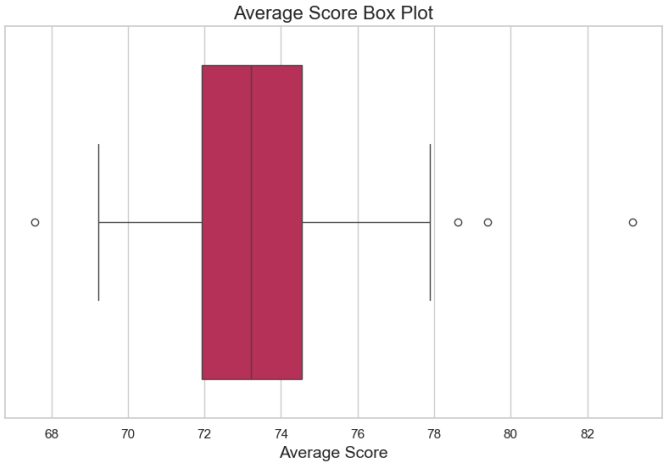
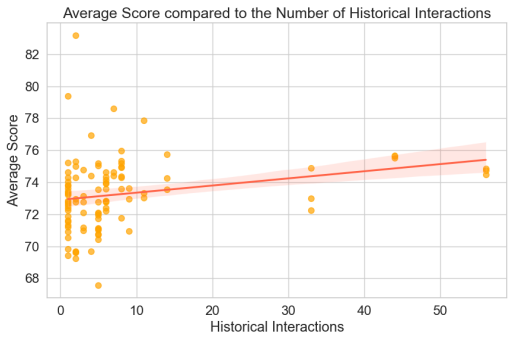


Figure 6: Average Score Box Plot

To explore the relationships among the different metrics in Table 6, we examine potential linear correlations. One key relationship to investigate is whether the number of historical interactions correlates with the time for one configuration. Another noteworthy relationship is the potential correlation between the average score and the number of historical interactions for a recruiter. Since more historical interactions provide additional data for the prompt to process, this could potentially impact the results. Both relationships are illustrated in Figure 7. The data in Figure 7a indicates a linear relationship between the number of historical interactions and the time taken for ranking generation. While most historical interactions remained within a runtime of 0 to 50 minutes, instances with a higher number of interactions occasionally exceeded two hours. This indicates a potential bottleneck if the dataset grows larger. On the other hand, the data in Figure 7b does not show any clear relationship. A slight linear correlation is observed, though it is not very significant and is primarily influenced by a few outliers. This suggests that the ranking scores remained stable regardless of the number of historical interactions. However, due

to the skewed distribution of historical interactions, with a significantly higher number of interactions between 0 and 10 compared to those between 10 and 70, establishing a clear linear relationship is challenging. For a more comprehensive and accurate analysis, additional data would be necessary to form reliable assumptions.

(a) Graph showing the relationship between the historical interactions and the time in minutes it took to create a ranking



(b) Graph showing the relationship between the average score given by the prompt and the historical interactions

Figure 7: Comparison of Historical Interactions with Time and Average Score

8.2 Comparative Analysis of Prompt Configurations

Continuing our analysis, we can compare the different configurations with each other. As shown in Figure 8, we plotted the results per Configurations of the important metrics. The average scores of all three configurations are very similar, which aligns with our previous findings. Since the ranking prompt is consistent across all three configurations, it is evident that the scores are provided stably and uniformly. However, differences emerge when examining the size and time averages of the three configurations. While the Regular and Complex configurations have similar statistics, the Simple configuration stands out. It has an average processing time of 28 minutes for creating the recommended list and their respective rankings. This discrepancy is likely due to the significantly lower average size of the Simple configuration's recommended list, which is 213 jobs compared to 633 for the Regular configuration and 573 for the Complex configuration. In the Simple configuration, the recommended list constituted, on average, 13.87% of the total jobs. This indicates that, despite using simpler language, the Simple configuration generated a more concise recommended list in less time.

To further analyse the data we have opted to create a frequency table of all the jobs in the top 10. We have opted to use the JDCO code to group certain jobs together and see per configuration which job is the most common. This analysis can indicate if a given configuration has a slight bias towards certain jobs. The result of this analysis is shown in tables 7, 8 and 9. On the left side, the tables show the five most frequent jobs that occupied any of the top 10 rankings created for any recruiter. Essentially these jobs occupied the top 10 rankings the most often and can be seen as the most popular ranked jobs. On the right side, the tables show the jobs with the frequency normalized

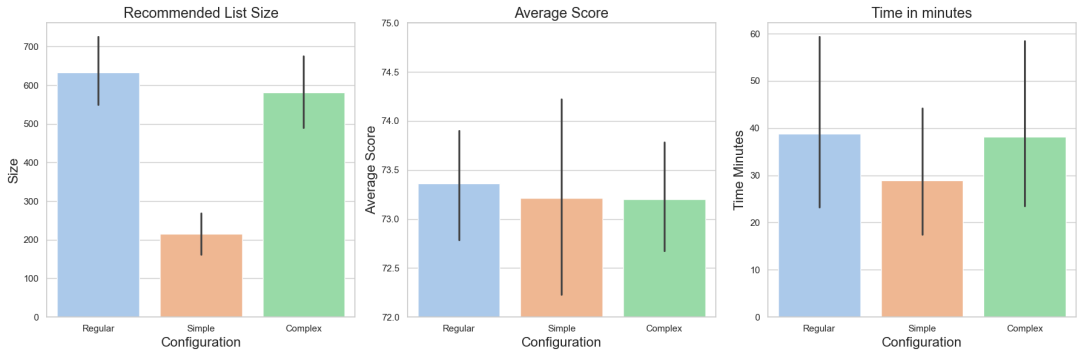


Figure 8: Average score, Time in minutes and Recommended Job List Size per Configuration

by dividing the value they occur in the rankings with the total times they occur in the total dataset J. Again it seen that the configurations of C1 and C3 are similar on the left side with the most frequent job scoring about the same. The C2 configuration stands out with "Administratief Medewerker" occurring for 46 times in the rankings. This job is the third most occurring job in the dataset J with 89 entries and over half are chosen to be in a top 10 which is significant. Another interesting detail is the average scores of the jobs. These average scores, unlike those in the previous section, are calculated only from the top 10 rankings. Notably, some average scores exceed 100, despite the ranking prompt in Table 4 instructing a score between 0 and 100. In total there were 156 scores above 100 from the 810 of total scores. This discrepancy suggests that the LLM occasionally failed to adhere to the prompt, thereby compromising the accuracy of the rankings. While the average scores across all created rankings are generally stable, the highest scores indicate potential inconsistencies in the LLM's scoring mechanism.

C1: Regular				C1: Regular		
Rank	JDCO code	Frequency	Avg. Score	JDCO code	Norm. Frequency	Avg. Score
1	administratief medewerker	28	102.29	1e medewerker bediening	3.0	99.33
2	productiemedewerker logistiek	22	118.54	assistent finance manager	2.0	97.39
3	logistiek medewerker magazijnmedewerker	15	280.70	chauffeur bezorger	2.0	98.50
4	assemblagemedewerker	11	390.90	customer service medewerker frans	2.0	194.75
5	inpakker	9	132.00	juridisch beleidsadviseur	2.0	100.00

Table 7: Frequency table per JDCO code for C1

C2: Simple				C2: Simple		
Rank	JDCO code	Frequency	Avg. Score	JDCO code	Norm. Frequency	Avg. Score
1	administratief medewerker	46	96.08	ambtelijk secretaris or	3.0	87.00
2	productiemedewerker logistiek	13	151.60	pakketsorteerder	3.0	115.83
3	financieel administratief medewerker	11	93.55	accountmanager binnendienst	2.0	86.07
4	logistiek medewerker magazijnmedewerker	10	92.69	assistent boekhouder	2.0	98.50
5	magazijnmedewerker	10	101.10	technician e	1.5	95.61

Table 8: Frequency table per JDCO code for C2

	C3: Complex			C3: Complex		
Rank	JDCO code	Frequency	Avg. Score	JDCO code	Norm. Frequency	Avg. Score
1	administratief medewerker	26	104.46	adviseur contractmanagement	2.0	171.50
2	productiemedewerker logistiek	17	245.51	agile software tester	2.0	97.50
3	logistiek medewerker magazijnmedewerker	12	110.95	financieel analist	2.0	205.75
4	magazijnmedewerker	9	155.70	medewerker weegbrug	2.0	97.50
5	assemblagemedewerker	8	104.44	projectleider engineering	2.0	97.50

Table 9: Frequency table per JDCO code for C3

8.3 Validation Survey

The survey was ultimately sent to 27 recruiters. We received a total of 17 responses, which is less than anticipated but sufficient to substantiate various assumptions. The survey results are presented in Table 10. Several open-ended questions were answered, which we will also discuss in this section however these open responses are not included in the table to enhance readability.

The first three columns are the answers to the top 10 rankings on a likert scale. The scale was set up like so:

- **1:** Totally not suitable for me
- **2:** Not suitable for me
- **3:** Neutral
- **4:** Suitable for me
- **5:** Very suitable for me

In total, recruiters rated the rankings as follows: a score of *1* was given 9 times, a score of *2* was also given 9 times, a score of *3* was given 8 times, and a score of *4* was given 19 times. Notably, no recruiter rated any ranking as very suitable (score of *5*). The ranking configuration that received the highest cumulative score was C3, with a total of 16 points, including eight scores of *4*. Configuration C2 followed with 15 points, and C1 received 14 points.
However, it is important to note that C3 also received the most *1* scores, indicating significant variability in the ratings. When we adjust the scoring system to treat neutral ratings (score of *3*) as 0 points, positive ratings (scores of *4* and *5*) as +1 and +2 points respectively, and negative ratings (scores of *1* and *2*) as -1 and -2 points respectively, the results change. Under this system, C1 scores -4 points, C3 scores -3 points, and C2 scores -1 point, making C2 the best-performing configuration.
This adjusted scoring system highlights that all three configurations received more negative ratings than positive ones. Even though there were 19 instances of a score of *4*, no recruiter found any ranking to be very suitable (score of *5*), indicating that while some configurations were somewhat effective, there is still significant room for improvement.

This is also evident in the question about the best configuration, where C1 received 4 votes, C3 received 3 votes, and C2 received 6 votes. Additionally, 4 recruiters selected the "None" option, indicating that no ranking was suitable for them. These results show that

Index	C1	C2	C3	Best Configuration	Is JR Useful for you	Is JR efficient for you
1	4	3	3	C1	Yes	Yes
2	4	3	2	C1	Yes	No
3	2	3	2	C2	Yes	Yes
4	2	3	4	C3	Yes	Yes
5	4	4	2	C1	Yes	No
6	4	4	4	C2	Yes	Yes
7	1	3	4	C3	No	No
8	1	1	1	None	Yes	Yes
9	4	2	2	C1	Yes	Yes
10	2	4	4	C2	Yes	Yes
11	3	4	4	None	No	No
12	3	4	3	C2	No	Yes
13	2	2	1	None	No	No
14	4	3	4	C3	No	No
15	4	4	1	C2	Yes	Yes
16	1	1	1	C2	Yes	No
17	4	4	4	None	Yes	Yes

Table 10: Recruiter Validation Survey Results

C2 clearly had the most votes for the most suitable configuration. While C3 had the highest total score on the Likert scale, it scored the worst in terms of the best configuration. This indicates that, although a few recruiters found C3 the most suitable configuration, the majority preferred another option.

Regarding the question of whether JobRank would be valuable for their work, the responses were clear: *12* recruiters voted "Yes" and *5* recruiters voted "No." Notably, most of those who voted "No" also indicated that none of the configurations were suitable for them. This correlation suggests that recruiters who did not find any ranking suitable were less likely to see JobRank as beneficial for their work. One experienced recruiter elaborated, "As an experienced recruiter, I prefer to make my own decision on which job should take priority." Another commented, "If the rankings were more suitable, they would be beneficial." Conversely, a recruiter mentioned, "It is very hard for me to determine, based on the information given in the top 10, which job should take priority. This decision depends on many more factors than just these features." When asked to clarify these factors, the recruiter mentioned aspects such as client relations and situational scenarios in their workload.

For the question of whether JobRank would make their work more efficient, the outcome was more divided. *10* recruiters voted "Yes" and *7* recruiters voted "No." Positive comments included: "If we have multiple job positions open, it could help divide the workload and prioritize them" and "It could help me determine which profiles suit me personally without having to keep track of them myself." However, some recruiters had difficulty

understanding how it would be beneficial. One commented, "I do not know how I would use it; I would need clear instructions on how this would work in my day-to-day tasks."

The recruiters were also asked whether any jobs in the rankings were ones they had already placed themselves. This would indicate that the model correctly recognized the recruiter's expertise and made accurate recommendations. Out of the 17 recruiters, *8* did not see any recognizable jobs in their rankings. However, *9* recruiters did identify recognizable jobs, with most of these jobs appearing in the C2 ranking. When asked for additional feedback, one recruiter responded: "Unfortunately, there were no jobs in the rankings that suit my expertise; I did not see any rating that would help me prioritize my jobs." Another commented: "I am only placing jobs for one client, and this client did not appear in any of the three rankings, which is a shame. However, I do see the potential in this model."

Overall, the survey provided some interesting insights. While most recruiters see the potential of this model, the outcomes did not benefit all of them. The model successfully recognized some recruiters' expertise and accurately ranked already placed jobs within the correct recruiter rankings. However, this occurred less frequently than anticipated. Additionally, the most suitable ranking configuration appeared to be C2, as it received the fewest negative reviews and the highest number of recognizable rankings from the recruiters. It also garnered the most votes as the preferred ranking among the three configurations.

9 Discussion

The objective of this thesis was to develop and implement an innovative job prioritization model tailored for recruiters, based on their historical interactions with job placements. This research aimed to assess the feasibility of creating such a model using a state-of-the-art Large Language Model (LLM) and data provided by Manpower Group.
Through various feature selection techniques, we successfully identified the most important features within the dataset. This process yielded a clean and usable dataset, reducing its dimensionality to enhance efficiency and decrease computational power while retaining critical information. A key aspect of this process was cleaning the dataset using domain knowledge. Additionally, by conducting correlation and variation analyses, we managed to filter out features that had minimal impact on the data, thus streamlining the dataset. A cluster analysis was also performed using the k-prototypes algorithm, which is suitable for our data since it works well with numerical data as well as categorical data [56]. By converting our categorical features using binary encoding, we successfully formed clusters that accurately differentiated a significant portion of our data, with certain clusters being particularly distinctive in their information.
This research also succeeded in generating a recommended job list based on the historical interactions of recruiters. By leveraging the existing TALLrec framework [13], we developed three distinct prompts: C1, a "regular" prompt primarily inspired by the TALLrec framework; C2, a simplified version of the regular prompt, characterized by shorter length and simpler language; and C3, a more complex prompt, utilizing more intricate language and providing a longer explanation. These prompts were capable of producing a recommended job list from an extensive candidate pool. Extensive testing and coding were done

to achieve the desired yes or no outcomes. By analyzing the history of placements for a given recruiter, the prompts generated a job list that averaged 30.79% of the total candidate pool. To ensure the prompts remained manageable and efficient, a window strategy was used for the historical interactions by a recruiter, as introduced in the RankGPT paper [68]. This strategy limited the size of the prompts, ensuring that the outcomes were stable and efficient. The size of the recommended job list indicated how strict a given prompt configuration was. While the C1 and C3 configurations managed to limit the recommended job list size to around 35% of the total candidate pool, C2 managed to shrink the size of the job list to 13.87%, which is a considerably lower number. This indicates the simplified prompt is much stricter and leads to more jobs being filtered out. This leads to a faster generation time, on average around 10 minutes per run faster than the C1 and C3 configurations. With 81 runs in total, this saved about 13 hours of run time, which is quite significant.

Following the creation of the recommended list, this research successfully developed a ranking prompt capable of assigning a score between 0 and 100 to the jobs in the recommended job list. On average, this resulted in a score of approximately 73/100 across all three configurations. This high average score suggests that the recommended lists consisted of jobs that closely matched the historical interactions, indicating the effectiveness of the recommendation stage. While most scores had a value between 0 and 100, there were 156 instances where this was not the case. These scores fluctuated between 100 and as high as 1400, which was not ideal. This indicated that the ranking prompt was not foolproof and certain rankings were compromised as a result. The reason for this could be that the output of the ranking prompt was not what was expected. Upon further analysis, it was found that the prompt sometimes would name the value of a feature, such as job number or salary, as an output as well as the score. The code would then pick up the first integer mentioned, which in this case was incorrect. To tackle this problem, certain foolproof measures need to be installed in the model to ensure that the score is between 0 and 100 before being put through. Considering that configuration C2 was significantly more efficient than the other two configurations and maintained a similar average score, it indicates that C2 is generating suitable recommendation lists while being more efficient. This efficiency, coupled with its comparable performance, suggests that C2 is an optimal choice for creating effective job recommendations.

Following the generation of the personalized rankings, we surveyed with the recruiters to test our model with the end-users. By presenting them with their personalized rankings from the three different prompts, we aimed to identify an optimal configuration and gather valuable insights on how our model would be interpreted, along with feedback for potential improvements. The C2 ranking was voted as most useful which is consistent with our earlier findings. The configuration also had the most similarities with earlier job placements being included in the rankings. The survey results indicated that while the rankings were on the right track, they were not ideal for everyone. When we evaluate the likert scale by scoring the scale ranging from -2 to 2, the overall sentiment towards the rankings was negative. Despite this, most recruiters recognized the potential of JobRank and believed it could be useful for their work. However, opinions were divided on whether it would enhance efficiency, with some recruiters unable to see how it would improve their workflow. Analyzing the recruiters' feedback, we concluded that they see the potential value of JobRank but not in its current state.For JobRank to be incorporated into their daily activities, the rankings must be highly accurate and error-free. Currently,

recruiters, especially those with significant experience, rely on their instincts. Therefore, only a sophisticated model with exceptional recommendations could potentially enhance their work efficiency.

10 Conclusion

This thesis aimed to develop a sophisticated job recommendation system tailored for individual recruiters by leveraging their historical placements and utilizing state-of-the-art Large Language Models to enhance efficiency in the recruitment process.
This research successfully created a model called JobRank using Llama3, generating a top 10 ranking for each recruiter based on their historical placements. In this study, we modified a complex dataset provided by Manpower and, through modern feature selection techniques, created a suitable baseline dataset.
Using this initial dataset, we developed three different prompt configurations to establish a recommended job list per recruiter. This was achieved by leveraging the TALLrec framework and limiting prompt sizes through a sliding window strategy. Consequently, three recommended job lists were created for each recruiter based on the different configurations. We then utilized an LLM again, employing a ranking prompt to generate a top 10 ranking for each configuration per recruiter. Concluding that the most effective configuration was C2, the simplified prompt. This configuration generated a recommended job list with similar scores to the other two configurations but achieved this in a significantly more efficient manner. Furthermore, C2 was voted as the best configuration in the recruiters' survey. Therefore, we can conclude that a simplified version of the prompt is the most optimal configuration for our model. This was achieved by addressing certain limitations, such as the challenging data quality and data availability within the Manpower organization. These challenges affected the overall performance of our model, as evidenced by the feedback from the recruiters. While they recognized the potential of the model, they concluded that improvements were necessary for it to be practically useful.
In conclusion, JobRank has the potential to significantly benefit the recruitment sector by increasing efficiency and assisting recruiters in prioritizing job placements. However, due to certain limitations and the need for further improvements, the model currently yields insufficient results for immediate use by recruiters. Enhancing data quality, availability, and model efficiency is essential to effectively integrate JobRank into the recruitment landscape. With these improvements, JobRank could become a valuable tool in streamlining the recruitment process and aiding decision-making for recruiters.
JobRank aimed to develop a job prioritization model tailored for recruiters, creating a novel system that matches recruiters with job vacancies based on their past experiences. With further improvements, JobRank has the potential to be integrated into the daily activities of recruiters, assisting them in making prioritization decisions and effectively increasing their efficiency.

11 Limitations and Future Work

During the course of this research, certain limitations emerged that hindered JobRank from reaching its full potential. These constraints were also evident in the survey results,

where users acknowledged the model's potential but found the rankings insufficient for their use case.

One of the limitations encountered in this research was the quality of the data provided by ManpowerGroup. Despite the organization having an extensive database, several issues compromised the effectiveness of JobRank. For instance, The dataset contained numerous custom columns that users could add, leading to a lack of standardization. These columns clouded the database with a lot of noise. This inconsistency resulted in redundant information and complicated the data cleaning and preprocessing stages. Knowledge about the data was dispersed among different individuals within the organization, complicating the identification of accurate and comprehensive information sources. The data would benefit of being regulated more strictly and not everyone should be able to create features or values. Missing values were also a problem in the dataset, with salaries of job applications not being filled in or Job titles being very noisy and hard to interpret. Ensuring uniform data entry practices across the organization is crucial to improve data quality.

Additionally, there were notable discrepancies between historical placements and the current candidate pool. This inconsistency made it challenging to accurately match recruiters' past successes with potential future candidates, affecting the reliability of the JobRank model. The assumption at the start was that dataset J would consist of all the placed jobs and R_u were the placements per recruiter. However, with further investigation it became clear that due to a mismatch in IDs a lot of jobs were not matched creating a disruption in the data. This fragmentation hindered the ability of our model to generate sufficient rankings.

The amount of data available was also challenging. After completing the cleaning processes and feature selection, the candidate pool was reduced to 1,536 jobs. Many jobs were removed during data cleaning due to inconsistencies or missing values, resulting in a significantly trimmed dataset. With a more extensive dataset, we could have gained a deeper understanding of the job data and increased the likelihood of achieving better rankings. Furthermore, the recruiter dataset also suffered from missing entries. This dataset, derived from Manpower's data architecture, was supposed to include all placements by active recruiters. However, during the prompt creation stage, we observed that a substantial amount of data was also missing from this dataset. Certain recruiters were missing previous placements, which explains why the average number of historical placements was quite low, and some recruiters were absent from the data. This incomplete dataset meant that the model lacked critical information for each recruiter and even omitted some recruiters altogether, resulting in insufficient rankings or no rankings. These data shortcomings compromised our ability to create accurate rankings and explain certain recruiters' feedback points.

To enhance our model, ensuring data quality is paramount. The data at ManpowerGroup requires a more structured approach, including clear documentation of different datasets for easy access. Standardizing and regulating data entries are essential to maintain high data quality. Regular updates are necessary to keep the dataset current and relevant. Addressing these issues could significantly improve the JobRank model, resulting in more accurate and usable rankings for each recruiter.

Other limitations arose because of the use of quite an extensive model like Llama3. Due to the model's sheer size, we were unable to run the prompts on a local device. Consequently, we had to utilize Ollama [84] for our testing, which could have resulted in different prompt outcomes. While testing on Ollama was straightforward, we experienced variations in

responses when using Llama3 on Snellius. This discrepancy could explain why certain similarity scores exceeded the 100 threshold, as the prompt responses differed between Ollama and the actual model on Snellius.
For future work the use of heavier computational power from the start is a good improvement. This way the prompts can be tested on the eventual system where the run is performed limiting the possible mismatches it can cause. The model would also benefit from a more streamlined use of an LLM. Making use of certain state-of-the-art methods to speed up generation processes like LLMA [90] or GPTcache [91]. Moreover, our model would benefit from being fine-tuned on the data present at Manpower. Due to the lack of human-labeled data, it was very difficult to fine-tune the data and apply more sophisticated metrics to the data. In this research, the data was insufficient to achieve optimal results. However, if sufficient data is gathered, it would be possible to fine-tuned a Large Language Model (LLM) specifically trained on Manpower's data. This tailored training approach could generate even more accurate and precise rankings, significantly enhancing the model's effectiveness.

References

[1] B. Hmoud and L. Várallyai, "Will artificial intelligence take over human resources recruitment and selection?," Dec. 2019.

[2] W. Albassam, "The power of artificial intelligence in recruitment: An analytical review of current ai-based recruitment strategies," International Journal of Professional Business Review, vol. 8, e02089, Jun. 2023.

[3] A. Upadhyay and K. Khandelwal, "Applying artificial intelligence: Implications for recruitment," Strategic HR Review, vol. 17, Oct. 2018.

[4] U. Karaboga and P. Vardarlier, "Examining the use of artificial intelligence in recruitment processes," Bussecon Review of Social Sciences (2687-2285), vol. 2, no. 4, pp. 1–17, Dec. 2020.

[5] E. E. Joh and W. B. White, "How we can apply ai, and deep learning to our hr functional transformation and core talent processes?" Cornell University, p. 6, Nov. 2018.

[6] D. Agarwal and et al. "Ai, robotics, and automation: Put humans in the loop." (), [Online]. Available: `www.deloitte.com/insights/us/en/focus/human-capital-trends/2018/ai-robotics-intelligent-machines.html`.

[7] J. S. Black and P. van Esch, "Ai-enabled recruiting: What is it and how should a manager use it?" Business Horizons, vol. 63, no. 2, pp. 215–226, 2020, Aritifical Intelligence and Machine Learning, ISSN: 0007-6813.

[8] G. George, M. Thomas, and M. Anson, "A systematic review of artificial intelligence and hiring: Present position and future research areas," vol. 20, pp. 57–70, Sep. 2021.

[9] A. Hemalatha, P. Kumari, and N. Nawaz, "Impact of artificial intelligence on recruitment and selection of information technology companies,"
2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), pp. 60–66, 2021.

[10] S. Zhang, Y. Tay, and L. Yao, Next item recommendation with self-attention, 2018.

[11] W. Cheng, G. Yin, and Y. Dong, "Collaborative filtering recommendation on users' interest sequences," PLOS ONE, vol. 11, e0155739, May 2016.

[12] Q. Tan and F. Liu, "Recommendation based on users' long-term and short-term interests with attention," Mathematical Problems in Engineering, vol. 2019, pp. 1–13, Oct. 2019.

[13] K. Bao, J. Zhang, and Y. Zhang, "Tallrec: An effective and efficient tuning framework to align large language model with recommendation," in Proceedings of the 17th ACM Conference on Recommender Systems, ser. RecSys '23, ACM, Sep. 2023.

[14] A. Vinogradova, Y. Fomina, and A. Gorodischeva, "Artificial intelligence capabilities classification in business environment," Journal of Physics: Conference Series, vol. 1399, p. 033 098, Dec. 2019.

[15] R. Le, W. Hu, and Y. Song, "Towards effective and interpretable person-job fitting," CIKM '19, pp. 1883–1892, 2019.

[16] C. Qin, H. Zhu, and T. Xu, "Enhancing person-job fit for talent recruitment: An ability-aware neural network approach," CoRR, vol. abs/1812.08947, 2018.

[17] M. Salehi, "An effective recommendation based on user behaviour: A hybrid of sequential pattern of user and attributes of product," Int. J. Bus. Inf. Syst., vol. 14, pp. 480–496, 2013.

[18] Z. He, W. Liu, and W. Guo, A survey on user behavior modeling in recommender systems, 2023.

[19] C. Yang, Y. Hou, and Y. Song, "Modeling two-way selection preference for person-job fit," in Proceedings of the 16th ACM Conference on Recommender Systems, ser. RecSys '22, ACM, Sep. 2022.

[20] C. de Ruijt and S. Bhulai, Job recommender systems: A review, 2021.

[21] Y. Deldjoo and M. Schedl, "Recommender systems leveraging multimedia content," ACM Computing Surveys (CSUR), vol. 53, pp. 1–38, 2020.

[22] U. Javed and K. Shaukat, "A review of content-based and context-based recommendation systems," Int. J. Emerg. Technol. Learn., vol. 16, 2021.

[23] L. M. D. Campos and J. M. Fernández-Luna, "Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks," Int. J. Approx. Reason., vol. 51, pp. 785–799, 2010.

[24] D. Goldberg, D. Nichols, and B. M. Oki, "Using collaborative filtering to weave an information tapestry," Commun. ACM, vol. 35, no. 12, pp. 61–70, Dec. 1992, ISSN: 0001-0782.

[25] R. Chen, Q. Hua, and Y.-s. Chang, "A survey of collaborative filtering-based recommender systems: From traditional methods to hybrid methods based on social networks," IEEE Access, vol. 6, pp. 64 301–64 320, 2018.

[26] J. Bobadilla and F. Serradilla, "Collaborative filtering adapted to recommender systems of e-learning," Knowl. Based Syst., vol. 22, pp. 261–265, 2009.

[27] X. Yang and Y. Guo, "A survey of collaborative filtering based social recommender systems," Comput. Commun., vol. 41, pp. 1–10, 2014.

[28] Q. Guo and F. Zhuang, "A survey on knowledge graph-based recommender systems," IEEE Transactions on Knowledge and Data Engineering, vol. 34, pp. 3549–3568, 2020.

[29] F. Cena and L. Console, "Logical foundations of knowledge-based recommender systems: A unifying spectrum of alternatives," Inf. Sci., vol. 546, pp. 60–73, 2021.

[30] Q. Chen and J. Lin, "Towards knowledge-based recommender dialog system," pp. 1803–1813, 2019.

[31] R. Burke, "Hybrid recommender systems: Survey and experiments," User Modeling and User-Adapted Interaction, vol. 12, pp. 331–370, 2002.

[32] T. K. Paradarami and N. D. Bastian, "A hybrid recommender system using artificial neural networks," Expert Syst. Appl., vol. 83, pp. 300–313, 2017.

[33] R. Burke, "Hybrid recommender systems: Survey and experiments," User Modeling and User-Adapted Interaction, vol. 12, pp. 331–370, 2002.

[34] J. Tarus and Z. Niu, "A hybrid knowledge-based recommender system for e-learning based on ontology and sequential pattern mining," Future Gener. Comput. Syst., vol. 72, pp. 37–48, 2017.

[35] P. Covington and J. Adams, "Deep neural networks for youtube recommendations," in Proceedings of the 10th ACM Conference on Recommender Systems, New York, NY, USA, 2016.

[36] Kaggle. "Kaggle: Level up with the largest ai and ml community." Accessed: 12/28/2023. (2023), [Online]. Available: https://www.kaggle.com.

[37] F. Ullah and B. Zhang, "Deep edu: A deep neural collaborative filtering for educational services recommendation," IEEE Access, vol. PP, Jun. 2020.

[38] M. Bustreo and C. Beltrán-González, "Recurrent neural networks," pp. 59–79, 2018.

[39] Z. Wu and S. Pan, "A comprehensive survey on graph neural networks," IEEE Transactions on Neural Networks and Learning Systems, vol. 32, pp. 4–24, 2019.

[40] W. Fan and Z. Zhao, "Recommender systems in the era of large language models (llms)," ArXiv, vol. abs/2307.02046, 2023.

[41] OpenAI. "Introducing chatgpt." Accessed: 12/28/2023. (2022), [Online]. Available: https://openai.com/blog/chatgpt#OpenAI.

[42] J. Miao and L. Niu, "A survey on feature selection," Procedia Computer Science, vol. 91, pp. 919–926, Dec. 2016.

[43] Wolf and Shashua, "Feature selection for unsupervised and supervised inference: The emergence of sparsity in a weighted-based approach," in Proceedings Ninth IEEE International Conference on Computer Vision, 2003, 378–384 vol.1.

[44] Z. Zhao and H. Liu, "Spectral feature selection for supervised and unsupervised learning," vol. 227, Jun. 2007, pp. 1151–1157.

[45] S. Huang, "Supervised feature selection: A tutorial," Artificial Intelligence Research, vol. 4, Mar. 2015.

[46] Z. Zhao and H. Liu, "Semi-supervised feature selection via spectral analysis," Apr. 2007.

[47] S. Solorio-Fernández and J. A. Carrasco-Ochoa, "A review of unsupervised feature selection methods," vol. 53, no. 2, pp. 907–948, Feb. 2020, ISSN: 0269-2821.

[48] E. Fowlkes, R. Gnanadesikan, and J. Kettenring, "Variable selection in clustering," Journal of Classification, vol. 5, no. 2, pp. 205–228, Sep. 1988.

[49] J. G. Dy and C. E. Brodley, "Feature selection for unsupervised learning," J. Mach. Learn. Res., vol. 5, pp. 845–889, Dec. 2004, ISSN: 1532-4435.

[50] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data mining, Inference and Prediction, 2nd ed. Springer, 2009.

[51] D. J. Stangroom, "An introduction to the use of correlation coefficients in clinical investigation," Postgraduate Medical Journal, vol. 51, no. 591, p. 693, 1975.

[52] H. Jeon and S. Oh, "Hybrid-recursive feature elimination for efficient feature selection," Applied Sciences, vol. 10, no. 9, p. 3211, 2020.

[53] W. Jerbi, N. Essoussi, and A. B. Brahim, "A review of random forest-based feature selection methods for data science education and applications," International Journal of Data Science and Analytics, vol. 13, pp. 1–20, 2022.

[54] N. Pudjihartono, T. Fadason, and A. W. Kempa-Liehr, "A review of feature selection methods for machine learning-based disease risk prediction," Frontiers in Bioinformatics, vol. 2, 2022, ISSN: 2673-7647.

[55] A. K. Jain, "Data clustering: 50 years beyond k-means," Pattern Recognition Letters, vol. 31, no. 8, pp. 651–666, 2010, Award Winning Papers from the 19th International Conference on Pattern Recognition (ICPR), ISSN: 0167-8655.

[56] Z. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," Data Mining and Knowledge Discovery, vol. 2, no. 3, pp. 283–304, 1998.

[57] H. Abdi and L. Williams, "Principal component analysis,"
Wiley Interdisciplinary Reviews: Computational Statistics, vol. 2, pp. 433–459, Jul. 2010.

[58] L. van der Maaten and G. Hinton, "Visualizing data using t-sne,"
Journal of Machine Learning Research, vol. 9, no. Nov, pp. 2579–2605, 2008.

[59] L. McInnes and J. Healy,
Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

[60] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018.

[61] J. Devlin, M.-W. Chang, and K. Lee,
Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[62] Z. Zheng, Z. Qiu, and X. Hu, Generative job recommendations with large language model, 2023.

[63] F. Yang, Z. Chen, and Z. Jiang, Palr: Personalization aware llms for recommendation, 2023.

[64] Z. Yue, S. Rabhi, and G. de Souza Pereira Moreira,
Llamarec: Two-stage recommendation using large language models for ranking, 2023.

[65] Z. Cui and J. Ma,
M6-rec: Generative pretrained language models are open-ended recommender systems, 2022.

[66] C. Raffel and N. Shazeer, "Exploring the limits of transfer learning with a unified text-to-text transformer," CoRR, vol. abs/1910.10683, 2019.

[67] L. Ouyang, J. Wu, and X. Jiang,
Training language models to follow instructions with human feedback, 2022.

[68] W. Sun, L. Yan, and X. Ma,
Is chatgpt good at search? investigating large language models as re-ranking agents, 2023.

[69] X. Ma, L. Wang, and N. Yang, Fine-tuning llama for multi-stage text retrieval, 2023.

[70] Z. Qin, R. Jagerman, and K. Hui,
Large language models are effective text rankers with pairwise ranking prompting, 2024.

[71] JobDigger. "Introducing jdco." Accessed: 12/28/2023. (2022), [Online]. Available:
`https://www.jobdigger.nl/nieuws/de-nederlandse-arbeidsmarkt-in-detail-dankzij-specifieke-jdco-classificatie/`.

[72] Microsoft. "Linkedin." Accessed: 12/28/2023. (2024), [Online]. Available:
`https://www.linkedin.com/feed/`.

[73] S. University. "Alpaca: A strong, replicable instruction-following model." Accessed: 12/28/2023. (2023), [Online]. Available: `https://crfm.stanford.edu/2023/03/13/alpaca.html`.

[74] E. J. Hu, Y. Shen, and P. Wallis, "Lora: Low-rank adaptation of large language models," CoRR, vol. abs/2106.09685, 2021.

[75] H. Touvron, T. Lavril, and G. Izacard, Llama: Open and efficient foundation language models, 2023.

[76] OpenAI. "Introducing chatgpt." Accessed: 12/28/2023. (2023), [Online]. Available:
`https://cdn.openai.com/papers/gpt-4.pdf`.

[77] R. Nogueira and K. Cho, Passage re-ranking with bert, 2020.

[78] E. Comission. "International standard classification of occupations (isco)." (), [Online]. Available:
`https://esco.ec.europa.eu/en/about-esco/escopedia/escopedia/international-standard-classification-occupations-isco#:~:text=The%20International%20Standard%20Classification%20of,08%20(dating%20from%202008)`..

[79] P. Sahoo, A. K. Singh, and S. Saha,
A systematic survey of prompt engineering in large language models: Techniques and applications, 2024.

[80] B. Chen, Z. Zhang, and N. Langrené,
Unleashing the potential of prompt engineering in large language models: A comprehensive review,
2024.

[81] AI@Meta. "Introducing meta llama 3: The most capable openly available llm to date." (),
[Online]. Available: https://ai.meta.com/blog/meta-llama-3/.

[82] R. A. et al., Palm 2 technical report, 2023.

[83] D. Rodríguez-Cárdenas and D. N. Palacio, "Benchmarking causal study to interpret large
language models for source code,"
2023 IEEE International Conference on Software Maintenance and Evolution (ICSME),
pp. 329–334, 2023.

[84] J. Morgan and M. Chiang. "Ollama 2024." (), [Online]. Available: https://ollama.com/.

[85] 1KG. "Ollama: What is ollama?" (), [Online]. Available: https://medium.com/@1kg/ollama-what-is-ollama-9f73f3eafa8b#:~:text=Ollama%20is%20an%20open%2Dsource%20project%20that%20serves%20as%20a,accessible%20and%20customizable%20AI%20experience..

[86] SURF. "Snellius." (), [Online]. Available:
https://servicedesk.surf.nl/wiki/display/WIKI/Snellius.

[87] AI@Meta. "Meta-llama/meta-llama-3-8b-instruct." (), [Online]. Available:
https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct.

[88] W. Kwon and Z. Li,
Efficient memory management for large language model serving with pagedattention, 2023.

[89] A. (available), "K-prototype clustering algorithm for segmentation of primary care patients,"
Journal Name, 2019, If available, add DOI or URL.

[90] N. Yang and T. Ge, "Inference with reference: Lossless acceleration of large language models,"
ArXiv, vol. abs/2304.04487, 2023.

[91] F. Bang, "Gpt-
cache: An open-source semantic cache for llm applications enabling faster answers and cost savings,"
Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023),
2023.

A Appendix

A.1 Glossary table

Table 11: Glossary of Equations

Equation	Description
(1)	$R_u = \{r_1 : \{F_{1,1}, F_{1,2}, \ldots, F_{1,m_1}\}, r_2 : \{F_{2,1}, F_{2,2}, \ldots, F_{2,m_2}\}, \ldots, r_n : \{F_{n,1}, F_{n,2}, \ldots, F_{n,m_n}\}\}$ Representation of user-specific feature sets for different items $r_1, r_2, \ldots, r_n$ with features $F_{i,j}$.
(2)	$J = \{j_1 : \{F_{1,1}, F_{1,2}, \ldots, F_{1,m_1}\}, j_2 : \{F_{2,1}, F_{2,2}, \ldots, F_{2,m_2}\}, \ldots, j_n : \{F_{n,1}, F_{n,2}, \ldots, F_{n,m_n}\}\}$ Representation of job-specific feature sets for different items $j_1, j_2, \ldots, j_n$ with features $F_{i,j}$.
(3)	$\max_\phi \sum_{(x,y)\in\mathcal{Z}} \sum_{l=1}^{\lvert y\rvert} \log(P_\phi(y_l \lvert x, y_{<l}))$ Maximum likelihood estimation for training the model parameters ϕ on the dataset $\mathcal{Z}$.
(5)	$\max_\theta \sum_{(x,y)\in Z} \sum_{t=1}^{\lvert y\rvert} \log(P_{\theta+e}(y_t \lvert x, y_{<t}))$ Lightweight tuning process from TALLrec for optimizing model parameters θ with embeddings e.
(6)	$\mathcal{L}_{\text{RankNet}} = \sum_{i=1}^{M} \sum_{j=1}^{M} \mathbb{1}_{d_i<d_j} \log(1 + \exp(s_i - s_j))$ Loss function for RankNet model, comparing scores s_i, s_j for items d_i, d_j.

A.2 Cluster analysis

Cluster	Categorical Highlights	Numerical Highlights	Size
0	96% KLM as corporation of which 80% Bagagemedewerker Schiphol	Having a substantial deviation of Number of openings from the mean indicating that this cluster contains more openings than the average, furthermore the Revenue per day and the number of offices have a negative deviation from the mean with about 50% less.	1339
1	92% PostNL as corporation of which 93% Postboden en postsorteerders as ISCO code niveau 4	Substantial deviation from the mean with the salary of the job being around 90% less than the average of the dataset in total, also the amount of years required to apply for the job is substantially lower than average	1595
2	Around 35% have the ISCO niveau 1 code of "Bedieningspersoneel van machines en installaties" compared to 4% in the dataset.	Seeing a negative deviation of the total mean with Number of Offices and Employees of the corporation around 80%	975
3	Around 81% of jobs having the ISCO code level 4 set as "Laders en lossers"	Having the biggest cluster size of all with a negative deviation of the total mean with the years required to apply and the Number of Employees at the Corporation.	2432
4	Very small size and contains primarily of the specific job: Klantenservice medewerker GGD (94%)	The smallest size of all clusters and a specific job highlighted with a very high deviation from the mean on several features as consequence.	161
5	Noticeable increase of jobs with ISCO code niveau 1: "Administatief Personeel" occupying 64% of the dataset compared to the 27% in the overall dataset	A positive deviation from the mean on salary (around +90%) and on average the corporation has a lot more offices than the overall mean (+212%)	1127
6	Similar to cluster 3 containing mostly "Laders en Lossers" (94%) predominantly for the company CEVA Logistics (60%)	A negative relation to the overall mean on several features including, Pay rate, Salary and the years of experience required	1576
7	Again similar to cluster 3, around 79% of jobs having the ISCO code level 4 set as "Laders en lossers"	A negative relation to the overall mean on the Salary and the Number of Offices and Employees of the corporation (Around -70%)	1933
8	Containing a lot of jobs with as JDCO code: "Unknown" (39%) compared to the 5% in the overall dataset	A positive relation to the overall mean on the Salary and Number of Offices of the corporation (Around +120%)	1287
9	Quite a random cluster with as only standout value the amount of jobs with ISCO Niveau 3: "Klantenvoorlichters" predominantly for the corporation: "UWV"	A positive and negative relation on a number of features having around +75% on features like Salary, Number of hours worked and Revenue amount and -75% on Number of Opening and Employees	1188

Table 12: In depth view on the cluster contents.

A.3 Pseudo Code

A.3.1 Generate Recommendation Algorithm

Algorithm 1 Generate Recommendation

Input: Prompt type P_t, Recruiter jobs R_u, Candidates J, Model M, Window size ws, Step s
Initialize `job_list` as empty
Initialize `yes_count` to 0
for each j in J **do**
  $W \leftarrow$ sliding_windows(R_u, ws, s)
  window_count $\leftarrow$ len(W)
  for each window w in W **do**
    `yes_count` $= 0$
    while True **do**
      $P \leftarrow$ prompt(P_t, R_u, j)
      response $\leftarrow$ LLM.generate(M, P)
      if response == "yes" **then**
        `yes_count` $=$ `yes_count` $+ 1$
      end if
      if `yes_count` $>=$ `windows_count`$/2$ **then**
        `job_list.append`(j)
        break
      end if
    end while
  end for
end for
return `job_list`

A.3.2 Generate Ranking Algorithm

Algorithm 2 Generate Ranking

Input: Ranking prompt P, Recruiter jobs R_u, Job_list J_list, Model M, Window size ws, Step s
Initialize `job_scores` as an empty dictionary
for each j in J_list **do**
  $W \leftarrow$ sliding_windows(R_u, ws, s)
  for each w in W **do**
    while True **do**
      $P \leftarrow$ prompt(P, R_u, j)
      response $\leftarrow$ LLM.generate(M, P)
      if isinstance(response, int) **then**
        score = response
        break
      end if
    end while
    `job_scores`$[j]$ = score
  end for
end for
return `job_scores`

A.4 Full prompt example

Based on the recruiter's historical placements, we aim to identify the most suitable job positions for this recruiter from a list of potential opportunities. For each candidate job presented, please respond with a clear "Yes" only if the job is a perfect recommendation for the recruiter and has extremely similar traits to the recruiter's historical placements, or "No" if it is not. Do not provide any additional feedback or comments. Limit your response to the single word 'Yes' or 'No'.

Recruiter's historical placements:
Job number: 15, **Job name:** administratief medewerker customer service, **Corporation name:** Boston Scientific International B.V., **Job Title:** Customer Care Representative Boston Scientific Kerkrade
Features of the job: Recruiter: Emily Lee, **Number of placed candidates:** 1, **Time to hire (in days):** 9.0, **Revenue gained from job:** €10246.24, **Education Degree:** MBO, **Quarter:** het tweede kwartaal van 2024, **State job is located in:** Limburg, **Cost Center:** 188 Boston Scientific Kerkrade, **Application Status:** Do not publish, **Corporation Zip code:** 6468EX, **Corporation City:** Kerkrade, **source_jobsub:** Manpower.nl, **ISCO Level 1:** Administratief personeel, **ISCO Level 2:** Administratieve medewerkers, **ISCO Level 3:** Administratieve medewerkers, algemeen, **ISCO Level 4:** Administratieve medewerkers, algemeen, **ISCO Code:** 4110, **Corporation Address:** Vestastraat, **Job Title:** Customer Care Representative Boston Scientific Kerkrade, **Job Address:** Vestastraat, **Mark up percentage:** -1.0, **Job Title length:** 55.0, **Revenue_per_day_hlre:** 301.36, **Job type:** 1, **Years Required:** 1.0, **Corporation offices:** 7.0, **Quantity_hlre:** 272.0, **Provincie_bevolking(15-65)_job:** 1824127.662915166, **Employees number:** 350.0, **Hours Per Week:** 40.0, **Salary:** €2700.0, **Job openings:** 1.0, **Cluster Category:** 7

Job number: 20, **Job name:** administratief medewerker customer service, **Corporation name:** Dow Benelux B.V., **Job Title:** Customer Service Specialist Dow Terneuzen
Features of the job: Recruiter: Emily Lee, **Number of placed candidates:** 4, **Time to hire (in days):** 22.0, **Revenue gained from job:** €33187.2, **Education Degree:** MBO, **Quarter:** het eerste kwartaal van 2024, **State job is located in:** Zeeland, **Cost Center:** 294 Dow Terneuzen, **Application Status:** Completed by CORE, **Corporation Zip code:** 4542NM, **Corporation City:** Hoek, **source_jobsub:** Manpower.nl, **ISCO Level 1:** Administratief personeel, **ISCO Level 2:** Administratieve medewerkers, **ISCO Level 3:** Administratieve medewerkers, algemeen, **ISCO Level 4:** Administratieve medewerkers, algemeen, **ISCO Code:** 4110, **Corporation Address:** Herbert H. Dowweg, **Job Title:** Customer Service Specialist Dow Terneuzen, **Job Address:** Innovatieweg, **Mark up percentage:** 0.0, **Job Title length:** 41.0, **Revenue_per_day_hlre:** 553.12, **Job type:** 1, **Years Required:** 0.0, **Corporation offices:** 25.0, **Quantity_hlre:** 960.0, **Provincie_bevolking(15-65)_job:** 1824127.662915166, **Employees number:** 0.0, **Hours Per Week:** 40.0, **Salary:** €2800.0, **Job openings:** 5.0, **Cluster Category:** 7

Job number: 21, **Job name:** administratief medewerker customer service, **Corporation name:** FedEx Express Netherlands B.V., **Job Title:** Allround administratief logistiek medewerker FedEx Schiphol-Rijk

Features of the job: Recruiter: Emily Lee, **Number of placed candidates:** 1, **Time to hire (in days):** 15.0, **Revenue gained from job:** €21967.84, **Education Degree:** MBO, **Quarter:** het derde kwartaal van 2023, **State job is located in:** Noord-Holland, **Cost Center:** 466 FedEx Schiphol-Rijk, **Application Status:** Do not publish, **Corporation Zip code:** 2132LS, **Corporation City:** Hoofddorp, **source_jobsub:** Manpower.nl, **ISCO Level 1:** Administratief personeel, **ISCO Level 2:** Administratieve medewerkers, **ISCO Level 3:** Administratieve medewerkers, algemeen, **ISCO Level 4:** Administratieve medewerkers, algemeen, **ISCO Code:** 4110, **Corporation Address:** Taurusavenue, **Job Title:** Allround administratief logistiek medewerker FedEx Schiphol-Rijk, **Job Address:** Bellsingel, **Mark up percentage:** 0.0, **Job Title length:** 64.0, **Revenue_per_day_hlre:** 244.0871111111111, **Job type:** 1, **Years Required:** 0.0, **Corporation offices:** 32.0, **Quantity_hlre:** 688.0, **Provincie_bevolking(15-65)_job:** 1955288.0, **Employees number:** 3078.0, **Hours Per Week:** 40.0, **Salary:** €2400.0, **Job openings:** 1.0, **Cluster Category:** 7

Job number: 23, **Job name:** administratief medewerker customer service, **Corporation name:** bol.com B.V., **Job Title:** PV Bol.com 30 oktober
Features of the job: Recruiter: Emily Lee, **Number of placed candidates:** 2, **Time to hire (in days):** 47.0, **Revenue gained from job:** €110306.0, **Education Degree:** MBO, **Quarter:** het derde kwartaal van 2023, **State job is located in:** Utrecht, **Cost Center:** 346 Bol.com, **Application Status:** Do not publish, **Corporation Zip code:** 3528BJ, **Corporation City:** Utrecht, **source_jobsub:** Indeed Direct Apply, **ISCO Level 1:** Administratief personeel, **ISCO Level 2:** Administratieve medewerkers, **ISCO Level 3:** Administratieve medewerkers, algemeen, **ISCO Level 4:** Administratieve medewerkers, algemeen, **ISCO Code:** 4110, **Corporation Address:** Papendorpseweg, **Job Title:** PV Bol.com 30 oktober, **Job Address:** Papendorpseweg 100, **Mark up percentage:** 0.0, **Job Title length:** 21.0, **Revenue_per_day_hlre:** 343.5, **Job type:** 1, **Years Required:** 0.0, **Corporation offices:** 1471.0, **Quantity_hlre:** 3355.0, **Provincie_bevolking(15-65)_job:** 908964.0, **Employees number:** 258.0, **Hours Per Week:** 40.0, **Salary:** €14.0, **Job openings:** 5.0, **Cluster Category:** 7

Candidate Job:
Job number: 0, **Job name:** .net architect, **Corporation name:** BMN Bouwmaterialen B.V., **Job Title:** Plaatsingsvacature BMN Groningen
Features of the job: Education Degree: MBO, **Quarter:** het tweede kwartaal van 2022, **State job is located in:** Groningen, **Cost Center:** 331 Groningen, **Application Status:** Do not publish, **Corporation Zip code:** 3439NC, **Corporation City:** Nieuwegein, **source_jobsub:** Eigen Database, **ISCO Level 1:** Intellectuele, wetenschappelijke en artistieke beroepen, **ISCO Level 2:** Specialisten op het gebied van informatie- en communicatietechnologie, **ISCO Level 3:** Software- en applicatieontwikkelaars en -analisten, **ISCO Level 4:** Systeemanalisten, **ISCO Code:** 2511.0, **Corporation Address:** Celsiusbaan, **Job Title:** Plaatsingsvacature BMN Groningen, **Job Address:** Gotenburgweg, **Mark up percentage:** -1.0, **Revenue gained from job:** €236836.76, **Job Title length:** 32.0, **Revenue_per_day_hlre:** 395.58, **Job type:** 0.0, **Years Required:** 0.0, **Corporation offices:** 142.0, **Quantity_hlre:** 6977.10, **Provincie_bevolking(15-65)_job:** 388059.0, **Employees number:** 2.0, **Hours Per Week:** 40.0, **Salary:** 0.0, **Time to hire (in days):** 109.2, **Job openings:** 1.0, **Cluster Category:** 7.0

Do you recommend this candidate job to the recruiter? Format your response by responding with just 1 word: Yes or No