Improving Candidate to Job Matching with Machine Learning

Agnes van Belle, Eike Dehling, and Daniel Foster

Textkernel, Amsterdam, the Netherlands
http://www.textkernel.com

Abstract. Recruiting candidates to fit a certain job is a task crucial to many companies in the field of human resources. This usually starts with a labor-intensive process of manually searching through the available applicants in a certain collection, reviewing their CVs, and eventually producing a shortlist of suitable candidates to be interviewed.
We researched how to use user feedback (annotations) to improve a software product for matching vacancies to candidates. We learned a ranking algorithm to execute on top of a standard search engine to improve candidate shortlist generation given a vacancy.
Here we describe the user instruction procedure we conducted, the evaluation procedure we used, and the challenges we faced when applying techniques from academia in a production system, along with their solutions. In particular, we dealt with ambiguous and incomplete assessment data by combining unassessed and assessed documents. Our assumption is unassessed documents carry some implicit feedback.
In the end we achieved around 20 percent improvement in relevant results on our evaluation data. A qualitative analysis showed us that because of the relatively small final filtered dataset this did not generalize well enough to be used in the production system.
We conclude with some further research and recommendations for future projects in the same area.

Keywords: Learning to Rank · Search Technology · Staffing Technology

1 Background

Textkernel is a company that develops software products for the HR and recruitment market - for example software for parsing CVs and vacancies, software for intelligent searching in large databases of candidates and vacancies, and software for matching vacancies and candidates. Our team is focused on matching vacancies with candidates in the best way possible. Here we are concerned with finding suitable candidates for a given vacancy.

In our matching system, a query (a set of matching criteria) can be either manually entered or automatically generated by uploading a vacancy. This query is executed against a standard search engine [1] containing all available candidates, returning a list of all matching candidates ordered by relevance. This

relevancy score is calculated using a basic retrieval model that uses keyword frequency statistics (a custom scoring formula similar to e.g. BM25 [15]).

In this project we explored how to use machine learning techniques to make an improved personalized ordering of the top results, such that this ranking is not only based on simple keyword frequency, but on more complicated preferences (e.g. balancing hard skills, years of experience, travel distance, etc.). We do this by learning a model that re-ranks the top N results returned by the search engine (common is reranking for example only the top 100, as models can be complex, to limit impact on performance).

To get the users' preferences, we put a feedback mechanism in the user interface through which users mark candidates as relevant or irrelevant. See Figure 1. Each annotation is stored together with the query and document in question.

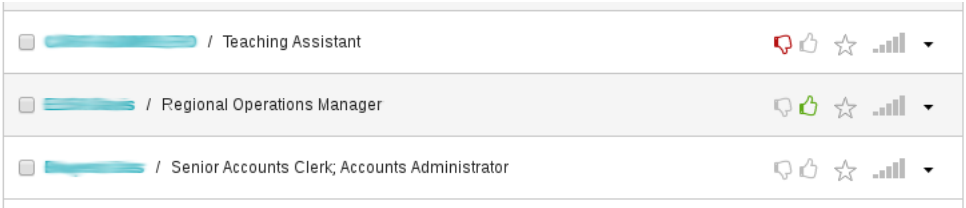

Fig. 1. Part of the assessment user interface as it appear to the users. Three documents are shown, the first two are annotated. The query is not shown.

The customer involved in this project is a Dutch-based recruitment and human resources company that is in the worldwide top 10 of global staffing firms in terms of revenue. They have a few hundred thousand candidates in the Netherlands alone which they match with incoming requests for personnel. Their staffing consultants use our system to find suitable candidates to fulfill these requests.

2 Related work

The concept of ranking results using a machine-learned model is called Learning to Rank [13] (henceforth LTR). Regarding search systems usually a two-phase process is applied: first an initial order is determined using simpler retrieval models which permit fast query evaluation (e.g. TF-IDF [14] or BM25 [15]). In the second phase, the more computationally expensive LTR model is used to re-rank for example the top 100 of these documents.

Conceptually, LTR-specific approaches learn a relative ordering by optimizing the ordering of all possible pairs in a list, or by directly optimizing the list order itself using a ranking quality metric defined for lists [4]. For a more complete reference of modern LTR approaches, we refer to [17]. We will here briefly mention ranking approaches applied to the field of HR.

There are some notable studies in this field using state-of-the-art LTR algorithms. In [2], LTR is used in a job search system of a large secondment company, describing the technical system architecture and evaluating different types of LTR algorithms. The approaches are evaluated by using historic placement data of the company over 49 jobs. In [7], LTR is employed to increase the precision of candidate search given a vacancy. The queries from vacancies are first expanded with more information regarding the possible previous jobs, which increases the recall of correct candidates. LTR is then employed to re-rank the top results to improve their precision. The approach is evaluated using human-made assessments.

Some other approaches calculate the candidate's suitability without considering the relative ordering of the initial result list. In [6], regression algorithms are applied in a job market context to give candidates a score given a job, combining matching signals from hard skills extracted from a CV with signals from soft skills found using social media. In [12] it is investigated how to estimate the hiring probability of freelancers in an online labor market using a probabilistic model with as features criteria of both the employer and the freelancer. They train and evaluate their approaches using historical hiring decision data.

Finally, [10] and [11] contain useful insights into the limitations and challenges related to collection of human-made assessments for evaluating search systems. In [9] the authors discuss practical challenges encountered obtaining implicit feedback signals to train and evaluate learning to rank models in an e-commerce setting.

3 Methodology

In the first phase of this study, users annotated data for how well candidates matched vacancies. Based on a vacancy a user writes a query and then sees the results. We added two widgets to the results in the user interface, a green thumbs up and a red thumbs down, so that users could indicate the matching quality of that result. See Figure 1. We also conducted meetings with the users to instruct them on the project goals and guidelines about annotating.

The guidelines given to users were to only assess vacancies where they noticed at least one relevant candidate and one irrelevant candidate in the results and else skip assessing that vacancy. This is because an LTR algorithm needs this contrast to learn. Furthermore we instructed users to annotate queries with a complete information need, to annotate approximately the first page of results, and to annotate one or two vacancies per week during daily work.

As we monitored user annotation activity over the next three months, we added extra instructional sessions because not all users were following the annotation guidelines. For example, some users annotated only some particularly relevant candidates or particularly irrelevant ones, or annotated before all criteria of a vacancy were entered.

The intent was to wait a few months until enough queries (a few hundred) were assessed, and then to train an LTR model directly on these assessed queries.

4 Approach

After six months we concluded the annotation efforts and started analyzing the data. We first preprocessed the data, filtering out queries with an incomplete information need. This is outlined in subsection 4.1. Then we discovered that not all users had assessed according to our guidelines, many queries had been assessed incompletely. Due to this we investigated how to best combine the assessed and un-assessed documents. This is detailed in subsection 4.2. Finally, we outline the LTR models and features we used in subsection 4.3.

4.1 Query field filtering

We first got all queries and their associated assessed candidates from the database. Note that our queries contain several fields, e.g. "location", "job title", "job class", "date of application" etc. The filtering of queries was an iterative process based on how and which fields were populated, as indicated below.

- **Relative date ranges** First, we had to remove relative dates from queries, because we had to re-run the queries for feature extraction and a query field like "applied previous week" selects different candidates when being run months after the original user query. Note that we kept the altered query in the dataset.
- **Location field criterium** Second, we also removed queries from the dataset with only a location part, such as "Amsterdam (50 miles radius)", since only location is an incomplete information need.
- **Multiple field criterium** A brief look at the dataset indicated that users often entered too few criteria in the query for it to be a believably accurate representation of an actual vacancy. For example, users would search for only the job title. Then they would assess candidates based on other criteria in the vacancy that were absent from the query. More about that in section 6. Hence, we decided to keep only queries with multiple fields.
- **Multiple assessments criterium** We also noticed some users only assessed part of the results, for example only assessing relevant documents, skipping the other type. For a LTR algorithm, both types of assessments are needed per query. We removed queries where this was not the case.

Table 1 shows less than a quarter of our data matched all criteria. Therefore we explored augmenting it with implicit signals in unassessed documents.

4.2 Augmenting assessments data

We expected that users would assess the top ten or twenty results for each query and would assess several documents as relevant and several as irrelevant per query. However as described in section 4.1 many queries were assessed incompletely, often only a few particularly (ir-)relevant candidates had been assessed.

Table 1. Filtering out queries with too few assessments or incomplete information

Selection of queries	**# queries**	**# assessments**
All queries	229	1514
Matching location field criterium	227	1512
Matching multiple field criterium	169	1092
Matching multiple assessments criterium	59	1235
Matching multiple assessments, multiple fields criterium	38	864

We had the hypothesis that there is implicit feedback in unassessed documents, we assume there was a reason why users assessed part of the documents and skipped some documents per query. For example if users only assessed some documents as irrelevant, you might assume the rest is at least slightly relevant. To test this hypothesis we augmented the training data by assigning a relevance to the unassessed documents by a heuristic. The heuristics for assigning a relevance are inspired by the common user behavior assumption that users start browsing a list of search results at the top, skipping irrelevant ones and continue until they have seen enough relevant results – as described for example in [5, 16]. In Table 2 are results for an LTR model with several heuristics for unassessed document relevance and for selecting query sets. The evaluation metric we used is NDCG [14], a common ranking quality metric.

Table 2. Effect of unassessed documents approach on reranked NDCG score

Query Set	**Heuristic for Unassessed**	**NDCG Change**
Queries with >= 1 assessed documents	Marked irrelevant	1 %
Queries with >= 1 positive and >= 1 negative assessment	Marked irrelevant	4 %
Queries with >= 3 assessments, and >= 1 positive and >= 1 negative assessment	Marked irrelevant.	6 %
Queries with >= 3 assessments, and >= 1 positive and >= 1 negative assessment	Above the last user assessment: marked irrelevant, below: slightly irrelevant	6 %
Queries with >= 3 assessments, and >= 1 positive and >= 1 negative assessment	Above the last user assessment: marked irrelevant, below: dropped	6 %

Based on these results we decided to use queries with at least three assessments, one positive and one negative assessment, and we marked un-assessed documents as being irrelevant. By augmenting the dataset this way, we ended up with **71** queries, and **5999** (non-unique) accompanying documents in total.

4.3 Reranker models and features

We applied two different LTR algorithms to our data:

- **LambdaMART** LambdaMART [3] is a state-of-the art LTR algorithm showing impressive results [17]. This is a gradient boosting approach that directly optimizes document order.
- **Linear regression** A simple least-squares linear regression algorithm was used as a baseline comparison approach. This approach directly produces a score per document by which you then sort the results.

As features we used properties of the vacancy, properties of the candidate and features describing the matching, below is a partial list:

- **Vacancy features** are properties of the vacancy, such as desired years of experience or job class.
- **Candidate features** are among others years of experience, number of jobs, skills categories, job classes.
- As **matching features** we used for example search engine score for jobtitle match, score for skills match, or travel distance for the candidate to the job.

Learning and evaluation was done using the common ranking quality metrics NDCG and Precision [14], over the top 10 results. We also conducted parameter tuning for LambdaMART, as described in section 5.

5 Results

We evaluated both models mentioned in section 4.3 on our dataset using cross validation, where we shuffled data and made 5 different train/test splits using 80% and 20% of the data, respectively. We then trained and evaluated 5 times and averaged the scores.

We tried several parameter conbinations for LambdaMART, what follows is a listing with the parameter value from the final best combination in bold: number trees (5, 10, 25, 50, 100, **200**), number leafs (3,**5**,7,9), minimum leaf size fraction (**0.0001**, 0.0002, 0.001, 0.01, 0.05), sampling fraction rate per tree (0.25, 0.5, 0.75, **1.0**) and feature sampling fraction rate per split (0.25, **0.5**, 0.75).

Table 3. Results from the best linear and the best LambdaMART reranker

Learn to Rank Model	**NDCG@10**		**Precision@10**	
	Baseline	Reranker	Baseline	Reranker
Linear reranker	0.35	0.41 (+18%)	0.18	0.20 (+7%)
LambdaMART reranker	0.33	0.47 (+42%)	0.23	0.32 (+39%)

As indicated in Table 3 the LambdaMART clearly outperformed the linear model and had a good relative improvement over the baseline.

6 Discussion

A qualitative analysis was done on several re-ranked query results to see the patterns the algorithm had learned and if they made intuitive sense.

Shown in Table 4 are the original and reranked top 10 results on a query only consisting of the fulltext keyword "*burgerzaken*" ("civil affairs"), numbered by their original ranking position. Relevant results are marked in bold and unassessed results are marked "N/A". There were no documents assessed as irrelevant. The reranker improves upon the original results by returning one additional relevant candidate in the top 10 and only relevant candidates in the top eight. When analyzing the user annotations for this query, it became clear that relevance of the results was at least partially based on location, even though this is not part of the query. All of the candidates in the results have similar past experience as civil affairs clerks, but the majority of candidates marked as relevant are located in or near Rotterdam (locations marked bold). To a degree, the reranker has learns to favor candidates located in the city.

Table 4. Reranked results for a query for *burgerzaken* (English: *civil affairs clerk*)

Original			**Reranked**		
Original ranking	Location	Relevant	Original ranking	Location	Relevant
1	**Rotterdam**	**Yes**	1	**Rotterdam**	**Yes**
2	**Rotterdam**	**Yes**	18	**Rotterdam**	**Yes**
3	**Gorinchem**	**Yes**	2	**Rotterdam**	**Yes**
4	Deventer	N/A	7	Klarenbeek	**Yes**
5	**Papendrecht**	**Yes**	6	**Rotterdam**	**Yes**
6	**Rotterdam**	**Yes**	17	**Rotterdam**	**Yes**
7	Klarenbeek	**Yes**	14	's-Hertogenbosch	**Yes**
8	Sneek	N/A	3	**Gorinchem**	**Yes**
9	Amsterdam	N/A	8	Sneek	N/A
10	Mijdrecht	**Yes**	13	's-Hertogenbosch	N/A
Precision@10 =0.7			Precision@10=0.8		
NDCG@10=.77			NDCG@10=0.87		

In Table 5, similarly, are the original and reranked top 10 results for a query for the fulltext keywords "*planner sita*". In contrast to the query seen in Table 4, this query is specifically semantic, seeking a transportation planner with experience working for the company *SITA*. Results that fit this profile are labeled with a bold *both*. The relevant results all fit this profile, while other results are mostly a partial match. In this example, the reranker intuitively moves three relevant results to the top of the ranking. Interestingly, two of the unassessed results which stayed in the top 10 after reranking have experience that seems to be a match. The candidate originally ranked second has recent experience that is unrelated, but past experience that is a match and is reranked at the fifth position.

In both of these examples, we see that the reranker is responding to user preference in an expected way. The reranker itself works: it learns and responds

Table 5. Reranked results for a query for the fulltext keywords *planner sita*

Original			**Reranked**		
Original ranking	SITA/planner experience	Relevant	Original ranking	SITA/planner experience	Relevant
1	**both**	**Yes**	3	**both**	**Yes**
2	**both**	N/A	1	**both**	**Yes**
3	**both**	**Yes**	15	**both**	**Yes**
4	neither	N/A	4	neither	N/A
5	only planner	N/A	2	**both**	N/A
6	**both**	N/A	6	**both**	N/A
7	**both**	No	8	only SITA	No
8	only SITA	No	7	**both**	No
9	**both**	**Yes**	5	only planner	N/A
10	**both**	**Yes**	9	**both**	**Yes**
Precision@10 =0.4			Precision@10=0.4		
NDCG@10=.71			NDCG@10=0.82		

to a location preference (not specified in the query) in Table 4. And it better favors candidates with the most relevant experience match in Table 5. Due to the fact that there was no location element in the query itself for the query in Table 4 it is however debatable what kind of patterns the reranker actually has learned. It is likely the reranker has based its reranking on some properties of the data that just happened to correlate with location for this query.

We had a detailed qualitative look at five queries, and three of them showed this kind of unexplainable behavior. When having a less detailed look at a few dozen queries not in the training data, we discovered more such unexpected behavior.

7 Future work

Some important lessons we learned conducting an LTR project with human-annotated data were as follows. First, users appear to have little intuition on how to construct a "rich" query accurately representing the vacancies needs, leading to relatively many sparse queries prone to bias a model. Second, users did not invest as much effort as we would like in annotating, despite careful instruction, perhaps because they do not benefit directly from participating. Third, a sufficiently large dataset is required to learn generalizable patterns. Finally, there is potential in combining un-assessed and assessed data.

It is important to note that using implicit feedback – interpreting user click actions as relevance signals – without explicit assessments, is also possible, see e.g. [8]. Such "click data" is intrinsically less reliable, but the vast amount of it is supposed to cancel out noise. Due to the deceptively small amount of usable explicit assessments we encountered when conducting this project, as well the fact that we have vast amounts of active users in this systems like these, we intend to explore the direction of implicit feedback more in our future work.

References

1. Banon, S.: Elasticsearch. https://www.elastic.co/products/elasticsearch, [Online; accessed 03-July-2018]
2. Braun, H.: Applying Learning-to-Rank to Human Resourcing's Job-Candidate Matching Problem: A Case Study. Master's thesis, Radboud Universiteit (2017)
3. Burges, C.J.: From ranknet to lambdarank to lambdamart: An overview. Learning **11**(23-581), 81 (2010)
4. Cao, Z., Qin, T., Liu, T.Y., Tsai, M.F., Li, H.: Learning to rank: From pairwise approach to listwise approach. Tech. rep. (2007)
5. Chuklin, A., Markov, I., de Rijke, M.: Click Models for Web Search. Morgan & Claypool (2015). https://doi.org/10.2200/S00654ED1V01Y201507ICR043
6. Faliagka, E., Ramantas, K., Tsakalidis, A., Tzimas, G.: Application of machine learning algorithms to an online recruitment system (2012)
7. Fang, M.: Learning to Rank Candidates for Job Offers using Field Relevance Models. Master's thesis, University of Groningen & Saarland University (2015)
8. Joachims, T., Granka, L., Pan, B., Hembrooke, H., Gay, G.: Accurately interpreting clickthrough data as implicit feedback. In: ACM SIGIR Forum. vol. 51, pp. 4–11 (2017)
9. Karmaker Santu, S.K., Sondhi, P., Zhai, C.: On application of learning to rank for e-commerce search (2017)
10. Kazai, G.: Information retrieval evaluation with humans in the loop. In: IIiX (2014)
11. Kazai, G., Zitouni, I.: Quality management in crowdsourcing using gold judges behavior. In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. pp. 267–276. WSDM '16, ACM, New York, NY, USA (2016)
12. Kokkodis, M., Papadimitriou, P., Ipeirotis, P.G.: Hiring behavior models for online labor markets. In: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining. pp. 223–232. WSDM '15, ACM, New York, NY, USA (2015)
13. Li, H.: A short introduction to learning to rank. IEICE TRANSACTIONS on Information and Systems **94**(10), 1854–1862 (2011)
14. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA (2008)
15. Pérez-Iglesias, J., Pérez-Agüera, J.R., Fresno, V., Feinstein, Y.Z.: Integrating the probabilistic models bm25/bm25f into lucene (2009)
16. Radlinski, F., Joachims, T.: Query chains: Learning to rank from implicit feedback. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining. pp. 239–248. KDD '05, ACM, New York, NY, USA (2005). https://doi.org/10.1145/1081870.1081899, http://doi.acm.org/10.1145/1081870.1081899
17. Tax, N., Bockting, S., Hiemstra, D.: A cross-benchmark comparison of 87 learning to rank methods. Information processing & management **51**(6), 757–772 (2015)