Towards Comparing Recommendation to Multiple-Query Search Sessions for Talent Search

Mesut Kaya[1], Toine Bogers[1]

[1]*Department of Communication & Psychology, Aalborg University Copenhagen, A.C. Meyers Vænge 15, 2450 Copenhagen SV, Denmark*

Abstract

Query-level evaluation metrics such as nDCG that originate from field of Information Retrieval (IR) have seen widespread adoption in the Recommender Systems (RS) community for comparing the quality of different ranked lists of recommendations with different levels of relevance to the user. However, the traditional (offline) RS evaluation paradigm is typically restricted to evaluating a single results list. In contrast, IR researchers have also developed evaluation metrics over the past decade for the session-based evaluation of more complex search tasks. Here, the sessions consist of multiple queries and multi-round search interactions, and the metrics evaluate the quality of the session as a whole.

Despite the popularity of the more traditional single-list evaluation paradigm, RS *can* also be used to assist users with complex information access tasks. In this paper, we explore the usefulness of session-level evaluation metrics for evaluating and comparing the performance of both recommender systems and search engines. We show that, despite possible misconceptions that comparing both scenarios is akin to comparing apples to oranges, it is indeed possible to compare recommendation results from a single ranked list to the results from a whole search session.

In doing so, we address the following questions: (1) how can we fairly and realistically compare the quality of an individual list of recommended items to the quality of an entire manual search session; (2) how can we measure the contribution that the RS is making to the entire search session.

We contextualize our claims by focusing on a particular complex search scenario: the problem of *talent search*. An example of professional search, talent search involves recruiters searching for relevant candidates given a specific job posting by issuing multiple queries in the course of a search session. We show that it is possible to compare the search behavior and success of recruiters to that of a matchmaking recommender system that generates a single ranked list of relevant candidates for a given job posting. In particular, we adopt a session-based metric from IR and motivate how it can be used to perform valid and realistic comparisons of recommendation lists to multiple-query search sessions.

Keywords

evaluation, session-based recommendation, recruitment, job recommendation, search

1. Introduction

Arguably the most common application scenario for a recommender system is presenting the user with a single list of recommended items that are personalized to their tastes and interests, after which they engage with some (or none) of these items, thereby providing

Perspectives on the Evaluation of Recommender Systems Workshop (PERSPECTIVES 2022), September 22nd, 2022, co-located with the 16th ACM Conference on Recommender Systems, Seattle, WA, USA.
mkaya@ikp.aau.dk (M. Kaya); toine@ikp.aau.dk (T. Bogers)
https://mesutkaya.github.io/ (M. Kaya); https://http://toinebogers.com/ (T. Bogers)

feedback on the quality of the recommendations. As a result, this scenario is also the one most commonly simulated in offline evaluation with many standardized evaluation metrics available, such as RMSE, hit rate, and nDCG [1]. Nevertheless, there are many other non-standard application scenarios that do not fit this pattern and are more challenging to evaluate, such as engaging with a recommender system over multiple interactions in a session instead of with a single list [2], or providing recommendations for a group of users instead of a single user [3].

Another, less-researched scenario is that of human augmentation where, instead of a primary role, the recommender system can play a supporting role to aid the user in fulfilling their information need [4]. This type of human augmentation by predictive systems is what Raisamo et al. [4] refer to as augmented cognition. This can be especially valuable in cases where parts of the decision-making process can be automated, but where the final decision on which items to recommend still requires or benefits from human oversight.

An example of such a scenario is talent search (or job matchmaking), where recruiters attempt to identify relevant candidates for an open job posting and shortlist them by assessing their qualifications, such as their knowledge, skills, abilities, work experience, education level, and interests [5]. Talent search is typical example of a complex, professional information access task. Typically, recruiters first analyze a current job posting for the relevant job requirements and then search a database of candidate CVs with multiple queries and query reformulations until they have shortlisted an acceptable number of candidates.

In recent years, various approaches have been proposed that use AI techniques to assist recruiters in this matchmaking process, for instance by automatically extracting relevant skills from the job postings [6] or generating recommendations for relevant job candidates [7]. However, instead of replacing human recruiters with automatic recommendations, a better approach is often to augment recruiters' tasks by using job recommender systems in a supportive role to augment their cognitive abilities. Without the benefit of human oversight and the experience that recruiters bring to the table, the job recommendation technology currently in use in the HR industry has been shown to be too restrictive and at risk of producing unfair rankings of candidate CVs [8]. By presenting recruiters with candidate recommendations at the start of or in parallel with their normal search process, one could potentially reduce the effort needed per job posting in terms of queries submitted and reformulated, time spent in total and even increase the number of contacted relevant candidates.

At first glance, however, assessing the effectiveness of a RS in such a scenario may seem like comparing apples to oranges. How can we fairly and realistically compare the quality of an individual list of recommended items to the quality of an entire manual search session (RQ1)? And how can we measure the contribution that the RS is making to the entire search session (RQ2)? In this paper, we introduce a possible approach to comparing and evaluating recommendation and search that attempts to answer these research questions by adopting and adjusting session-level evaluation metric(s) from the field of IR [9, 10, 11, 12]. We argue this allows for a realistic comparison between single recommendation lists and entire search sessions, and that it supports offline evaluation.

In the remainder of this paper, we start by providing a brief overview of the related work on evaluating single-list recommendation and session-level search. We then contextualize our work by introducing a motivating example centered around talent search in Section 3 and show how we can adopt and adjust session-level evaluation metrics from the field of IR to answer our research questions. We discuss and conclude in Section 4.

2. Related Work

While online A/B testing experiments are generally recognized as providing the most realistic evaluation conditions for a recommender system, it is common to first run offline evaluation experiments to narrow down the space of promising candidate algorithms. Due to its lack of real user interaction, offline testing allows for rapid prototyping and comparison of a wide range of candidate algorithms at much lower cost [13, 1]. An important element of offline testing is to use valid and established evaluation metrics to make the offline evaluation as realistic as possible. While the evaluation of single-list recommendations appears to be a settled issue, this is far less clear for evaluating entire recommendation sessions, for comparing the results obtained using search and recommendation with each other, or for assessing the contributions of a recommender system in a supporting capacity.

2.1. Single-list evaluation

One of the most common evaluation setups involves presenting the user with a single list of recommended items and requesting feedback on the recommendations. In offline testing, this reference ranking is then compared to the recommendations produced by one or more algorithms [13, 1]. This setup was inspired by Cranfield paradigm dating back to the 1960s [14], which prescribes how retrieval algorithms should be evaluated offline [9, 13]. As a result, the RS community has adapted several standardized evaluation metrics from IR, with DCG being one of the most popular ones [1]. Discounted Cumulated Gain (DCG) is a measure of ranking quality that takes into account the varying degrees of relevance that different items may have for a user through the assignment of gain values, i.e., how much does the user gain from interacting with a specific item? Results lists that contain more highly relevant items and that return more highly relevant items near the top of the ranking represent more effective results lists. Formally, the effectiveness of a single result list, denoted as $f(Q)$ can be calculated by taking the inner product of a gain vector $\vec{g}$ and a discount vector $\vec{d}$ [12]:

$$f(Q) = \sum_{n=1}^{N} g_n(Q) \cdot d_n \tag{1}$$

where query Q returns N results, $g_n(Q)$ is the gain a user gets from the n-th result, and d_n is the discount factor for the n-th result. In IR, $g_n(Q)$ is mostly referred to as a graded relevance judgment with ratings being common in RS research. The discount factor d_n is commonly estimated as the probability that searcher will interact with the n-th result and ensures that highly relevant items that are ranked further down the list are penalized more. As lists of recommendations can differ in size, comparing discounted gain vectors directly is problematic. To address this, Järvelin and Kekäläinen [15] proposed normalizing the discounted gain vector against the ideal gain vector to produce a single-figure normalized DCG (nDCG) score at each position n.

2.2. Session-based evaluation

Information seeking often takes the form of a multi-stage process with several rounds of query formulation and interaction with search results [16], especially in complex, professional search scenarios such as talent search.

As single-list evaluation metrics such as nDCG are aimed at measuring the effectiveness of a single query or list of recommendations, several session-level evaluation metrics have been proposed over the years [9, 10, 11, 12]. All of these session-level metrics generalize to the following Eq. 1:

$$f(s) = \sum_{m=1}^{M} \sum_{n=1}^{N} g_{m,n}(Q_m) \cdot d_{m,n} \tag{2}$$

where s is a search session in which the searcher submits M queries with gain $g_{m,n}(Q_m)$ and discount factor $d_{m,n}$ for the n-th result returned for the m-th query of s respectively. One instantiation of Eq. 1 is the Session Discounted Cumulated Gain (or sDCG), an extension of nDCG to entire search sessions as proposed by Järvelin et al. [10]. For sDCG the discount factor $d_{m,n}$ is formulated as follows:

$$d_{m,n}(sDCG) = \frac{1}{(1 + log_{b_r} n)(1 + log_{b_Q} m)} \tag{3}$$

This discount factor has two different discount components: b_r for the rank discount also part of nDCG, and b_Q for the query discount, which penalizes results interacted with near the end of the session more. These two components can be controlled by changing their logarithm base values, where larger values of b_r and b_Q can be used to model users that spend more time, submit more queries and assess more search results. For a session consisting of a single query, sDCG is equal to DCG. The main difference between the different instantiations of Eq. 1 is in the use of different discount factors, such as the sRBP metric proposed by Lipani et al. [11].

There has also been related work on session-based recommendation and evaluation in the RS community. In session-based RS, the goal is to generate recommendations for an ongoing session based on a time-ordered sequence of interactions, organized in sessions [2]. Offline evaluation of session-based RS does not differ much from single-list evaluation: interaction data is split into training and test splits, after which the training data is used to learn a recommendation model to predict the held-out preferences in the test set. To the best of our knowledge, no one has proposed using session-level metrics to evaluate the success of an entire session of RS interactions.

2.3. Combining search and recommendation

As we argued in Section 1, RS can also be used in conjunction with other information access methods, such as search engines in the case of talent search. Dzyabura and Tuzhilin [17] studied the problem of how best to combine search and recommendation results as they compared the performance of pure recommendation and pure search separately to that of an interleaved combination of search and recommendation results lists. They evaluated these three conditions on their recall of relevant items and found that the combination outperformed the individual results lists. However, none of the individual components represented entire interaction sessions, which is contrast with our focus.

Another approach that considered the intersection of search and recommendation was the narrative-driven recommendation scenario presented by Bogers and Koolen [18]. They defined narrative-driven recommendation as a recommendation scenario where the recommendation process is driven by both a log of the user's past interactions as well as a narrative description of their current interest(s). This has many similarities with the case of talent search being supported by a RS: past search behavior and past interactions between job seekers and job postings constitute the interaction data and the job posting can be seen as a representation of the current information need of the recruiter. The authors illustrated their RS scenario with book recommendation, but they only evaluated single lists using nDCG and never considered search sessions.

To the best of our knowledge there are no other examples of direct comparisons between search and recommendation or evaluations of the contribution of RS in a supporting role. In the next section, we introduce our use case and argue for the value of session-based evaluation metrics such as sDCG for evaluating such an human-augmentation scenario.

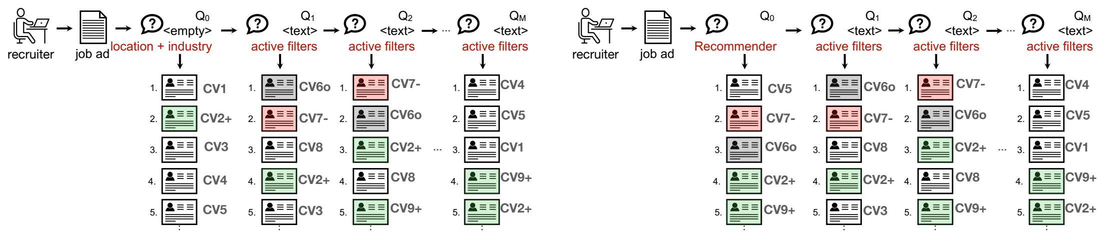
3. Use Case: JobIndex

To contextualize our advocacy of session-based evaluation metrics from IR to help with the offline evaluation of a recommender system's contributions in a human-augmentation scenario, we will first introduce a concrete use case for talent search at Jobindex[1], a major job portal and recruitment agency in Scandinavia. We start by explaining their recruitment process and the augmentation scenario in more detail, followed by notational definitions and a description of our dataset. We then present our results to aid in answering our research questions.

3.1. Recruitment at Jobindex

Figure 1a is a visualization of the recruitment process at Jobindex. For each open job posting, recruiters extract the most important job requirements and then search Jobindex's CV database for relevant candidate. When a recruiter starts a new job by selecting an open job posting j, the search engine uses the location and the industry of the job posting as the initial query (Q_0), in effect performing what Allan et al. [19] refers to as zero-shot retrieval. The results for this initial query tends to be imprecise but with

[1]https://www.jobindex.dk


recruiter
job ad
Q_0 <empty>
location + industry
Q_1 <text>
active filters
Q_2 <text>
active filters
Q_M <text>
active filters
1. CV1 2. CV2+ 3. CV3 4. CV4 5. CV5
1. CV6o 2. CV7- 3. CV8 4. CV2+ 5. CV3
1. CV7- 2. CV6o 3. CV2+ 4. CV8 5. CV9+
1. CV4 2. CV5 3. CV1 4. CV9+ 5. CV2+
recruiter
job ad
Q_0
Recommender
Q_1 <text>
active filters
Q_2 <text>
active filters
Q_M <text>
active filters
1. CV5 2. CV7- 3. CV6o 4. CV2+ 5. CV9+
1. CV6o 2. CV7- 3. CV8 4. CV2+ 5. CV3
1. CV7- 2. CV6o 3. CV2+ 4. CV8 5. CV9+
1. CV4 2. CV5 3. CV1 4. CV9+ 5. CV2+

(a) Original recruiter search session (b) Recommendation-augmented search session

Figure 1: Visualization of the search behavior of Jobindex recruiters and its potential for cognitive augmentation. Figure (a) shows the original search behavior of Jobindex recruiters. They search for relevant candidates for a job posting in multiple stages (Q_0 to Q_M) and assess a subset of the returned CVs. All colored CVs are judged as relevant by the recruiter and shortlisted for contact; all white CVs are deemed not relevant. Contacted candidates can respond positively (green), negatively (red) or not at all (grey). Figure (b) shows the scenario where the talent search process is augmented using a recommender system. To avoid reducing the autonomy of the recruiters, the initial results list for Q_0 is replaced by the recommendations generated by a job RS.

high recall, to give recruiters an indication of the difficulty of the task . Recruiters can inspect the results list Q_0 to see whether it contained relevant candidates (e.g., candidate CV2+).

After this initial results list, the recruiters start their search process with Q_1 by formulating a search query and (de)selecting filters. After issuing a new query, recruiters are shown another results list corresponding to the query in question. After each query, they may identify more relevant candidates (e.g., CV6o and CV7- for Q_1 and CV9+ for Q_2). It is possible (and common) that the same CV is returned different queries (e.g., CV2+). Recruiters are required to identify at least 20 relevant candidates based on their historic response rates, so they will keep assessing results lists, shortlisting candidates and reformulating their queries until they reach this number at query Q_M. Next, recruiters send the shortlisted candidates a message, to which they can respond positively (green in Figure 1a), negatively (red), or not at all (grey). White CVs in Figure 1a were not seen as relevant candidates by the recruiter.

This talent search process at Jobindex is a clear example of professional search and falls somewhere between pure lookup search and pure exploratory search. Typically, recruiters do not know in advance who they are looking for and will reformulate their queries and adjust their strategy based on their intermediate findings like in exploratory search. However, the recruiters have a clear goal in mind, are usually experts on the domain of their goal, and know how to achieve their goals—all uncommon for exploratory search [20].

3.2. Augmenting the recruiters

Figure 1b shows how RS can assist recruiters for this search session. The ideal outcome for the recommendation algorithm would be to return a ranked list that (1) contains all the CVs that the recruiter would find after issuing multiple queries in the course of a search session; (2) ranks the most relevant candidates near the top (i.e., shortlisted candidates that also responded positively in the example, such as CV2+ and CV9+)); and (3) ranks relevant candidates higher if the recruiter spent more effort finding them. This way, the RS can arguably assist the recruiter in such a way that they will issue fewer number of queries, spend less time and effort.

In principle, exposing the recruiters to this list of CV recommendations need not be restricted to only the start of the search process. The automatic recommendations could be combined or interweaved with their manual search results at each search step either automatically or by giving the recruiters full control over the combination process. This augmented cognition scenarios shares several similarities with interactive recommendation [21], although they are far from identical.

Estimating the recruiters' effort to find relevant candidates is a difficult task. According to Vakkari [16]'s searching-as-learning model and to our recent work [22] analyzing the information seeking behaviour of the Jobindex recruiters, we see that recruiters tend to increase their use of filters and formulate longer queries with more diverse query terms and advanced search operators as they progress in their search task. That is why, in this work, we make the assumption that relevant candidates that are found at the later stages of a search session are the ones that recruiter spent more effort on. We can use this information to make the offline evaluation of the recommender system's contribution more realistic. For instance, we would expect a perfect recommendation algorithm to rank CV9+ before CV2+ since recruiter, arguably, spends more effort to find CV9+ and the gain for CV9+ would be greater than that of CV2+. Of course, in our example the ordering of the relevant candidates in the recommendation list depends on the objective of the recommender, i.e., assisting recruiter such that they will submit fewer number of queries, spend less time etc. If we were to measure the effectiveness only by looking at the response types of the contacted candidates, we could argue that the order of CV2+ and CV9+ does not really matter, since both responds positively.

3.3. Notation

For a job posting j, a recruiter performs the talent search task and issues M queries during the course of a session $s = \{Q_1, Q_2, \dots, Q_M\}$. For each submitted query $Q_m \in s$, a ranked result list $L_m = \{r_1, r_2, \dots, r_N\}$ that contains N CVs is returned where r_1 is the first shown CV and r_N is the last shown CV respectively. The session can be represented as $s = \{L_1, L_2, \dots, L_M\}$. For s, a recruiter selects K relevant candidate CVs from $\bigcup_{m=1}^{M} L_m$ and sends them a contact message about j, where the set of relevant candidates for session s can be referred as $C_s = \{\langle r_1, rsp_1, fs_1 \rangle, \dots, \langle r_K, rsp_K, fs_K \rangle\}$, where rsp_k is the response type (positive, negative or no response) given by contacted CV r_k, and fs_k is the first time CV r_k has been shown to the recruiter during the course of the s, i.e., after the m-th query for instance.

As argued in the previous section, augmenting recruiter's search process using a RS fits best in the initial Q_0 results list through zero-shot retrieval or zero-query search [19], thereby anticipating the recruiter's needs and predicting which candidate CVs are most relevant for the job posting in question. We can compute and denote the recommendation list generated for a job posting j as $RL_j = \{r_1, r_2, \dots, r_N\}$, which contains a ranked set of CVs sorted by relevance as computed by some recommender algorithm. In our use case, for all job postings available in the dataset provided by Jobindex, we compute recommendation list by using a content-based recommender that uses job-title based embeddings of job postings and CVs to recommend relevant candidates for each job [23]. It is important to note that the focus of this paper is not on developing a new recommendation algorithm or claiming that our example algorithm is the best choice. Instead, we use this simplistic baseline algorithm to illustrate the evaluation approached. While this is a non-personalized run where the job ad is used as a query, one could use interaction data between job seekers and CVs to integrate item-to-item recommendations and even personalize the results per recruiters based on their past selection of filters. Different recommendation algorithms that have been proposed recently can be used instead of the content-based recommender we are using to recommend relevant candidates for jobs [24, 25].

3.4. Dataset

We use search log data provided by Jobindex consisting of 7,425 unique search tasks performed by Jobindex's recruiters in the period of February 12, 2022 to April 20, 2022. Each of these search tasks corresponds to a single job posting which is handled by a single recruiter. For each of the completed tasks, Jobindex logs the following data that we can use as part of our use case evaluation:

Retrieved CVs For each submitted query in a session, a list of shown CV IDs based on the submitted query that includes the rank of the CV in that query's result list. All CV IDs are anoynmized and no personal information is available in the dataset.

Response data For each completed session, response data on the set of contacted candidates is available, which includes the contacted candidate's CV ID and the type of their response to the contact message (positive, negative, none).

3.5. Results

3.5.1. Comparing search sessions to recommendation lists

For a given session s, we could consider the CVs that have been assessed as relevant by the recruiters, C_s as test set and use evaluation metrics like nDCG to estimate and compare the effectiveness of the different recommendation lists. However, single-list evaluation metrics such as nDCG are not suitable for comparing single lists to the effectiveness of entire search sessions, since they will miss the searcher's reformulation effort.We therefore propose using sDCG to measure the effectiveness of the search session and also recommendation list by using the example from Figure 1 and the data provided by Jobindex.

First, for each session s, we assign relevance weights based on the response types of the contacted CVs, C_s. There are a few constraints for assigning these relevance weights in the Jobindex use case. First, we argue that all of the contacted candidates have been shortlisted and contacted by the recruiter should be seen as relevant, even if they respond negatively. Considering the job a good feedback but not wanting to switch jobs at the moment is also covered by the negative response, so these negative responses could represent a relevant candidate. In addition, the relevance weights for the different responses—positive, negative and no response (= 'nr')—should follow $w_{pos} > w_{nr} > w_{neg}$. While these relevance weights are contextually dependent, we set them to $w_{pos} = 10$, $w_{nr} = 2$ and $w_{neg} = 1$ for our use case to allow us to optimize more for the positive responses[2].

To show how the sDCG metrics are calculated, we include Table 1, which corresponds to the toy example shown in Figure 1a. It shows the vectors representing each query result list containing the gain (G), discounted gain (sDG), discounted cumulated gain (sDCG) and normalized sDCG (nsDCG) values for the initial results list Q_0, the three queries issued by the recruiter Q_1–Q_3. We also include an artificial top-5 recommendation list from Figure 1b which takes the place of the initial results list in Figure 1a. Because sDCG defaults to nDCG for single-query sessions, we can use the same metric to compare the single list of recommendations to the full search session and compute the G, sDG, sDCG and nsDCG values for it, allowing us to compare to seemingly disparate information access artefacts.

Note that some CVs may be returned multiple times for different queries in a session. In order to compute the gain vectors, we must decide whether to consider the gain only the first time a CV is returned or every time it is returned by the system. In our dataset, we do not have the information about at which point (m-th query) the recruiter assessed a candidate CV as relevant, so we include a CVs gain value every time it is shown to the recruiter. For computing the nsDCG values, we need to construct the ideal gain vector for each query. In the ideal situation, a recruiter would only have to issue a single query Q to find all the relevant candidates to be contacted C_s in the result list returned for the Q. When calculating the ideal gain vector for the entire search session, we follow Järvelin et al. [10] and concatenate this ideal gain vector M times for each query in the session, as each submitted query can be considered to be another attempt to find ideal result list.

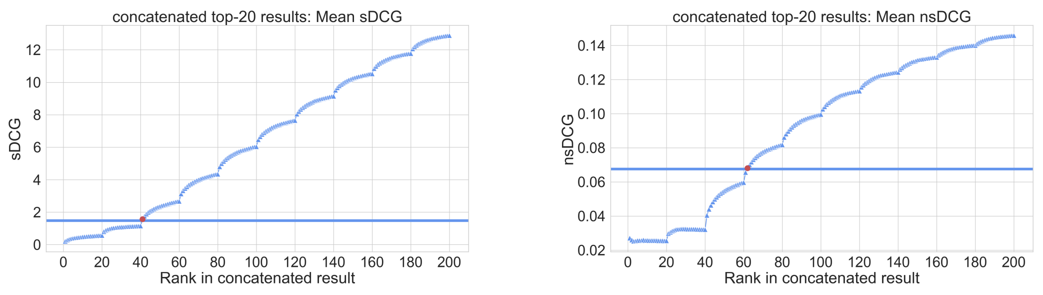
To produce a summary statistic similar to nDCG, we calculate the average sDCG and nsDCG values at different ranks (RQ1). This allows us to compare at which rank N of query M the sDCG for the search session surpasses the sDCG of the recommendation algorithm. In other words, how much of the recruiter's work can the recommender system do before the recruiter's experience takes over. We also plot the gain vectors containing the sDCG values to visually show this intersection point, as shown in Figures 2a and 2b.

Figures 2a and 2b show the sDCG and nsDCG values averaged over the 7,425 unique search sessions in our Jobindex dataset. For each session s we computed the sDCG and nsDCG values for top-10 queries and their top-20 results. In addition, the vertical lines represent the average sDCG and nsDCG scores for our baseline embeddings-based

[2]We note that, we have data about whether candidates responded to a job suggestion positively or not, we do not have any information about whether they actually applied for the position or whether they were hired. This means, if available applying or being hired information can be assigned larger relevance weights.

Table 1
Example search session gains (G), session discounted gains (sDG), session discounted cumulated gains (sDCG), and normalized sDCG (nsDCG) based in Figure 1. For discounts in Equation 3, we use b_r=2 and b_q=4. For sDCG value for this search session based on Equation 2, with number of queries M=4 and number of search results N=5. We also include the recommendation list from Fig. 1b as *Recommendations* and its corresponding G, sDG, sDCG and nsDCG values.

Query ID	Gain vectors
Q_0	G: $\langle 0, 10, 0, 0, 0\rangle$
	sDG: $\langle 0.0, 5.0, 0.0, 0.0, 0.0\rangle$
	sDCG: $\langle 0.0, 5.0, 5.0, 5.0, 5.0\rangle$
	nsDCG: $\langle 0.0, 0.33, 0.32, 0.31, 0.31\rangle$
Q_1	G:$\langle 2, 1, 0, 10, 0\rangle$
	sDG:$\langle 1.33, 0.33, 0.0, 2.22, 0.0\rangle$
	sDCG: $\langle 6.33, 6.67, 6.67, 8.89, 8.89\rangle$
	nsDCG: $\langle 0.28, 0.25, 0.25, 0.33, 0.33\rangle$
Q_2	G:$\langle 1, 2, 10, 0, 10\rangle$
	sDG:$\langle 0.56, 0.56, 2.16, 0.0, 1.68\rangle$
	sDCG: $\langle 9.45, 10.01, 12.16, 12.16, 13.84\rangle$
	nsDCG: $\langle 0.29, 0.28, 0.34, 0.34, 0.39\rangle$
Q_3	G:$\langle 0, 0, 0, 10, 10\rangle$
	sDG:$\langle 0.0, 0.0, 0.0, 1.67, 1.51\rangle$
	sDCG: $\langle 13.84, 13.84, 13.84, 15.51, 17.01\rangle$
	nsDCG: $\langle 0.34, 0.32, 0.32, 0.35, 0.39\rangle$
Recommendations	G:$\langle 0, 1, 2, 10, 10\rangle$
	sDG:$\langle 0.0, 0.5, 0.77, 3.33, 3.01\rangle$
	sDCG:$\langle 0.0, 0.5, 1.27, 4.61, 7.62\rangle$
	nsDCG: $\langle 0.0, 0.03, 0.08, 0.29, 0.47\rangle$



(a) sDCG, rec vs search

(b) nsDCG, rec vs search

Figure 2: sDCG (a), nsDCG (b) comparison of recommendation vs entire search session.

recommendation algorithm. The latter scores are represented as a horizontal line to show at what session-query position the recruiter's experience finally beats the recommender system. The further to the right this intersection point is, the closer the recommender system is compared to the entire search session. If the recommendation list corresponded to the ideal result list, then the horizontal line would always be above the curve representing the search session.

Figures 2a and 2b also have horizontal lines showing the sDCG and nsDCG values of recommendation lists computed by a recommendation algorithm. For Figures 2a and 2b, we can look at the point (marked with red in the plots) that searcher surpasses the recommendation list effectiveness during the course of multiple-query search session. This can be done by looking at the intersection point of the horizontal lines (sDCG of recommendation algorithm) and line plots (sDCG values for the search session). For instance, if we look at Figure 2b, we can see that nsDCG value of the multiple-query search session surpasses that of recommendation algorithm right at the beginning of the 3rd query. Ideally, the better the recommendation list generated is, the further to the right of the plot the intersection point is. This shows that by using sDCG we can effectively compare a recommender system supplying the initial zero-query search results to the recruiter's entire session in an offline setting.

We can also use sDCG to measure the actual contribution that the RS is making to the recruiters' search process (RQ2) once it is integrated into Jobindex's systems (as visualized in Figure 1b). In such an online evaluation scenario, we can set up an A/B test where we test their original search process to the recommendation-augmented process. Once the contacted candidates have had a chance to respond to the contact messages sent by the recruiters, we can again compute sDCG for both variants and compare them. For instance, in the toy example in Figure 1b and Table 1, the sDCG@5 score for the recommendation list would be 7.62, which would allow one to estimate the potential contribution of the recommendation list to the entire search session.

4. Discussion & Conclusions

In this paper, we investigated how session-level evaluation metrics can be used to evaluate and compare both recommender systems and search engines. Through our use case of Jobindex, we show that it is possible to compare the effectiveness of recommendations from a single ranked list to that from a whole search session with multiple queries issued. Although we used talent search as a use case in this paper, our proposed evaluation scenario can be used for any professional, complex information access tasks where RS can be used to augment the cognitive abilities of an information professional, such as patent search, search for legal documents. We believe that this is an under-investigated problem setting that deserves more attention.

4.1. Limitations

We note that this is a preliminary study to explore how to compare both recommender systems and search engines by using session-level evaluation metrics. There are some limitations of our current work. In this paper, we do not make a principled analysis of the robustness and the discriminative power of the adapted session-level evaluation metrics for the offline evaluation of recommender systems.

4.2. Future Work

Searcher effort can be incorporated into session-based evaluation and can be used to compare the performance of separate recommendation lists generated by different recommendation algorithms against each other using/adapting metrics like sDCG. For instance, for the toy example shown in Figure 1a, as we argued before , for CV9+ the effort that the recommendations can save for the recruiter is more than CV2+, then we should incorporate searcher's effort factor when comparing different recommendation lists. This factor for instance will give higher gain in this example to CV 9+ than CV2+.

One way to do this is to incorporate a dynamic value adjustment for a more realistic evaluation and comparison of different recommendation lists. Consider the gain factor in Eq. 1, $g_n(Q)$, which is the relevant judgment for the results. For completed search sessions, we know the set of returned search results that had been assessed by the recruiters and we also know the response type given by the contacted job seekers. We can multiply the relevance weights by a factor similar to recency effect proposed by Zhang et al. [12], where they consider the later submitted queries more important. For us, a relevant result that is first found in the later submitted queries is more important, since arguably by showing it in the recommendation list we can assist recruiter such that they submit fewer queries, spend less time to find that result. We can adjust gain factor $g_{m,n}(RS)$ by multiplying it to $e^{-\lambda(M-m)}$ [12], where λ is a parameter that reflects the rate of searcher's effort. $\lambda = 0$ means that searcher's effort is the same for all relevantly assessed results, $\lambda > 0$ gives larger weights to the results found at the later stages of a session. M is the total number of submitted queries in the search session that we use RS to assist, and m denotes the first time the n-th document in RS, if it is a relevant document, have been returned in the search session. This way, we dynamically adjust gain values by taking into account the possible effort the recruiter spent on finding a relevant document.

We leave this and the discussion of how to design better function(s) for dynamically adjusting the gain values based on, for instance, a historic analysis of the time and effort spent by searchers, to future work.

We are also planning to get feedback of job recruiters on the usefulness of the recommendations during their search sessions, and compare the results of the user study with the results of the proposed evaluation methodology.

Acknowledgments

This research was supported by the Innovation Fund Denmark, grant no. 0175-000005B.

References

[1] A. Gunawardana, G. Shani, Evaluating recommender systems, in: Recommender systems handbook, Springer, 2015, pp. 265–308.

[2] D. Jannach, M. Quadrana, P. Cremonesi, Session-based recommender systems, Recommender Systems Handbook (2022) 301.

[3] J. Masthoff, Group recommender systems: Combining individual models, in: Recommender systems handbook, Springer, 2011, pp. 677–702.

[4] R. Raisamo, I. Rakkolainen, P. Majaranta, K. Salminen, J. Rantala, A. Farooq, Human Augmentation: Past, Present and Future, International Journal of Human-Computer Studies 131 (2019) 131–143.

[5] J. A. Breaugh, Employee Recruitment: Current Knowledge and Important Areas for Future Research, Human Resource Management Review 18 (2008) 103–118.

[6] M. Liu, J. Wang, K. Abdelfatah, M. Korayem, Tripartite vector representations for better job recommendation, arXiv preprint arXiv:1907.12379 (2019).

[7] P. Montuschi, V. Gatteschi, F. Lamberti, A. Sanna, C. Demartini, Job recruitment and job seeking processes: how technology can help, It professional 16 (2013) 41–49.

[8] J. B. Fuller, M. Raman, E. Sage-Gavin, K. Hines, Hidden workers: Untapped talent, Harvard Business School, September (2021).

[9] M. Liu, J. Mao, Y. Liu, M. Zhang, S. Ma, Investigating cognitive effects in session-level search user satisfaction, in: Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining, 2019, pp. 923–931.

[10] K. Järvelin, S. L. Price, L. M. L. Delcambre, M. L. Nielsen, Discounted cumulated gain based evaluation of multiple-query IR sessions, in: Proceedings of the IR research, 30th European conference on Advances in information retrieval, ECIR'08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 4–15.

[11] A. Lipani, B. Carterette, E. Yilmaz, From a User Model for Query Sessions to Session Rank Biased Precision (sRBP), in: Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 109–116. URL: https://doi.org/10.1145/3341981.3344216. doi:10.1145/3341981.3344216.

[12] F. Zhang, J. Mao, Y. Liu, W. Ma, M. Zhang, S. Ma, Cascade or Recency: Constructing Better Evaluation Metrics for Session Search, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Association for Computing Machinery, New York, NY, USA, 2020, pp. 389–398. URL: https://doi.org/10.1145/3397271.3401163.

[13] D. Valcarce, A. Bellogín, J. Parapar, P. Castells, Assessing ranking metrics in top-n recommendation, Information Retrieval Journal 23 (2020) 411–448. doi:10.1007/s10791-020-09377-x.

[14] E. M. Voorhees, The Evolution of Cranfield, in: Information Retrieval Evaluation in a Changing World, Springer, 2019, pp. 45–69.

[15] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of ir techniques, ACM Transactions on Information Systems (TOIS) 20 (2002) 422–446.

[16] P. Vakkari, Searching as learning: A systematization based on literature, Journal of

Information Science 42 (2016) 7–18.

[17] D. Dzyabura, A. Tuzhilin, Not by search alone: How recommendations complement search results, in: Proceedings of the 7th ACM Conference on Recommender Systems, RecSys '13, Association for Computing Machinery, New York, NY, USA, 2013, p. 371374. URL: https://doi.org/10.1145/2507157.2507231. doi:10.1145/2507157.2507231.

[18] T. Bogers, M. Koolen, Defining and Supporting Narrative-driven Recommendation, in: RecSys '07: Proceedings of the 11th ACM Conference on Recommender Systems, 2017, pp. 238–242.

[19] J. Allan, B. Croft, A. Moffat, M. Sanderson, Frontiers, challenges, and opportunities for information retrieval: Report from swirl 2012 the second strategic workshop on information retrieval in lorne, in: Acm sigir forum, volume 46, ACM New York, NY, USA, 2012, pp. 2–32.

[20] R. W. White, R. A. Roth, Exploratory Search: Beyond the Query-Response Paradigm, Synthesis Lectures on Information Concepts, Retrieval, and Services 1 (2009) 1–98.

[21] C. He, D. Parra, K. Verbert, Interactive Recommender Systems: A Survey of the State of the Art and Future Research Challenges and Opportunities, Expert Systems with Applications 56 (2016) 9–27.

[22] M. Kaya, T. Bogers, Under Review (Anon.).

[23] M. Kaya, T. Bogers, Effectiveness of job title-based embeddings on résumé-to-job-ad recommendation, in: Proceedings of the RecSys in HR 2021 workshop, 2021, pp. 35–41.

[24] E. Lacic, M. Reiter-Haas, T. Duricic, V. Slawicek, E. Lex, Should we embed? a study on the online performance of utilizing embeddings for real-time job recommendations, in: Proceedings of the 13th ACM Conference on Recommender Systems, 2019, pp. 496–500.

[25] D. Lavi, Learning to match job candidates using multilingual bi-encoder bert, in: Fifteenth ACM Conference on Recommender Systems, 2021, pp. 565–566.