AI4HR Recruiter: a job recommender system for internal recruitment in consulting companies

François Mentec, Zoltan Miklos, Sébastien Hervieu, Thierry Roger

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 225 (2023) 1231–1240

Procedia Computer Science

www.elsevier.com/locate/procedia

27th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2023)

AI4HR Recruiter: a job recommender system for internal recruitment in consulting companies

François Mentec[a,b,*], Zoltan Miklos[a], Sébastien Hervieu[b], Thierry Roger[b]

[a] *Univ Rennes CNRS IRISA, 263 Av. Général Leclerc, 35000 Rennes, France*
[b] *ALTEN, 12 Rue du Patis Tatelin, 35000 Rennes, France*

Abstract

Recruiting job candidates is a crucial activity for consulting companies to fulfill their consulting activities for customers or internal projects. Due to the complexity of the recruitment process, companies adopt standardized resume formats to be able to identify or compare candidates. However, this process requires significant involvement from human resource specialists and project leaders. In this paper, we present AI4HR Recruiter, a recommender system designed to improve the internal recruitment process of consultants. The system uses artificial intelligence-based techniques, specifically pre-trained sentence transformers. It aims at supporting recruiters in identifying and selecting candidates, rather than automating the process. We measure improvement in efficiency in terms of gains in recruiters time induced by the recommendation explanation presence. We also measure the impact on users' satisfaction of two different types of algorithm to generate recommendations. To do so, we have been conducting a two fold study in a real-life setting for six months, where on the one hand, telemetry data was gathered while recommendation process user interaction was being performed and on the other hand, surveys about users' satisfaction were conducted afterward. Our results evaluation demonstrates the effectiveness of the system in assisting recruiters in candidates selection process.

Keywords: Job Recommender System

1. Introduction

Organizing internal projects is challenging task for consulting companies. By the very nature of the consulting business, consultants no longer working for customer projects become available for use by internal projects, usually for a short period of time and at short notice. Internal recruiting tasks to allocate these engineers are complex and costly and require supporting tools.

In the context of this recruitment process, one can assume that every candidate has a digital resume in a standardized format that contains specific qualifications and job experiences. This format is specific to the company, but it is

* Corresponding author. Tel.: +33-689-377-879
E-mail address: francois.mentec@alten.com

1877-0509
10.1016/j.procs.2023.10.111

common for consulting companies to have their own structured resume format. The availability of these standardized resumes opens the way to design recommendation services for this internal recruitment process without the need for complex parsing and information extraction tools. In this paper, we propose a artificial intelligence-based recommendation tool that supports this specific recruitment. This tool's goal is to help recruiters efficiently explore available candidates, and in no way automate the recruitment process. Our tool has already been in use for 6 months, and the data generated continues to grow each week.

Traditional recommendation techniques for products or services are not well suited to all job recommendations. The main reason is the difference in items availability: while goods and digital content can be replicated to be distributed to multiple people simultaneously, this is not the case for employees who can be allocated to only one full-time occupation at a time. Thus well-known collaborative filtering techniques based on items' popularity among similar users can create a lot of competition in a recruitment context. Content and knowledge-based techniques are better suited to our context (Section 3) in which users are to collaborate to hire a maximum of the available talents.

In this paper, we present AI4HR Recruiter, a tool designed to support users in a specific internal recruitment process. Our methods exploit resumes content with the help of transformer-based embeddings. Our recommendation techniques are content-based and provide the user with an explanation of the recommendation. These explanations highlight parts of the resume that are relevant to the user. Users can provide explicit feedback to the system through these explanations. This feedback is effectively used by the recommender system to improve recommendations. The tool also collects implicit feedback (e.g. recording of actions such as opening a resume), so we can analyze usage and further improve the system.

We evaluate our methods and tools in a real context. Our tool is in use in a real (internal) recruitment process in the consulting company ALTEN. Our experiments show that recommendations explanation translates into a significant gain in efficiency. We also compare a embedding-based recommendation techniques to a keyword-based search. Our results show that users prefer the embedding-based recommender system.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 presents the context and business process we aim to improve through recommendations. Section 4 discusses our recommendation techniques and our AI4HR Recruiter tool. Section 5 present our experimental evaluation, and Section 6 concludes the paper.

2. Related work

Recommender Systems are tools that enhance processes for making choices, by providing suggestions about goods and services to users. The term was introduced by Resnick et al.[14] back in 1997, but this type of systems were in use much earlier. For example, Golberg et al. [5] present such a system using the Collaborative Filtering algorithm. This kind of algorithm is one of the most widely-used techniques to power recommendation services [1]. Recommender Systems can be classified into 4 categories [2]:

- **Collaborative Filtering** systems rely on implicit and explicit feedbacks from users.
- **Content-based** systems rely on items attributes/features. The recommender systems featured in AI4HR Recruiter fits in this category.
- **Knowledge-based** systems rely on a knowledge base, such as an ontology, for deriving the recommendation.
- **Hybrid** systems rely on a combination of the above categories.

Freire et al.[3] survey job recommendation techniques, they emphasize that these systems often rely on a variety of machine learning techniques and do not fit well into the usual recommendation system categories. They suggest to use an "Other Techniques" category. Another recent survey on Job Recommender Systems includes the work by De Rujit et al.[15]. They are critical of the "Other Techniques" category of Freire et al.[3] and propose to regroup this Other Techniques category with the Hybrid category, but divide them into Monolithic and Ensemble categories, following the classification of [2]. Monolithic recommender systems do not include at least one subsystem which could produce recommendations on its own. Ensemble recommender systems include at least one subsystem which could produce recommendations on its own.

In the literature, publications from the industry can be found, like Lavi et al.[7] from Randstad, in which embeddings produced by a Transformer to match candidates to vacancies are used. They also highlight some problems in the field like risk of discrimination, or the vocabulary gap between Resume and Job Offer. Like most industrial publications in this field they do not present the details about how they train their model. To the best of our knowledge, the vocabulary gap between resumes and job offers was first identified by Schmitt et al. [16] and its importance depends

of the job/field. Because we only use resumes and small queries from project leaders that use a technical vocabulary similar to the one used by candidates, we disregarded this issue in our work. Similarly to Lavi et al.[7] we use embeddings, but unlike them, we do not train them ourselves and instead rely on pre-trained sentence-transformer models. However, we evaluate which model is the most suited to our task using a dataset built from user feedback gathered in AI4HR Recruiter (Section 4.4.2).

Wenxing Hong et al. [6] propose a system that classifies users and adapts its recommendation strategy based on the three types of users. First, there are Proactive users, who make requests and engage in various activities referred to as active. Next, there are Passive users, they mostly look at positions that are popular among other users. Finally, there are Moderate users, this is an in-between of the two others. The authors use a content-based strategy for proactive users, a collaborative-filtering one for passive users, and for moderate they use HyR which is a combination of collaborative filtering and knowledge-based. If we apply this user classification to our own specific business process (Section 3), we only have proactive users, so our system should relies on content-based strategies.

Explanation of decisions made by an algorithm (i.e. recommendations), is an essential part of human/machine interactions. As presented by Nava Tintarev et al. [17], in the case of Recommender Systems the usage of explanation can aim at improving: transparency, scrutability, trustworthiness, effectiveness, persuasiveness, efficiency, or even satisfaction . It is neither possible nor desirable to pursue all of these at once. In our work we have been focusing on scrutability by allowing users to provide feedback on the explanation, and efficiency by highlighting fields of the resume that are the most relevant to the user.

Freire et al. [4] propose a framework for a Job Recommender System that ranges from preferences acquisition, to the collection of feedbacks and their integration in the recommendation process. Their system is designed for both recruiters and candidates whereas ours is focused solely on recruiters.

3. The ”Mercato” process

In this section, we present the context and the business process in which AI4HR Recruiter is used. The details of this process are specific to the ALTEN company, however, other companies can have similar processes. At ALTEN, consultants work on missions for customers, but when there isn't any well-suited mission available for a consultant, ALTEN then tries to assign the consultant to an internal mission. These missions goals are to perform tasks for the company and serve as a training opportunity during which consultants can acquire competencies, strengthen their skills and gain valuable experience. Consultants without a customer mission are placed in a pool of available consultants. Project leaders hire consultants from this pool during a weekly event named ”Mercato” during which they look at available consultants resumes and express preferences as to whom they want for their projects. At the end of the preferences expression phase, managers use these preferences to assign consultants to project leaders.

The company wants a maximum of consultants engaged in a mission, and ideally, none should be left unoccupied. To achieve this goal, managers need as many wishes from project leaders as possible. However, looking at every consultant's profile (about a hundred each week) can be time-consuming for project leaders who have other commitments, this is where a recommender system can be useful by only recommending consultants relevant to their projects needs, hence reducing the time needed to perform their part of the ”Mercato” process.

In general, there is a conflict between the goal of the process and the goal of the recommender system: the company wants a maximum of consultants occupied, so managers need every consultant to have a preference from a project leader. However, the recommender system wants to reduce the pool of consultants to be presented to a project leader, hence lowering the number of preferences produced. To maximize the number of preferences we want a system with high recall, but in order save time we want a system with high precision. Ideally, our system should have both, but if we have to choose, we should favor the recall. Furthermore, reducing the pool of consultants presented to project leaders is not the only way we can lower the time required from them to select consultants. Rather than reducing the number of recommendations, we can reduce the time a project leader allocates to each recommendation by providing insightful explanations to the recommendation.

Finally, we wish to insist that our recommender system does not automate the recruitment process. Its goal is to help project leaders to identify relevant candidates and to improve the internal recruitment process.

4. The AI4HR Recruiter Recommender System

This section describes AI4HR Recruiter and its recommendation strategies. Here is a non-exhaustive list of its features: user's preferences acquisition (section 4.2), exploration of the catalog of consultants, consultation of a con-

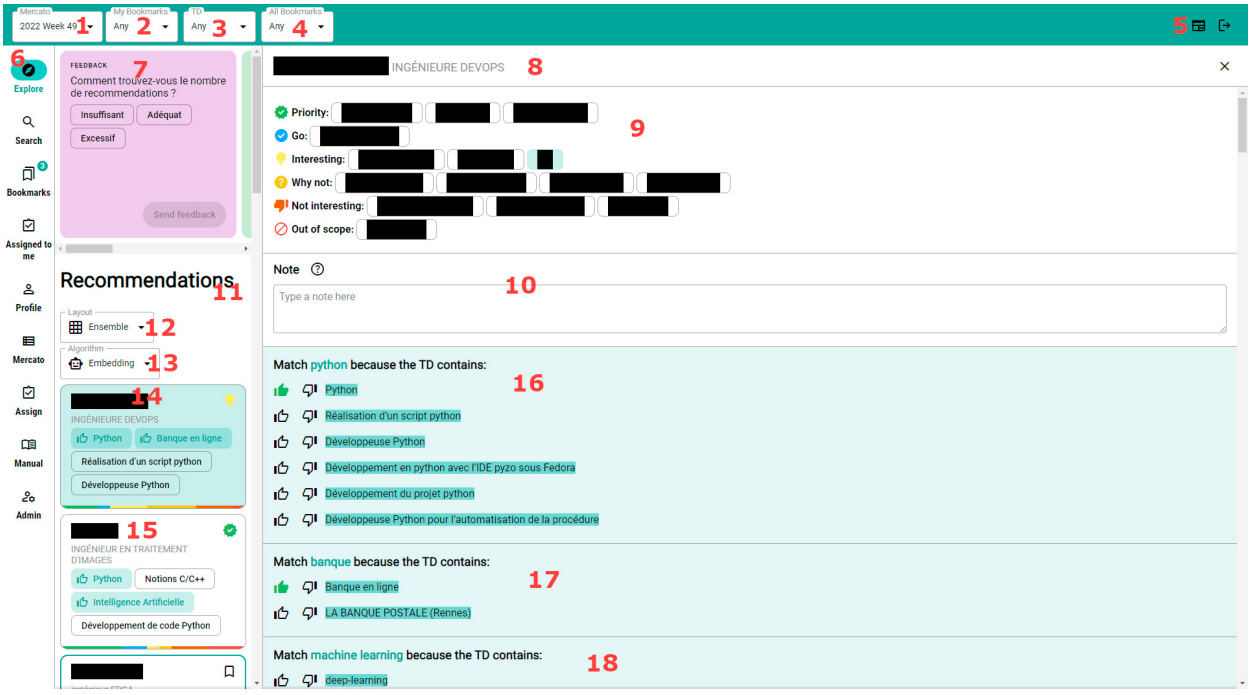


Fig. 1: The interface of AI4HR Recruiter.

- **1** is a dropdown menu to select the week to display.
- **2**, **3**, and **4** are filters users can use to hide some consultants.
- **5** is next to a button that opens a dialog informing users of the newer functionalities and another button to log out of the tool.
- **6** is a navigation rail to switch between the various panels. Currently, we are in the explore panel that mostly contains recommendations.
- **7** is a card that asks a question to the user. We use those to retrieve explicit feedback for this study. It is significantly more efficient at surveying users than asking them to fill out a form in a separate tool.
- **8** is a side panel showing the selected consultant. The black rectangle hides the consultant's name for confidentiality. The side panel is closed if there isn't any consultant selected. In this case, the main panel fills the entirety of the screen. It contains various information about the consultant, including his resume (you need to scroll down past the bookmarks and recommendations to see it).
- **9** is a section containing all of the consultant's bookmarks. The black boxes hid the name of the users who made them.
- **10** allows the user to add a note to the consultant. This note is visible to the managers and is used to make the final assignment. Typically, a user would use this note to explain to the managers why they should assign the consultant to him.
- **11** is the list of recommended consultants.
- **12** is a dropdown menu to select the layout used to display recommendations. "Carousel" make a sub-list (carousel) for each user's interests so a consultant can appear in multiple lists. "Ensemble" combine all the interests in the same list, so a consultant only appears once.
- **13** is a dropdown menu to select the recommendation algorithm. Its value can be enforced and hidden from the user for experiments.
- **14** and **15** are consultants cards. They contain the name of the consultant (hidden by a black box), the title of the consultant (i.e. his typical job), the bookmark button, and a list of chips that represent fields that match some of the user's interests.
- **16**, **17**, and **18** are recommendations and explanations for the consultant. In this example, the system recommends the consultant for "python", "banque", and "machine learning".

sultant's profile, search the catalog of consultants, receive recommendations, bookmark consultants (see section 4.5), gather explicit and implicit feedback from users. The interface of the tool is presented in Figure 1.

To obtain recommendations, users first define their interests. We discuss this preference acquisition phase in section 4.2 while section 4.3 presents our recommendation algorithms and techniques. Section 4.4 describes the explanation of the recommendation, how we use this explanation to gather explicit feedback from users, and how we leverage those feedback to improve the recommendation. Finally, in section 4.5, we describe how users specify which consultant they wish to hire using bookmarks.

4.1. Structured Resume

As mentioned in the introduction, ALTEN uses a structured resume format named Technical Document (TD). It is a rich format designed to hold any information that can be potentially useful to staff a consultant. It contains every formation, from a university degree to a MOOC (Massive Open Online Course) certificate, with various information

Table 1: A comparison of the number of recommendations produced by each algorithm. The number of events indicates the number of recommendation processes registered by the tool. A recommendation process is initiated every time a user connects to the tool or changes his list of interests. Every other metric is calculated based on what was available when the recommendation was made.

Metric	Keyword	Embedding
# Events	441	2342
Mean catalog size	50.52	51.59
Mean # interests	6.46	6.27
Mean # recommendations	37.78	62.85
Mean # recommendations per interests	5.85	10.03
Mean # unique recommendations	18.36	29.86
Mean % of the catalog as unique recommendations	36%	58%

like the school, the university course, or the year of graduation. It contains multiple lists of skills. One is dedicated to known languages and includes the proficiency level in the language. The format also contains a list of professional experiences. Each experience has a title, a period, the position held, the goal (e.g. designing an API for an e-commerce platform), the context (e.g. the IT department of a sportswear company), a list of tasks accomplished during the experience, and a list of skills used.

For the recommendation, we reduce a TD to a list of fields. A field can contain any information from the original resume in a natural language format (e.g. a task from an experience or a known language with its level of proficiency). The structure is lost in the process: we do not retain to which experience a task or a skill belongs. Every field is considered equally during the recommendation process. In other words, to our recommender system, a resume is reduced to a list of texts (word/sentence/paragraph) that we call fields.

The motivation to reduce a resume to a list of texts for the recommender system is that it can be adapted to any format with relative ease without needing an underlying structure. It is important because different companies use different resume formats. The trade-off is that we lose some information carried by the structure in the process.

Our solution could be improved upon by weighting the fields. A weight could be determined by the nature of the field (a position may be more important than a skill), the date of the experience (a recent experience may be more important than an older one), or a combination of both. We do not explore this idea in this publication.

4.2. Preference Acquisition

The first time a user opens AI4HR Recruiter, the tool asks him to fill his profile with his interests so the system can start making recommendations. Interests are texts the user can write as he pleases without any restriction or indication. According to Narducci et al. [9], leaving the user to freely chose his preferences can lead to better recommendations. A user can have as many interests as he wants. We discuss how users effectively use interests in section 5.1.

4.3. Recommendation/Search Algorithms

Once the tool acquires user preferences, it can make recommendations. AI4HR Recruiter has two content-based recommendation strategies: keyword-based and word embedding-based. These recommendation techniques do not suffer from the cold start problem.

Both recommender systems associate a score with every consultant for every user's interests. The tool shows every consultant with a score greater than 0 to the user. There is no limit to the number of recommended consultants other than the catalog size. The tool present recommended consultants in descending order based on their scores.

Table 1 presents a variety of metrics about the number of recommendations made. Because the catalog size and the mean number of interests are very similar between both algorithms, results are easily comparable. The keyword algorithm makes fewer recommendations (37.78) than the embedding algorithm (62.85), because it is more restrictive. The number of recommendations made by the embedding algorithm is higher than the catalog size because the systems can recommend the same consultant for multiple interests.

There are two layouts to present recommendations to the user. The first one is the carousel layout in which a carousel of consultants is made for each interest. This layout can show the same consultant multiple times. When a user clicks a consultant from a carousel, only the explanation of the recommendation for the interest associated with the carousel is shown. This cause an issue because users rely a lot on the recommendation explanation to make their

decision. So a consultant rejected in a carousel may be hidden and hence missed in a later carousel despite being a good match.

The second layout (Figure 1) is "ensemble". There is only one list of consultants, and when the user clicks a consultant he is presented with explanations for every interest the consultant was recommended for. This layout reduces the number of recommendations presented to the user ("Mean # unique recommendations" instead of "Mean # recommendations" in Table 1), and when reviewing a consultant he is presented with every explanation relative to the consultant. The recommendations explanations are sorted in descending order based on the score of the corresponding recommendation.

4.3.1. Keyword Algorithm

The keyword algorithm tokenizes the query and counts occurrences of the tokens in the Resume. For a recommendation, the query is an interest of a user. The more occurrences, the higher the Resume is ranked. The tokenization split the query into keywords based on the presence of white space and quotes (words inside quotes are not separated). If a query contains multiple keywords but does not contain any words inside quotes, the system will also search for occurrences of the entire query: for the query *Artificial intelligence* it looks for *artificial*, *intelligence*, and *artificial intelligence*. This algorithm is case insensitive. This algorithm also uses a filter based on explicit feedback from the user. More about that in section 4.4.1.

4.3.2. Embedding Algorithm

In Natural Language Processing, embeddings are a way to represent text in a vector of numbers with a fixed size. Those embeddings usually carry semantic information and permit the use of mathematical functions. It is common to compare two embeddings semantically using the cosine similarity. The first two embedding techniques to become very popular in the field are Word2Vec [8] and GloVe [10]. Nowadays, embeddings based on neural networks are the most popular. The sentence-transformers framework makes more than a hundred pre-trained embedding models readily available, all based on the work of Reimers et al.[12, 13] with sBERT and they are the only models we experimented. You can find an evaluation of 30 of those models on our data in Section 4.4.2.

The embedding algorithm embeds the query and the fields of the resume and looks for fields that have a similar embedding to the one of the query. Two embeddings are similar if their cosine similarity is greater or equal to a threshold. We choose the embedding model and compute the threshold using explicit feedback from users, more about that in section 4.4.2. The consultant with the most similar field to the query is ranked the highest. The case sensitivity of this algorithm depends on the embedding model.

Similarly to the keyword algorithm, this algorithm also uses a filter based on explicit feedback from the user. More about that in section 4.4.1.

4.4. Explanation of Recommendation

The explanation is a list of fields that lead to the consultant recommendation (Figure 1). We propose this explanation of the recommendation with the goals to improve **Efficiency** and **Scrutability**. We highlight relevant fields to reduce the time a recruiter spends reading resumes, we experiment on this in section 5.3.1. And we give users the possibility to like/dislike highlighted fields to improve recommendations and build a dataset to evaluate models.

An explanation highlights a maximum of 6 fields. Highlighting more could overwhelm the user. For the keyword algorithm, the system selects fields that contain the most occurrences of the query's keywords. For the embedding algorithm, the system selects fields that are the most similar to the query (defined in section 4.3.2).

4.4.1. Feedback Filter

As illustrated in Figure 1 (elements 16, 17, and 18), users can like/dislike fields in the explanation. When it happens, feedback is registered in the database and provides us over time with a dataset of items composed of: the **query** for which the system recommended the consultant, the **field** for which the user provided feedback, if the field was **liked or disliked**, the **user** who provided the feedback, and the **recommender system** that made the recommendation. The recommender system uses this feedback to provide an additional filter on top of the keyword and embedding algorithms. If a user disliked a field for a given query, we filter out the field without running it through the keyword/embedding algorithm. On the other hand, liked fields are further highlighted in the explanation by being shown as such. Feedback is tied to users because we hypothesized that different users may associate different meanings to the same query. A user can use the query "bank" to find bankers, while another user may use it to find developers of banking software.

Table 2: Evaluation of embedding models. F1 stands for F1-score, Acc for Accuracy, Pre for Precision, and Rec for Recall. Models are sorted from best F1-score to worst. We add the keyword algorithm for comparison. This evaluation is detailed in Section 4.4.2. In our experiments we evaluated 30 models, but in this table we only present the top three.

Model	F1	Acc	Pre	Rec
multi-qa-MiniLM-L6-cos-v1	**76.5**	**80.0**	**81.3**	**72.3**
paraphrase-MiniLM-L6-v2	72.5	76.7	77.3	68.2
all-MiniLM-L12-v2	74.8	78.7	79.8	70.4
keyword	68.0	74.4	77.7	60.4

Table 3: Comparison of the performances of Recommender Systems. W stands for Week Number, G stands for group, F1 stands for F1-score, Acc for Accuracy, Pre for Precision, and Rec for Recall.

W	G	Algorithm	F1	Acc	Pre	Rec
36	A	keyword	50.4	46.8	35.1	89.5
37	A	embedding	46.8	41.9	30.7	98.6
37	B	keyword	40.2	55.8	27.2	76.7
36	B	embedding	34.7	44.8	21.1	97.2

4.4.2. Embedding models evaluation

We use the feedback dataset created by the users actions of liking/disliking fields of the explanation (section 4.4.1) to evaluate 30 embedding models and find their optimal cosine similarity threshold. These models were chosen for being the most liked sentence transformers models on HuggingFace Model Hub.

Each model embeds the query and the field of every feedback then their cosine similarity is computed. Finally, we train a linear regression model to predict if the pair is liked or disliked using only the cosine similarity. For the keyword algorithm, we use the number of occurrences in the field of keywords extracted from the query instead of cosine similarity. It is essentially a binary classification on a single parameter. We can determine the optimal threshold to use with our embedding algorithm (section 4.3.2) by looking at the weight and bias of the linear model. Performances of the top 3 models can be found in table 2 along with our keyword algorithm.

4.5. Bookmarks

Users can bookmark consultants to express their preferences as to whom they want working on their projects via a button in the top right corner of a consultant card (see 14 and 15 in Figure 1). Managers use these bookmarks to assign consultants to project leaders. There are seven types of bookmarks:

- **Priority**: the user wants the consultant absolutely.
- **Go**: the user wants the consultant but it is not critical.
- **Interesting**: the user is interested in the consultant.
- **Why not**: the user is not so interested in the profile, but since the company wants them to hire as many consultants as possible, he could envisage taking him if nobody else has a job for him.
- **Not interesting**: the user is not interested in the consultant and does not have a job for him.
- **Out of scope**: the profile is unrelated to the user's activities.
- **Missing information**: there is not enough information in the consultant's profile to decide.

A user can change (e.g. from "Priority" to "Go") or delete a bookmark at any time. For our research, we always register an event when a user creates, changes, or deletes a bookmark. This event stores the user, the consultant, the bookmark type, the moment in time, and the card's location in the tool. If its location is the explore panel, we know it comes from a recommendation, and we register the interest leading to the recommendation along with the explanation. If it comes from the search panel, we register the search query if there is one. These events are part of implicit feedback produced by users' behaviors.

We consider "Priority", "Go", "Interesting", and "Why not" bookmarks as "positive bookmarks". They indicate that the user wishes to hire the consultant). We consider "Not interesting", and "Out of scope" bookmarks as "negative bookmarks". They indicate that the user does not wish to hire the consultant. A "negative bookmark" does not always denote a bad recommendation. The user may reject the consultant because he already has enough consultants to work on his projects. He cannot manage an infinite number of consultants. "Missing information" is a special case, it indicates that the consultant's resume is missing or not filled out properly.

5. Experimental Evaluation

In this section, we evaluate how AI4HR Recruiter performs and analyze how users effectively use it. The results presented here are based on 6 months of active usage, during which the tool has seen: 61 users (not all active recruiters), 1,700 consultants, 9,000 bookmarks, 3,500 explicit feedbacks, 69,000 implicit feedbacks.

The number of users corresponds to the total number of people with an account. Some are not even allowed to access the tool: account creation happens during the first connection, but it cannot access until an administrator approves it. Some users are inactive, some are recruiters, some are managers, and some are people who need access to present the tool to others inside the company. For each experiment, we specify the exact number of active users involved.

We start by looking at users' preerences (interests), how many they have, and how they are formatted (section 5.1). Then we look at the time users spend on the tool (section 5.2). And finally, we performed two crossover trials, one on the impact of the explanation (section 5.3.1) and one on the differences between the keyword and the embedding recommender system in terms of performance (section 5.3.2).

5.1. Analysis of users' preferences

For 23 users we have a total of 268 interests. The average number of interest an user has is 11.65, the median is 11, the minimum is 4 (so every user filled their interests), and the maximum is 24. An interest has an average length of 1.4 words, the median is 1, the maximum is 7. So most interests are a single keyword, usually a skill of a field. The most common interest is "Python" with 14 appearances, followed by "Java" with 5 appearances. The computation of the number of appearances is case insensitive: For example "DevOps" is written 3 different ways: "DevOps", "devops", and "Devops". If an interest is included in another interest but is not alone, it doesn't count as an appearance. For example, "java" is included in the interest "java eclipse" but it is not counted as a "java" appearance.

5.2. Time spent on AI4HR Recruiter

We first look at user events during a Mercato. The average duration between the first event of a user and its last is 1 hour 2 minutes and 15 seconds, but it is not representative of the time spent in the tool. Some users perform the Mercato in one go without interruption, while others do it in multiple sessions. Those users may or may not close AI4HR Recruiter between sessions. Other behaviors we notice are doing a quick check at the end of a Mercato or stopping right at the start of a session: the user opens the tool, barely does anything then comes back much later.

To get a better idea of the time spent on the tool, we look at the duration between two consecutive events. We find a mean duration of 1 minute and 19 seconds. The maximum duration between two consecutive events is more than 6 hours. We are confident users do not look at the same resume for that long. So we arbitrarily define a maximal duration between events of 3 minutes. If the duration between an event and the next is longer than 3 minutes, the user took a pause and it is now a different session. Using this method, we find a mean of 2.87 sessions per Mercato per user, with an average session duration of 4 minutes and 51 seconds and an average total duration of 13 minutes and 56 seconds.

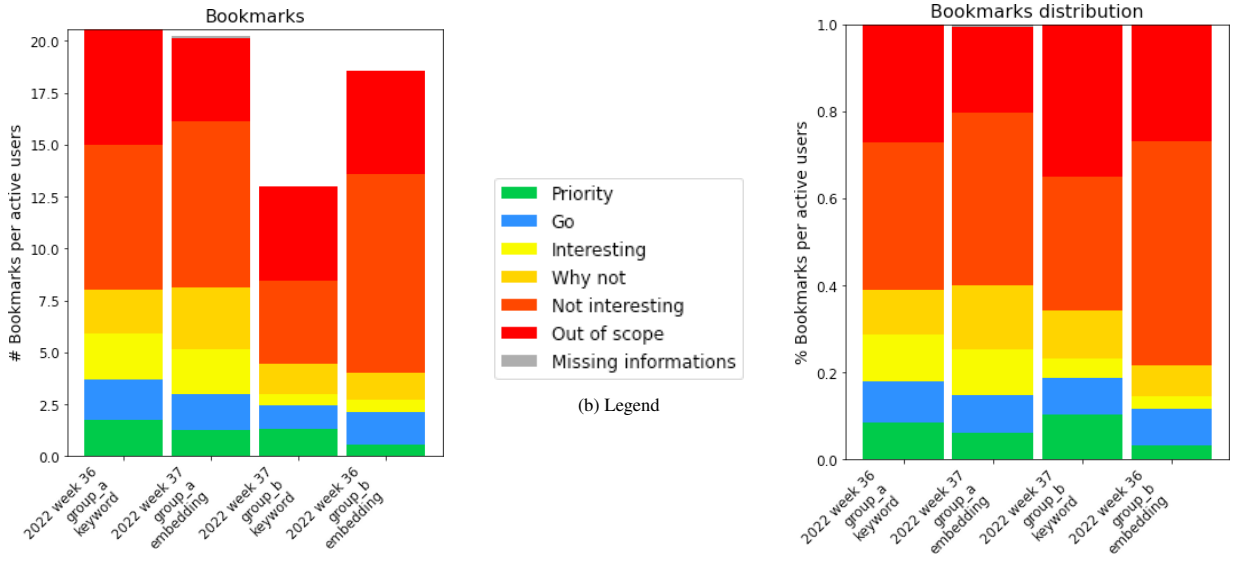
5.3. Crossover trials

To ensure our experimental results are not too heavily affected by our restrained pool of users and our ever-changing catalog, we conduct our study as a crossover trial. Our user population and trial period are both split in two. A group will test one feature during the first part of the trial, while the other group can test a second feature we wish to compare with the first. Then we swap both groups during the second part of the trial. So each group is its own control group, and doing so should ensure results when comparing two features are independent of the groups and catalog compositions.

5.3.1. Explanation

During this experimentation, we look at efficiency using the average time a user spends on a resume, Trustworthiness using the ResQue framework [11], and Satisfaction using both ResQue and asking the question inside of the tool. The ResQue evaluation framework from Pearl Pu et al. [11] evaluates the experience a user has with a Recommender System. This framework is composed of 31 questions (e.g. "I became familiar with the recommender system very quickly.") spread through 15 constructs (e.g. "Perceived Ease of Use"). Users answer questions using a 5-point Likert scale ranging from "strongly disagree" (1) to "strongly agree" (5). Following the authors' advice, we ask only one question per construct to make the form faster to fill.

For this experiment we use the keyword algorithm (Section 4.3.1) which can be explained by showing the user which fields contain occurrences of tokens from the query and highlights them in the document. Thirteen users participated in both weeks. The overall time spent on resumes is significantly higher during the second week, regardless of the groups. The main factor is probably the second week having more consultants. Because of it, recruiters could tire during the process and take more time. Also, a user notified us the tool got slow (the server lacked memory, we



(a) Number of bookmarks by mercato and by group

(b) Legend

(c) Percentage of bookmarks by mercato and by group

Fig. 2: Number/Percentage of bookmarks per bookmark's type, by mercato (year and week), by group, and by algorithm.

solved the issue by increasing memory from 4GB to 8GB). Nonetheless, we can notice that in both weeks the group with an explanation spent significantly less time looking at a resume than the group without an explanation.

We now look at the results of the ResQue framework. 14 users answered the quiz for the first week, 9 for the second, and 7 of them answered both weeks. Using Student's t-test we determine there is no significant statistical difference for any questions between the groups. The results are similar for a t-test with paired samples using only the 7 users who answered both weeks. So despite a significant time gain, users did not perceive any meaningful difference. However, the number of users who answered the quiz in both weeks may not be enough. It was difficult to have them answer a long quiz outside of the tool, and it required asking them over email many times.

5.3.2. Recommender Systems Comparison

During two weeks we split our pool of users in two groups of 13 people, one with the keyword algorithm and another with the embedding one. Users with the keyword algorithm during the first week switched to embedding during the second week and vice versa. The algorithm an user has is hidden to him. Because the tool is important to ALTEN functioning, users have the possibility to opt out of the experiment at any point. None chose to do so.

To evaluate our models we want to compute their precision, recall, accuracy and f1-score. To do so we need true positive, false positive, true negative and false negative. Here is how we proceed:

- **True Positive:** # consultants recommended and bookmarked positively.
- **False Positive:** # consultants recommended and bookmarked negatively, or clicked but not bookmarked.
- **True Negative:** # consultants not recommended and bookmarked negatively, or clicked but not bookmarked.
- **False Negative:** # consultants not recommended and bookmarked positively.

Positive bookmarks are: "Priority", "Go", "Interesting" and "Why not". Negative bookmarks are: "Not interesting" and "Out of scope". Bookmarks "Missing information" are considered neutral hence ignored. It is important to consider consultants that have been clicked but not bookmarked, because users do not take the time to negatively bookmarks consultant they are not interested in. For example between 8% and 19% (depending of week and group) of False Positive are consultants that have been recommended and clicked but not bookmarked. The results of this analysis are visible in Table 3.

We notice that the keyword Recommender System has a better F1-score, Accuracy and Precision than the embedding one, the latter has a better recall. However and as talked about in Section 3 Recall is most important because the mecato's goal is to find a mission for a maximum of consultant. The precision may seem very low, but this is because we do not take into account consultants who are not positively bookmarked because the user is not interested in them right now. A user can find a consultant interesting but not bookmark it positively either because he already has someone with a similar profile, or has reached the limit of consultants he can effectively manage at the same time.

Now we look at the number and distribution of bookmark types (Figure 2). We use the number of active users in the group during the corresponding week to normalize the data. An active user is a user who bookmarked at least one consultant. For the first week, we have 12 active users for group A (keyword) and 7 for group b (embedding). And for the second week, we have 8 for group A (embedding) and 7 for group B (keyword). The keyword algorithm results in more ”priority” bookmarks, but other types seem more dependent on the group composition. Group A uses bookmarks ”interesting” and ”why not” more than Group B, regardless of the algorithm. This difference between groups highlights the importance of using a crossover protocol. We do not detect any major difference linked to the algorithm apart from a higher number of positive bookmarks for the keyword algorithm, which is correlated to our measure of Precision and has already been highlighted previously.

6. Conclusion

In this paper we presented a recommendation tool, AI4HR Recruiter that is intended to improve the internal recruitment process of a consulting company. The AI4HR Recruiter relies on content-based recommendation techniques and helps to identify consultants who are relevant to the recruiter's needs. The tool is in use at the consulting company ALTEN in a context of internal recruitment. We evaluated our tool through a number of experiments that indicate improvements in the recruitment process. The recommendations are complemented with explanations that shed light on why a specific consultant has been recommended and greatly reduce the time a user spend on a recommendation. The high recall of the embedding-based recommendation helps to achieve the company's goal to provide project to a maximum of available consultants. Both the users and the managers were satisfied with the improvements of the related business processes.

In the future, we want to ask more questions to our users through the tool because it is the most effective way to obtain answers. We want to improve the precision of our recommender systems which could be achieved by leveraging the ever-growing dataset of explicit feedbacks from the recommendation to train a model instead of relying only on the cosine similarity of embeddings produced by a pre-trained model. Also, we want to propose recommendations based on the previous bookmarks of the user to not rely solely on his manually filled interests.

References

[1] Adomavicius, G., Tuzhilin, A., 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE transactions on knowledge and data engineering 17, 734–749.
[2] Aggarwal, C.C., et al., 2016. Recommender systems. volume 1. Springer.
[3] Freire, M.N., de Castro, L.N., 2021. e-recruitment recommender systems: a systematic review. Knowledge and Information Systems 63, 1–20.
[4] Freire, M.N., Castro, L.N.d., 2020. A framework for e-recruitment recommender systems, in: International Conference on Artificial Intelligence and Soft Computing, Springer. pp. 165–175.
[5] Goldberg, D., Nichols, D., Oki, B.M., Terry, D., 1992. Using collaborative filtering to weave an information tapestry. Communications of the ACM 35, 61–70.
[6] Hong, W., Zheng, S., Wang, H., Shi, J., 2013. A job recommender system based on user clustering. J. Comput. 8, 1960–1967.
[7] Lavi, D., 2021. Learning to match job candidates using multilingual bi-encoder bert, in: Fifteenth ACM Conference on Recommender Systems, pp. 565–566.
[8] Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013. Efficient estimation of word representations in vector space, in: Bengio, Y., LeCun, Y. (Eds.), 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings. URL: http://arxiv.org/abs/1301.3781.
[9] Narducci, F., de Gemmis, M., Lops, P., Semeraro, G., 2018. Improving the user experience with a conversational recommender system, in: International Conference of the Italian Association for Artificial Intelligence, Springer. pp. 528–538.
[10] Pennington, J., Socher, R., Manning, C.D., 2014. Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532–1543.
[11] Pu, P., Chen, L., Hu, R., 2011. A user-centric evaluation framework for recommender systems, in: Proceedings of the fifth ACM conference on Recommender systems, pp. 157–164.
[12] Reimers, N., Gurevych, I., 2019. Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics.
[13] Reimers, N., Gurevych, I., 2020. Making monolingual sentence embeddings multilingual using knowledge distillation, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics.
[14] Resnick, P., Varian, H.R., 1997. Recommender systems. Communications of the ACM 40, 56–58.
[15] de Ruijt, C., Bhulai, S., 2021. Job recommender systems: A review. arXiv preprint arXiv:2111.13576 .
[16] Schmitt, T., Caillou, P., Sebag, M., 2016. Matching jobs and resumes: a deep collaborative filtering task, in: GCAI 2016-2nd Global Conference on Artificial Intelligence.
[17] Tintarev, N., 2007. Explanations of recommendations, in: Proceedings of the 2007 ACM Conference on Recommender Systems, Association for Computing Machinery, New York, NY, USA. p. 203–206. doi:10.1145/1297231.1297275.