Conversational recommendations for job recruiters

François Mentec[1,2], Zoltán Miklós[2], Sébastien Hervieu[1] and Thierry Roger[1]

[1]ALTEN Labs, 35000 Rennes, France
[2]Univ Rennes CNRS IRISA, 35000 Rennes, France

Abstract

Recruitment is a long, complex, and crucial process for every company. Matching candidates to job offers is time-consuming and requires a broad knowledge of the different domains, in particular the relevant skills and qualifications and their relationships. By comparing the required skills with candidates' profiles, recruiters can identify potential collaborators for a given position. To support recruiters' work, we propose recommendation techniques that help the identification of suitable candidates. While job offers contain required skills for a position, the recruiter's preferences are not explicit: for example, some skills might be more important than others etc. These preferences are very hard to elicit, and they even might depend on the particular job offer. We propose conversational recommendation techniques that can support recruiters' work and recommend candidates to a given job offer, based on relevant skills, that can provide an explanation for the recommendation, so that the recruiter has specific information as to why a candidate is recommended. Interaction with the system that can reveal more details about preferences is possible. In this way, a new set of recommendations can be obtaines or he can reinitialize the recommendation procedure, if preferred. Our system is evaluated using a real Resume/Job Offer dataset from a company database. Performance of generated recommendations is compared to a reference Deep-Learning based matching system trained on the same dataset.

Keywords

Conversational recommendation, Recruitment, Explainable recommendation

1. Introduction

Recruitment is a long, complex, and crucial process for every company. Recruiters and human resources experts often need to evaluate a large number of candidates. There are many tools to support the work of recruiters that have been in use for decades [1]. These tools can help with various tasks, including information extraction or matching candidates to job offers.

Our work also focuses on this matching task and we propose a recommender system for recruiters that can support their work through recommendations. We do not aim to replace the human agent in the recruitment process, on the contrary we would like to empower the involved persons. In large companies, the number of potential candidates for a position can be overwhelming. In particular, we focus on reducing the number of candidates a recruiter has to look at using skill-based matching of job offers and candidates. Understanding how various skills listed in job offers or resumes relate to each other is often not self-evident. For example, for a "backend developer" position where the company searches for developers experimented with the "express" development framework, one might also consider candidates with similar competencies or experiences. For example, a candidate who knows well the "hapi" framework (that is a different framework for the same task) could be well suited for the job. Identifying such highly competent candidates requires a broad knowledge of the technical terms. Moreover these technologies evolve quickly. This means that it is not easy for the recruiters to understand the relation between these skills. Clearly, other factors than skills are highly relevant for the recruitment (such as geographic proximity or other factors), and our methods can be extended to deal with these aspects as well.

Our recommender system can help recruiters to identify potential candidates and is capable to suggest candidates beyond a simple keyword match, based on the relatedness of the necessary skills and the candidate's competencies. There are a number of challenges involved in designing and realising such a recommender system. A job offer in general should express all the requirements for the position in question, however in reality it contains only partial information. For example, some skills might be more important than others. In other cases, the recruiters can have specific politics: they might prefer someone with a broad range of skills, while in other cases, one specific skill is crucial for the position, so they seek someone who has this skill. These preference structures are not known in general. They might even be different for each position, or each company. It is difficult to elicit these preferences, as they might not even be clear to the recruiter: he might realize it as he consults the candidates

3rd Edition of Knowledge-aware and Conversational Recommender Systems (KaRS) & 5th Edition of Recommendation in Complex Environments (ComplexRec) Joint Workshop @ RecSys 2021, September 27–1 October 2021, Amsterdam, Netherlands
francois.mentec@alten.fr (F. Mentec); zoltan.miklos@irisa.fr (Z. Miklós)
http://people.irisa.fr/Zoltan.Miklos/ (Z. Miklós)
0000-0002-3701-6263 (Z. Miklós)

resumes. To overcome these challenges, our system follows the conversational recommendation paradigm [2]. We have developed techniques that can propose a candidates based on some initial assumptions. The user can later refine the recommendations through interactions with the system or restart the recommendation process if he prefers so.

In our application context, it is particularly important that recommendations be explainable and configurable: a recruiter should be able to understand why a candidate has been recommended, that is, what is the relation between the candidate skills and the job offer.

Our contributions in this work are the following:

- We propose a recommender system for resumes to job offers, based on a domain-specific ontology.
- We generate explanations for the generated recommendations.
- We propose a conversational recommendation strategy to update recommendations through interactions with the user.
- We evaluate our methods based on a real word dataset and compare to a full deep learning-based recommendation method.

The rest of the paper is organised as follows. We formulate the job candidate recommendation problem in Section 2. We present our conversational recommender system for job recruitment in Section 3. We present the experimental evaluation of our approach in Section 4, including a comparison to a deep learning-based matching system. We discuss related works in Section 5. Section 6 concludes the paper.

2. The job-candidate recommendation problem

Our goal is to recommend candidates for a specific job offer to support recruiters' work. In our recommendation model, each recommendation is accompanied by an explanation and complementary information (see Section 3.2). The user can observe these pieces of information and can decide to change certain parameters of the recommendation process and obtain a new, more refined, set of recommendations. This process can be repeated as many time as necessary, resulting in a conversation between the system and the user that should lead to a few good recommendations.

In the following we describe the data model used to generate job recommendations. Positions are described in "job offers". We recommend job candidates, who are represented by their "resume". In this work we focus on recommendations based on skills only. The model could be extended to take into account other aspects, such as geographic location, education, etc. The Job Offer inform us on the skills required for the position, and the Resume on the skills possessed by the candidate. We assume that Resumes and Job offers are semi-structured, that is we can identify skill labels in these documents. A skill label represent a skill and is composed of a few words. However, different labels can refer to the same skill depending on the vocabulary used by the person which redacted the label, for example, "back-end development" and "server-side development" refer to the same skill, but expressed in a different vocabulary.

A Job Offer in our model is a document containing a set of labels of the necessary skills $o = \{l_1, l_2, \ldots, l_n\}$. A Resume is also a document containing a set of labels, but they represent the skills possessed by the candidate $r = \{l_1, l_2, \ldots, l_n\}$. Those labels can't be compared directly, because they are redacted by different persons, hence use different vocabulary. A function to translate those labels to the appropriate skill in a unified referential is required $f(l) \rightarrow s, s \in S$. For this referential, we propose to use an ontology. An Ontology is a hierarchical structure which takes the form of a Directed Acyclic Graph that represent a set of concepts and their relations. In our work we use a skill ontology, in which nodes correspond to skills and edges denote a "is more specific skill than" relation $O = \{I, r\}$.

3. Conversational recommendation of job candidates

Section 3.1 presents an ontology-based recommendation approach, that can produce interpretable job candidate recommendations, which represents the basis of our conversational recommendation approach. This method generate recommendations to identify potentially relevant candidates. For each recommendation explanations are generated that we describe more in detail in Section3.2. Section 3.3 discusses how the user can interact with the system and further refine the recommendations.

3.1. Ontology-based recommendation

Our recommendations are based only on skills from the candidate's resume and the job offer. In order to deal with the linguistic differences between the various terms that denote skills we rely on a widely adopted ontology of skills: ESCO (There are a number of alternatives which could also be used, including the American ontology O*NET, or company specific proprietary skill catalogs). Our recommendation technique can identify a set of candidates who are likely to suit for a given job offer. The recommendations are generated if a user -who is the

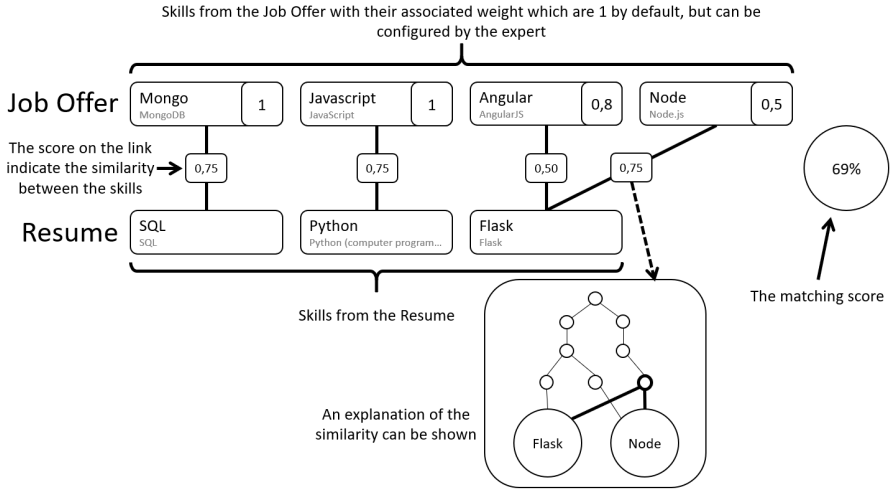

Figure 1: Our model can show why a recommendation was made and how the score was computed. We can see how important each skill from the job offer is (this value can be adjusted by the recruiter), we can see with which skill from the Resume it has been matched, how similar both skills are, and how this similarity has been estimated based on the ontology.

agent in charge of the recruitment- is engaged in searching for a candidate. We obtain recommendations for a given job offer in the following way:

- **Skill matching.** We first match skill labels of the resume and the offer to nodes of the ontology. This is a preparatory step, that we execute only once, we do not need to repeat this step for each recommendation.
- **Skill similarity.** We estimate the similarity of individual skills in the job offer and resume using the ontology.
- **Aggregation.** We aggregate individual skill similarities (between the skills in the offer and resume) depending on the search politic of the user.
- **Generating recommendations.** We generate candidate recommendations for the job offer, together with an explanation.

In the following, we detail these steps.

3.1.1. Skill matching

Job Offers and Resumes are usually written by different people, hence use different vocabularies. To deal with this problem we use an ontology and map each skill in the Resume and in the Job offer to the corresponding skill in the ontology. To realize this task, we use deep learning-based matching techniques.

We trained a classifier on top of a multilingual version of BERT because we can find documents in French and English in our dataset, and the multilingual model showcased stronger performances during experimental evaluation (we input this the larger amount of data used for training). We used every label in French, English, Spanish, German, and Italian to do so. Despite using multiple languages, we made sure that every label would only appear once in the dataset for a given skill to prevent the same label to appear in multiple subsets (i.e. train, dev, or test) because it is the same for multiple languages (e.g. SQL would appear in every language, it is never translated).

Level	0	1	2	3	4
Accuracy	99%	96%	94%	93%	91%

Table 1
ESCO Skill Classifier performances by level of the ontology.

For each label from the ontology, the model was trained to predict the skill it comes from, as well as each of its ancestors up to the top of the ontology. This ensured strong accuracy for skills placed higher in the hierarchy 1.

This approach enables us to compare individual skills (of the Resume and Job offer): we can compare the corresponding terms in the ontology.

3.1.2. Skill similarity

Skills from the offer are requirements and those from the resume can match/fill those requirements. They can be related to each other using a similarity function based on the ontology. One could use different similarity function. We have chosen a similarity function that works particularly well for skills: we would like that the sim-

ilarity function respects the structure of the ontology. We have decided to use the similarity function that was proposed by Panagiotis et al. [3] for comparing skills in a hierarchical structure (Equation 1) :

$$Sim(a,b) = \frac{d_{max} - depth(lca(a,b))}{d_{max}}, \quad (1)$$

where a and b are skills from the ontology, d_{max} is the maximal depth of the ontology, and $lca(a,b)$ their lowest common ancestor.

3.1.3. Aggregating skill similarity scores

We are reducing Resumes and Job Offers to a list of skills which can be compared using their relationships in an ontology. We want a scoring function that measure how much a Resume fits to a Job Offer using only their skills and the ontology, and keep it simple enough that it can be easily explained. The higher the score, the higher the match. Also, the score should be easy to interpret for a human: 1 is a perfect match, and 0 is no match at all. A resume scoring higher than another resume for the same offer is to be interpreted as a better match. Also, $Score(r_1, o_1) > Score(r_2, o_2)$ means the resume r_1 is a better match for job offer o_1 than r_2 is for o_2.

Using this measure we can associate each skill from the offer with its most similar counterpart from the resume. From there we can design multiple possible aggregation politics to match offers and resumes that can correspond to different preferences or recruitment strategies:

1. Average: compute the average of the best association for each skill of the Offer (this is the politic used by default).
2. Min: only consider the association with the lowest similarity.
3. Max: only consider the association with the highest similarity.

Average politic:

$$Score(o,r) = \frac{\sum_{a \in S_o} \max_{b \in S_r} Sim(a,b)}{|S_o|} \quad (2)$$

The Min politic:

$$Score(o,r) = \min_{a \in S_o} \max_{b \in S_r} Sim(a,b) \quad (3)$$

The Max politic:

$$Score(o,r) = \max_{a \in S_o} \max_{b \in S_r} Sim(a,b) \quad (4)$$

o denotes the job offer o, r the resume, S_x the skills set of x, and Sim the similarity between two skills (equation 1).

3.1.4. Generating recommendations

Usually a recommendation consists in presenting a user with an item he could be interested in. In the case of recruitment the item isn't for the user (i.e. the recruiter), but for a job from a customer. So given an offer, we would present the user with a list of candidates that could match the offer. Candidates are recommended based on their score.

The most straightforward way to make recommendations given an offer and a set of resumes, is to rank the resumes based on their matching score with the offer according to our model, and recommend the top-k to the recruiter. On our 6181 offers and resumes, with $k = 10$, this would lead to an average score for the top-k of 0.98 which ensure high precision, but only 2204 resume would ever be recommended meaning that almost two third of our consultants would never be attributed a job.

An other approach would be to probabilistically recommend a resume based on its score. The probability to recommend a resume i is its score divided by the sum of every score: $P(i) = \frac{S_i}{\sum_{j \in S} S_j}$. This method reduce the average score of the recommendation to 0.74 but increase the catalog coverage to almost 100% (6124 resumes are now recommended). We could also use the exponential of the score instead of the score itself; this leads to an average score of 0.71 and 6179 resumes being recommended.

3.2. Explanation

By using a manually designed scoring function and a human-understandable representation of skills (skill ontology), we can generate an explanation for each recommendation. An example of such explanation can be seen in Figure 1. The resume match the job offer at 69% because its skills are close to those contained in the job offer: SQL is close to Mongo, Python to Javascript, Flask to Node, and Flask is also not too far away from Angular. If the user wants to know why a skill is matched to another one with a certain score, he can click on the link between them to see where are the corresponding nodes located in the ontology and how they are linked.

An explanation can have different forms, graphical or textual, and pursue different aims [4]. Law et al. [5] compared a graphical explanation to a textual one, and if the first was preferred by users, the last one yielded the best accuracy in terms of decision making. In our case and given the use of an ontology, we can generate an explanation by making sentences of the form "s_1 is adj to s_2 because both are $lca(s_1, s_2)$". If skills s_1 and s_2 would be matched together, adj would be an adjective selected based on the similarity between the skills, and $lca(s_1, s_2)$ their lowest common ancestor. For example, the link between Mongo and SQL from Figure 1

give: **SQL** is **close** to **Mongo** because both are **database management systems**. [1]

3.3. Conversational recommendation

The explanation enables also to refine the search and guide the user to identify her (or her company's) preferences. In this way, the user can fine-tune the query and obtain a new set of recommendations. To allow this fine-tuning, we propose a slight variation from our initial scoring function where a weight is associated to each skill from the Offer. W is a weighting function that associates to each skills of the job offer a weight between 0 and 1. Those weights can be seen in the Figure 1.

$$Score(o, r, W) = \frac{\sum_{a \in S_o} max_{b \in S_r} Sim(a, b) \times W(a)}{\sum_{a \in S_o} W(a)} \tag{5}$$

If the recruiter notice a lot of resumes are recommended because they contain a skill he doesn't find that important, he can lower the weight attributed to the skill, or even remove it. On the opposite, if he notice a very important skill is always missing from recommended resumes, he can increase its weight.

For example, if a recruiter is looking for an "Angular" developer and gets a lot of recommendations that do not contain the "Angular" skill, but javascript and other web frameworks (e.g. Vue) he can increase the weight of this "Angular" skill to get more recommendations containing it.

4. Experimental evaluation

4.1. Dataset

To evaluate our model, we have access to a database of 343.802 resumes from the ALTEN company. It uses a specific format called "Technical Files", that is a structured resume which contains a list of professional experiences that include a job title, a description and a set of skills. We have a total of 249,721 experiences, less than the number of resumes. This means that some resumes are incomplete and do not contain any experiences. After preprocessing we keep only resumes that contain at least 2 experiences, so we're left with 30,907 resumes and 154,033 experiences. Most of these resumes are in French, but we also have some in English. We generated a resume/job matching dataset by creating pairs $(Resume, JobOffer)$, and we considered the last experience of a resume as a job offer.

[1] Our example is hypothetical, as in the ESCO ontology, that we use, there is currently not any mention of Mongo, Angular, Flask or Node. Though we can find SQL, JavaScript and Python along other IT technologies like CMS (e.g. Wordpress). ESCO is being extended, and also one can use a company specific enriched version that would contain all relevant skills and competences.

4.2. Evaluation methodology

We evaluate our system as a Job Recommender System and as a Conversational Recommender System through the following aspects:

- Raw accuracy compared to Deep Learning on a binary classification task.
- Weighting techniques, to understand the impact the user can have.
- Utility/Efficiency of its conversational components.

4.3. Evaluation as binary classification

We evaluate the performance of our techniques on the following task: given a resume and a job, the system must predict if they constitute a match or not. For this evaluation we can rely on a variety of classification metrics: accuracy, precision, recall, and F1-Score. A high precision would greatly improve the efficiency of the recruiters (less irrelevant resumes to look at), and a high recall would reduce the risk of missing a good resume. Also too low a recall would raise ethical concerns, some resume might never be recommended, hence preventing their owners from getting a job.

We're going to compare our ontology-based recommender system to a system entirely based on Deep Learning.

4.3.1. Dataset for binary classification

We constructed a new evaluation dataset from our dataset introduced in section 4.1. The original includes a number of resumes matched with an appropriate job. For this evaluation we also need pairs of resumes with jobs that do not match. We duplicated every pairs and replaced the job with an experience randomly selected from another resume. In this way, we obtained an equal amount of good and bad pairs of resume and job. This procedure is depicted in Figure 2.

This methodology has a few shortcomings:

- If an Experience is part of a Resume, this does not mean the person was a good fit to the job (staffing errors can occur).
- Taking an Experience at random for bad pair may sometimes land a good match, especially since all jobs are from the same company, and as such in in similar fields. However, the fact that ALTEN employ a wide variety of profiles can mitigate this effect.

We split our dataset into 3 subsets:

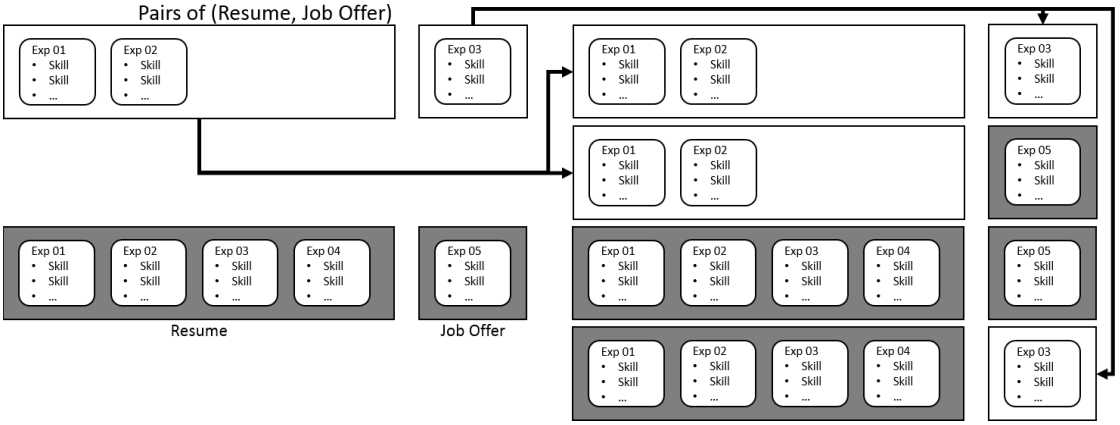

Figure 2: From pairs $(Resume, JobOffer)$ to a Matching Dataset. Every pair is duplicated and the Job Offer is replaced by another Job Offer randomly picked from another pair, thus generating bad pairs.

- a Train set composed of 49,452 pairs, on which we trained a Deep Learning baseline and the Linear Regression on top of the score from the Ontology-based model.
- a Dev set composed of 6,181 pairs, used to decide when we must stop training the Deep Learning model and to fine-tune the Ontology-based model.
- a Test set composed of 6,181 pairs, to make a final evaluation of the models and compare them between each other.

4.3.2. Deep Learning Recommender System

We use as a baseline a full deep learning recommender system that we trained using BERT [6]. We use the Transfomers library from Hugging Face [7] for implementation. The pairs were fed to the model using the following structure : <cls>Resume<sep>job Offer<sep>. <cls> is the classification token, it indicates the position at which the model makes its prediction (i.e. does the resume and the job offer match). The resume and job offer in the input are the list of skills they contain separated by a comma followed by a space (", "). <sep> is the separator token, it indicates the end of a document (resume/job offer).

We used the BertForSequenceClassification model from the Transformers library which use a cross entropy loss when the number of labels is 1. It was trained with a batch size of 16 until the accuracy on the dev set stopped showing any improvement for 3 consecutive epochs. The best model (3 epochs prior) was kept. We used the optimization function AdamW [8] with an initial learning rate of 5e-5.

Metric	Full Deep Learning	Ontology-based
Accuracy	83.03%	62.76%
Precision	86.96%	61.29%
Recall	77.92%	70.32%
F1-Score	82.20%	65.50%

Table 2
Recommender Systems performances. This table show the accuracy, precision, recall and F1-Score on our binary classification dataset for a full Deep Learning model based on BERT and our ontology-based recommender system.

4.3.3. Results for binary classification

We now have 2 Job Recommender Systems which can be compared on the same dataset. The results in table 2 are in favor of Deep Learning with a strong accuracy of 83.03% higher than the Ontology-based one, which is only 62.76%. This result is not surprising as the ontology based similarity distances cannot compete with deep learning based ones. However, as we show below, one can obtain much better figures through conversational interactions.

4.4. Weighting evaluation

We test the impact of the weighting paradigm introduced in section 3.3 on the accuracy by randomly generating a thousand weighting for each pair, and measuring the portion of those weightings that led to a good prediction. In the case in which the expert always finds the perfect configuration regarding the importance of each skills in the job offer, the accuracy may rise up to 87%. Obviously it is unlikely even an expert would always find the one in a thousand best weighting, but we could easily expect

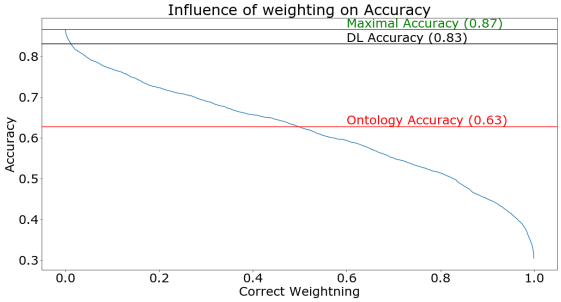

Figure 3: The influence of weighting on the accuracy of the ontology-based RS. We can see that 63% of pairs have about 50% of their weighting that led to a correct prediction. And 87% of the pairs have at least one weighting that led to the right prediction, which is better than the Accuracy of the DL RS.

him to do better than average. Empirical testing would be required to evaluate how and with which accuracy a recruiter would find a good weighting.

4.5. Evaluation of conversational interactions

We take a subset of 100 pairs of offers with the right resume from the test set, and for each offer we recommend n resumes, the set of recommended resumes is denoted by R. If the correct resume is recommended it's a success, otherwise we take a step to fine-tune the recommendation until it is recommended or we reached the maximum number of steps (100).

In the absence of real users, We adopted heuristic techniques to try to mimic human reasoning. The actions that can be taken are the following:

- Add a skill: we consider the recruiter has an idea of the kind of profile he wants, and that profile is the right resume for the offer. So if a skill from the correct resume is infrequent among recommended resume, it can be added to the offer.
- Increase the weight of a skill: if a skill from the offer is in the correct resume but is infrequent among recommended resumes, its weight can be increased.
- Decrease the weight of a skill: if a skill from the offer is not in the correct resume but is frequent among recommended resumes, its weight can be decreased. If its weight reaches 0, this action is equivalent to removing the skill from the offer.

Actions are determined and weighted in the following way:

- Add skill: for each skill in the correct resume but not in the offer, the probability of adding it to the offer is the inverse of its frequency among recommended resumes (equation 6).
- Increase the weight of a skill: for each skill in the correct resume and in the offer, the probability to increase its weight, is also the inverse of its frequency among recommended resumes (equation 6).
- Decrease the weight of a skill: for each skill in the offer but not in the correct resume, the probability to decrease its weight is its frequency among recommended resumes (equation 7).

$$P_{add/increase}(s) = \frac{|R|}{|r \in R \wedge s \in r| + 1} \tag{6}$$

$$P_{decrease}(s) = \frac{|r \in R \wedge s \in r|}{|R|} \tag{7}$$

At each step/turn, we only apply the action with the highest probability.

We also evaluated our conversational recommendation w.r.t. the following metrics: success rate, low rank rate, average number of turns, see Sun et al. [9]. For a list of 100 offers and 100 resumes, and $n = 5$ recommendations per step/turn, we achieve a 74% success rate, 8.69 average turns/steps, and 26% low rank rate.

5. Related work

5.1. Recommender systems for recruitment

There are a number of systems that try to support the recruitment process through recommendations. Surveys of these methods include [10], [11]. A commonly used approach is to try to match job candidates to job offers, based on different available sources of information. For example, Lin et al. [12] propose a machine learned resume-job matching method. Their method is composed of three phases: First, they use unsupervised feature extraction methods to identify relevant features from job offers and resumes. Then they train different classifiers to match these features. Finally they rely on ensemble learning techniques to combine these classifiers. Another technique based on machine learning methods was proposed by Malinowski et al. [13]. Papparizos et al. [14] proposes a job recommendation technique that is based on the job transition graphs, which is constructed using the candidates resumes. A conceptually similar approach was proposed by Heap et al. [15] and [16] who exploit the job history of the candidates to offer better recommendations. Other methods of job candidate matching include [14]. Yang et al. [17] developed a job recommendation technique that combines content-based filtering techniques with collaborative filtering.

More recent proposals rely on bilateral preference matching methods [13], [18] or on deep learning [19], [20], [21]. A major shortcoming of machine learning and deep learning based techniques in this context is the lack of interpretability. This poses even an ethical problem in this context. Even if recruitment decisions are finally made by humans, recommendations that influence the process are not interpretable and they are not transparent. The usage of ontologies inside Recommender Systems isn't new, they can be used to overcome the cold-start problem and improve accuracy, by displaying the profile of a user in a human-readable format he can easily edit to fine-tune his recommendations [22].

The article [23] surveys explainable recommendation techniques [23]. One can distinguish model-intrinsic methods and model-agnostic methods. For model-intrinsic techniques the decision mechanism is transparent, one can derive explanations directly for the model decisions, while for model-agnostic (or post-hoc) explanations the recommendation decision is black box, and explanations try to give some insights event if they are not directly derived from the model. Our method is model-intrinsic, we obtain explanations directly from our recommendation model. This choice has a price: one could potentially obtain more accurate recommendations with the help of deep learning based methods, but then we could not have transparent model-intrinsic explanations. An interpretable job-person recommendation, with post-hoc explanations was presented in [24].

Recommender systems (for movies, products, services, etc.) try to exploit and reconstruct the user's preferences, based on some signals or patterns in the data. Yan et al. [25] try to reconstruct the preference structures for person-job recommendation from the interview history of candidates. Such information is rarely available and in general, the preferences of the recruiter are not present in the data. For example, the required skills in job descriptions correspond to some ideal candidate, but finally the companies hire candidates, who do not have all of the required skills in the offer. The preference structure is not present in the data. For example, for some jobs the recruiter might prefer someone who has a specific skill that is of high importance for the position, while in other cases, some closely relevant skills could be enough and a candidate with a diverse set of skills is better. Moreover, these preferences might be tacit to the recruiter, he might not be aware his own preferences prior to the selection process or the preference might depend on some other factors, such as situations where the company needs someone urgently for a project, in which case, they might opt for a less qualified candidate. To the best of our knowledge there is no job recommender system that could deal with this problem of tacit preference structures. As preference structures are not available to the recommendation process we argue that a conversational recommendation technique is a suitable approach.

5.2. Conversational recommendation

Conversational recommender systems (CRS) [2] rely on interactions with the users to compute recommendations. This exchange with the user can take the form of a conversation in natural language, but isn't limited to this modality. For example, Yo et al. [26] use visual dialogues in the conversation. CRSs often realize the following the following phases: Request, Recommend, Explain, and Respond. In our work we adopt the conversational recommendation process. In our context, the Request corresponds to the situation where the recruiter starts the search for candidates and asks for potential resumes. He can obtain recommendations, together with complementary information that serves as explanation as to why a given candidate was recommended. Based on these pieces of information the user can respond, refine or restart the search for candidates. The excellent survey paper [2] highlights a number of application areas of CRS, but to the best of our knowledge they have not yet been applied to the recruitment domain.

6. Discussion and Conclusion

We have proposed a job candidate recommendation technique based on conversational recommendation principles. Our system helps identifying suitable candidates, based on the skills in the resumes and their relation to the required skills in the job offer. Our recommendation is completely transparent and for each recommendation, it is clear, why a given candidate is recommended (or not recommended). The explanations serve also as a basis of a conversation with the user, who can then refine the recommendation process. We evaluated our recommendation technique through extensive experimentation.

Currently the deep-learning baseline showcase much stronger performances. There are 3 ways our system could catch up: improve the mapping between skill labels from documents and skills from the ontology, refine the ontology by adding new skills and relationships, use more complex similarities metrics.

We plan to introduce more sophisticated conversational techniques to better guide the user, by exploiting the different statistics of skill distributions in the data. Also we would like to investigate more realistic preference structures also including multi-objective variants. We also plan to evaluate our system in use, w.r.t. perception of the recommendation process and interpretability of recommendations and the guiding suggestions with real users.

References

[1] M. Tixier, Employers' recruitment tools across europe, Employee relations (1996).

[2] D. Jannach, A. Manzoor, W. Cai, L. Chen, A survey on conversational recommender systems, CoRR abs/2004.00646 (2020). URL: https://arxiv.org/abs/2004.00646. arXiv:2004.00646.

[3] P. Mavridis, D. Gross-Amblard, Z. Miklós, Using hierarchical skills for optimized task assignment in knowledge-intensive crowdsourcing, in: J. Bourdeau, J. Hendler, R. Nkambou, I. Horrocks, B. Y. Zhao (Eds.), Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016, ACM, 2016, pp. 843–853. URL: https://doi.org/10.1145/2872427.2883070. doi:10.1145/2872427.2883070.

[4] N. Tintarev, Explanations of recommendations, in: Proceedings of the 2007 ACM conference on Recommender systems, 2007, pp. 203–206.

[5] A. S. Law, Y. Freer, J. Hunter, R. H. Logie, N. McIntosh, J. Quinn, A comparison of graphical and textual presentations of time series data to support medical decision making in the neonatal intensive care unit, Journal of clinical monitoring and computing 19 (2005) 183–194.

[6] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[7] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, A. M. Rush, Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Online, 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[8] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, arXiv preprint arXiv:1711.05101 (2017).

[9] Y. Sun, Y. Zhang, Conversational recommender system, in: The 41st international acm sigir conference on research & development in information retrieval, 2018, pp. 235–244.

[10] S. T. Al-Otaibi, M. Ykhlef, A survey of job recommender systems, International Journal of Physical Sciences 7 (2012) 5127–5142.

[11] Z. Siting, H. Wenxing, Z. Ning, Y. Fan, Job recommender systems: A survey, in: 2012 7th International Conference on Computer Science Education (ICCSE), 2012, pp. 920–924. doi:10.1109/ICCSE.2012.6295216.

[12] Y. Lin, H. Lei, P. C. Addo, X. Li, Machine learned resume-job matching solution, arXiv preprint arXiv:1607.07657 (2016).

[13] J. Malinowski, T. Keim, O. Wendt, T. Weitzel, Matching people and jobs: A bilateral recommendation approach, in: Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06), volume 6, 2006, pp. 137c–137c. doi:10.1109/HICSS.2006.266.

[14] I. Paparrizos, B. B. Cambazoglu, A. Gionis, Machine learned job recommendation, in: Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys '11, Association for Computing Machinery, New York, NY, USA, 2011, p. 325–328. URL: https://doi.org/10.1145/2043932.2043994. doi:10.1145/2043932.2043994.

[15] B. Heap, A. Krzywicki, W. Wobcke, M. Bain, P. Compton, Combining career progression and profile matching in a job recommender system, in: D. N. Pham, S. Park (Eds.), PRICAI 2014: Trends in Artificial Intelligence - 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia, December 1-5, 2014. Proceedings, volume 8862 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 396–408. URL: https://doi.org/10.1007/978-3-319-13560-1_32. doi:10.1007/978-3-319-13560-1_32.

[16] B. Heap, A. Krzywicki, W. Wobcke, M. Bain, P. Compton, Combining career progression and profile matching in a job recommender system, in: Pacific Rim International Conference on Artificial Intelligence, Springer, 2014, pp. 396–408.

[17] S. Yang, M. Korayem, K. AlJadda, T. Grainger, S. Natarajan, Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive statistical relational learning approach, Knowledge-Based Systems 136 (2017) 37–45. URL: https://www.sciencedirect.com/science/article/pii/S095070511730374X. doi:https://doi.org/10.1016/j.knosys.2017.08.017.

[18] W. Hong, L. Li, T. Li, W. Pan, Ihr: An online recruiting system for xiamen talent service center, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '13, Association for Computing Machinery, New York, NY, USA, 2013, p. 1177–1185. URL: https://doi.org/10.1145/2487575.2488199. doi:10.1145/2487575.2488199.

[19] Y. Luo, H. Zhang, Y. Wen, X. Zhang, Resumegan: An optimized deep representation learning framework for talent-job fit via adversarial learning, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM '19, Association for Computing Machinery,

New York, NY, USA, 2019, p. 1101–1110. URL: https://doi.org/10.1145/3357384.3357899. doi:10.1145/3357384.3357899.

[20] C. Qin, H. Zhu, T. Xu, C. Zhu, L. Jiang, E. Chen, H. Xiong, Enhancing person-job fit for talent recruitment: An ability-aware neural network approach, in: The 41st International ACM SIGIR Conference on Research amp; Development in Information Retrieval, SIGIR '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 25–34. URL: https://doi.org/10.1145/3209978.3210025. doi:10.1145/3209978.3210025.

[21] C. Zhu, H. Zhu, H. Xiong, C. Ma, F. Xie, P. Ding, P. Li, Person-job fit: Adapting the right talent for the right job with joint representation learning, ACM Trans. Manage. Inf. Syst. 9 (2018). URL: https://doi.org/10.1145/3234465. doi:10.1145/3234465.

[22] S. E. Middleton, D. De Roure, N. R. Shadbolt, Ontology-based recommender systems, in: Handbook on ontologies, Springer, 2004, pp. 477–498.

[23] Y. Zhang, X. Chen, Explainable recommendation: A survey and new perspectives, Found. Trends Inf. Retr. 14 (2020) 1–101. URL: https://doi.org/10.1561/1500000066. doi:10.1561/1500000066.

[24] R. Le, W. Hu, Y. Song, T. Zhang, D. Zhao, R. Yan, Towards effective and interpretable person-job fitting, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 1883–1892. URL: https://doi.org/10.1145/3357384.3357949. doi:10.1145/3357384.3357949.

[25] R. Yan, R. Le, Y. Song, T. Zhang, X. Zhang, D. Zhao, Interview choice reveals your preference on the market: To improve job-resume matching through profiling memories, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining, KDD '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 914–922. URL: https://doi.org/10.1145/3292500.3330963. doi:10.1145/3292500.3330963.

[26] T. Yu, Y. Shen, H. Jin, A visual dialog augmented interactive recommender system, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 157–165.