How to Get Them a Dream Job?

Entity-Aware Features for Personalized Job Search Ranking

Jia Li
University of Illinois at Chicago
1200 W Harrison St
Chicago, IL, USA
jli213@uic.edu

Dhruv Arya
LinkedIn
2029 Stierlin Ct
Mountain View, CA, USA
darya@linkedin.com

Viet Ha-Thuc
LinkedIn
2029 Stierlin Ct
Mountain View, CA, USA
vhathuc@linkedin.com

Shakti Sinha
LinkedIn
2029 Stierlin Ct
Mountain View, CA, USA
ssinha@linkedin.com

ABSTRACT

This paper proposes an approach to applying standardized entity data to improve job search quality and to make search results more personalized. Specifically, we explore three types of entity-aware features and incorporate them into the job search ranking function. The first is query-job matching features which extract and standardize entities mentioned in queries and documents, then semantically match them based on these entities. The second type, searcher-job expertise homophily, aims to capture the fact that job searchers tend to be interested in the jobs requiring similar expertise as theirs. To measure the similarity, we use standardized skills in job descriptions and searchers' profiles as well as skills that we infer searchers might have but not explicitly list in their profiles. Third, we propose a concept of entity-faceted historical click-through-rates (CTRs) to capture job document quality. Faceting jobs by their standardized companies, titles, locations, etc., and computing historical CTRs at the facet level instead of individual job level alleviate sparseness issue in historical action data. This is particularly important in job search where job lifetime is typically short. Both offline and online experiments confirm the effectiveness of the features. In offline experiment, using the entity-aware features gives improvements of +20%, +12.1% and +8.3% on Precision@1, MRR and NDCG@25, respectively. Online A/B test shows that a new model with these features is +11.3% and +5.3% better than the baseline in terms of click-through-rate and apply rate.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: [Search process]

KDD '16, August 13 - 17, 2016, San Francisco, CA, USA

ISBN 978-1-4503-4232-2/16/08…$15.00

DOI: http://dx.doi.org/10.1145/2939672.2939721

Keywords

Information Retrieval; Learning-to-Ranking; Personalization

1. INTRODUCTION

Traditional information retrieval systems, and particularly Web search engines, have focused on keyword-matching. In this search paradigm, users typically input their information needs as a set of keywords and the search engines match the keywords with documents and use some additional signals, such as document popularity (e.g., document historical click through rate, PageRank etc.) to find relevant documents. This paradigm makes it very simple and easy for users to use. Moreover, it allows the efficient retrieval of documents at the Web scale. However, while this paradigm has been effective for the majority of the queries on generic Web search engines, it does not work that well for LinkedIn job search. In our use case, user information needs are typically rather complicated. For example, when a user issues query "software engineer Cambridge Microsoft", he or she does not mean to find a job that its description contains these keywords. Instead, he or she looks for a job with *title* software engineer at *company* Microsoft in Cambridge *city*. Thus, it is important to understand the structure of the information need. Moreover, even if we can structure the query as above, it is still unclear which city does "Cambridge" refers to. Even assuming we know that "Cambridge" refers to the city in Massachusetts (USA), there could still be plenty of Microsoft software engineer jobs in the city. In this case, which specific ones match best with the user's interests and expertise? Thus, it is crucial to go beyond keyword-matching when scoring documents to provide personally relevant results on LinkedIn job search.

Semantic search, on the other hand, represents information needs and documents in a structured way and semantically matches the information needs with the documents. A challenge with semantic search is that it is typically difficult for an end-user to describe his or her information need in a semantic representation. Moreover, semantic search is often restricted by concepts and relations predefined in a knowledge base. Thus, it does not scale well to open and dynamic document sets like the Web.

This work aims to bridge the two search paradigms to make LinkedIn job search experience more relevant and personalized yet still simple to use while keeping it efficient at the Web scale. From a user perspective, he or she is still able to describe their information needs in a free-text form. Given user queries, we extract and standardize the entities, such as, *job title*, *company*, *skills* and *location*. On the document side, we leverage standardized entities mentioned in job descriptions and include them in the search index. Based on the standardization information, we construct entity-aware features matching queries and documents as well as entity-faceted historical CTR features capturing global document qualities (query-independent). Moreover, as mentioned in the motivating example above, even when a query is perfectly understood, the query is not enough to represent user information need. To overcome this issue, we leverage standardized skills in members' profiles and job descriptions to capture searcher-job expertise homophily. For instance, if the searcher is a machine learning expert, he or she will be more likely to apply for software engineer jobs focusing on machine learning domain rather than on software system infrastructure. Thus, the ranking function should rank the former higher than the latter. Finally, to combine these features, we apply learning to rank technique to automatically learn a personalized search ranking function.

We conduct offline and online experiments on LinkedIn job search. Our experiments confirm the benefit of using standardized entities in the ranking function to make job search results more relevant and personalized. Specifically, in offline experiment on randomized data, a new model with the proposed features improves over the baseline without the features +20%, +12.1% and +8.3% in terms of Precision@1, MRR and NDCG@25, respectively. Online A/B test shows that the features can improve click-through-rate and apply rate, the two most important metrics of the product, +11.3% and +5.3%, respectively. At the time of this writing, the new ranking model has served all of LinkedIn job search traffic.

The main contributions of this paper are the following:

1. Proposing a new approach to incorporating entity data into personalized ranking functions for job search. Specifically, we propose three classes of entity-aware features including:
    - Query-job matching: Extract and standardize entities mentioned in queries and documents, such as, job titles, skills, companies and locations. Then, construct features matching queries and documents based on these entities.
    - Searcher-job expertise homophily: Analyzing log data, we discover that job searchers tend to be interested in the jobs requiring similar expertise as theirs. Thus, we propose features capturing searcher-document expertise homophily based on the standardized skills in job descriptions and users' profiles as well as the skills that members might have but not explicitly list. These features make results more personalized to the searchers.
    - Job historical CTRs: We introduce a concept of entity-faceted historical CTRs to capture job document popularity. Job documents are faceted on job attributes like company, title or location. Then, historical CTRs are computed as the facet level to overcome data sparseness issue.
2. Presenting practical challenges when deploying the entity-aware features in LinkedIn job search engine such as ambiguity, missing values, sparseness, parameter tuning, scalability, online efficiency, biases in log data and how to combine the new features with existing ones in an optimal way as well as solutions for these challenges.
3. Experimenting and demonstrating significant benefit of using standardized entity data for personalized search ranking in an industrial setting serving hundreds of millions of users.

The rest of the paper will be organized as follows. Section 2 reviews related work. Section 3 presents query-document matching features based on standardized entities in queries and documents. Section 4 discusses searcher-document features capturing expertise homophily. Section 5 details how we construct entity-faceted historical click-through-rate features. In Section 6, we present a learning-to-rank approach to combine the proposed features and the existing ones. Offline and online experimental results are discussed in Section 7. Finally, concluding remarks are in Section 8.

2. RELATED WORK

2.1 Semantic search

Semantic search has been an active research area recently. Generally speaking, semantic search uses semantics to make search systems more effective. More specifically, semantic search approaches extract semantic meanings and structures from search queries and documents and exploits them in search process [6]. The main approach to semantic search typically represents queries and documents in a structured way and applies semantic query languages like SPARQL [19] to retrieve results.

SHOE system [12] is one of the early work in this direction. It annotates Web pages with semantic information. On query side, users specify their information needs based on nodes and relations in an ontology. Swoogle [8] and Corese [4] are Semantic Web search engines. These engines store documents in Resource Description Framework (RDF) format and provide structured query languages to retrieve the documents. Similarly, Zhong et al. [28] represent queries and documents by conceptual graphs then propose an ontology-based matching algorithm estimating the similarity between a query and a document.

A common problem of these approaches is that it is challenging for end users to specify their information needs in such structured ways [24]. Also, these approaches are often restricted by concepts and relations predefined in ontologies. Thus, they do not scale to open and dynamic document sets like the Web [7]. Moreover the algorithm matching queries and documents in the structured representation are typically computationally expensive. Thus, it is challenging for industrial search engines to apply them in real time at the Web scale.

2.2 Enhancing traditional IR with semantics

Another research direction related to our work is using semantic technology to enhance traditional keyword-based information retrieval. Guha et al. [9] propose an approach to improve keyword-based search by using data retrieved from the Semantic Web. Specifically, given a text query, the

terms in the query are then mapped to nodes in the Semantic Web. They also propose heuristics to resolve ambiguity when mapping. Then, the nodes are used to augment the query. SemSearch [24] also allows users to specify their information needs by free text. Then, it maps the terms into classes, properties or instances in a knowledge base. A novel aspect of the work is that for a query containing multiple terms, the whole semantic meanings of all of the terms are taken into account. Fernandez et al . [7] introduce a new architecture to scale semantic search by supporting both pure semantic search on ontologies and enhancing traditional keyword search with semantics. In particular, their approach deals with knowledge incompleteness by switching from the former to the later search paradigm when there is no match in the ontologies. Buscaldi et al. [2] use semantic relationship such as synonymy in WordNet to compute similarity of a document and a query and then to rank the results. More recently, Dalton et al. [5] propose an approach called entity feature query expansion that expands user queries by structured data in knowledge bases. Given a query, the approach first annotates it with both explicit and latent entities and link them to two knowledge bases including Wikipedia and Freebase. Then, the corresponding structured data is used to do query expansion. The expanded queries are used to match with documents in retrieval phase. They show effectiveness of the approach on TREC text collections.

Compared to the previous work, our work is different to them in the following aspects. First, unlike the most of the previous approaches focusing on semantically matching documents and queries, our work goes beyond that. In particular, our work exploits standardized entities to match a searcher and documents to make results personalized as well as to model document popularities (query-independent) on various facets. As shown in evaluation section, these feature categories give even higher impacts than document-query matching. Second, even though our document-query matching approach is similar to the ones in the literature, our approach is customized for job search domain. To the best of our knowledge, there is no prior work focusing on using semantic knowledge in job search. Third, our approach uses entity data at a feature level in personalized ranking function (as opposed to query expansion level in some previous work). At the feature level, by applying learning to rank, our approach can softly combine semantic features and traditional features in an optimal way learnt from training data instead of making a hard switch between semantic search and keyword search as in some previous work, such as [7]. Finally, besides typical offline experiments, we experiment on live traffic on an industrial search engine. The online metrics are derived from searcher actions, thus they allow demonstrating effectiveness of the approach in a personalized setting.

2.3 Personalized Search

An important aspect of our work is personalization, which is also an active research area in the literature. The key idea behind personalized search is that different users might have different backgrounds and interests. As a result, they might look for different results even when issuing the same queries [1]. Thus, the one-size-fit-all model is not effective. There are multiple ways to achieve personalization in search. One direction is to ask users to describe their interests. For instance, Google personalized web search allows users to explicitly select some of the predefined topics to specify their interests. Another direction is to implicitly model user's interests. Sugiyama et al. [22] and Sontag et al. [21] propose approaches to model user's long-term interests. Some other work like [20, 25] focuses on immediate history to represent user's short-term interests. When data is too sparse to model user's interests individually, personalization could be achieved by grouping users in cohorts [27]. When a user issues a query, the corresponding cohorts can be used to customize results for the user.

A novel aspect of our work in terms of personalization is that we propose a new concept of expertise homophily to personalize job search results. Expertise homophily is measured by similarity between user's expertise and expertise requirement of each job. Moreover, to represent user's expertise, we do not only use standardized skills explicitly in a user profile but also infer skills that the user might have. The inference step is done by collaborative filtering technique exploiting skill co-occurrence patterns in the whole member base.

3. ENTITY-AWARE DOCUMENT-QUERY MATCHING

As mentioned before, when a user issues a query like "software engineer Microsoft", the user implicitly links the keywords to different typed entities, such as, title and company and expects the results matching with the information need in terms of the structure he or she has in mind. To satisfy this, we propose an approach that indexes documents in a structured way. At searching time, the user query is segmented and linked to one of the typed entities used in the document index. Then, we construct various features matching typed entities mentioned in the query with the corresponding ones in the documents. In the next subsections, we describe how documents are structured and indexed, how to segment a query and link the segments to entities and how to construct features semantically matching the query and documents.

3.1 Job Document Indexing

To help job seekers search and discover jobs, we build a search index on some of the key attributes of the job. Jobs on LinkedIn are structured to present the following key attributes: *job title*, *company*, *location*, *industry* and *skills*. An example is shown in Figure 1. When a job is posted on LinkedIn, it goes through a standardizer which looks at above mentioned fields to extract out standardized entities. The extracted entities are based on curated dictionaries built over time from our member profiles. The standardizer has been engineered through multiple iterations to understand what parts of the job posting are critical for different entities. The standardized job is then indexed and is searchable both on the entities as well as the free text as entered by the job poster. Since there is huge research literature on mapping textual mentions to entities, this paper does not emphasize this step. Instead, through out the paper, we focus on how to use these entities in job search ranking.

3.2 Query Processing

When a searcher enters a query, e.g. "software engineer Microsoft Cambridge", we first apply a query tagger to segment the query and tag the segments into entity types that

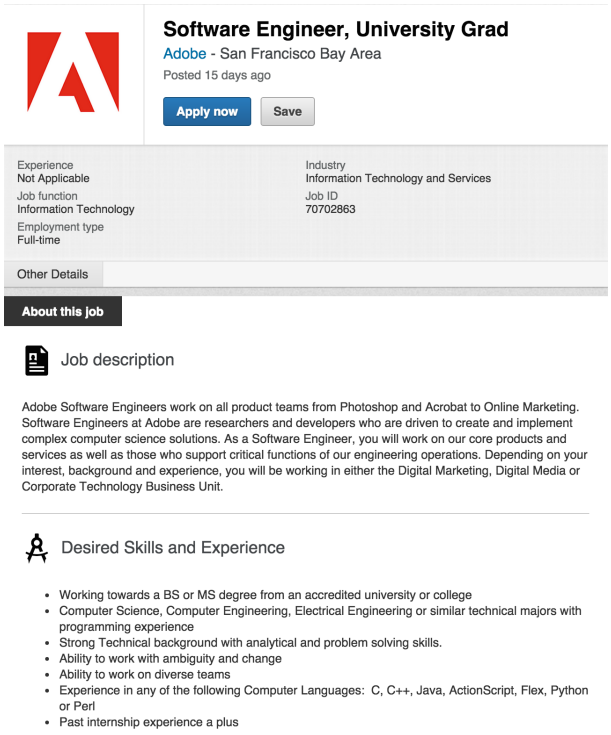

Figure 1: A job posting on LinkedIn. Entities in the job such as title, skills, company and location are extracted, standardized and indexed.

are important to the job search domain, such as, *job title*, *skill*, *company* and *location*, etc. For instance, the query above is segmented into "software engineer", "Microsoft" and "Cambridge" segments. These segments are then tagged into types of *job title*, *company* and *location*, respectively. Our query segmentation approach is similar to [23]. We refer interested readers to the reference for more details.

Given entity types of query segments, the next step is to map the segments into specific entities. The segments are matched against the dictionary of the corresponding types. In ambiguous cases, e.g., "security" (ambiguous title) or "Cambridge" (ambiguous location), the standardized entities in the searcher's profile are then used to resolve the ambiguity in a personalized way. For example, if the standardized location in the searcher's profile is Massachusetts (USA), "Cambridge" in his or her query is more likely to refer to the city in the US rather than the one in the UK. Similarly, if the searcher has title of computer security or related skills, such as, computer security or fraud detection, the term "security" in his or her query is more likely to refer to "computer security" title rather than "physical security".

3.3 Query-Document Matching Features

Given typed entities mentioned in user queries and in documents, we construct entity-aware features to capture the semantic similarity between the queries and the documents. In particular, we match entities in the queries and the ones with the same *types* in the documents. For instance, for the query "software engineer Microsoft Cambridge", after extracting standardized entities from the query as described above, we match title entity ("software engineer") with the title entity in each job document. Similarly, we match company entity and location entity with the corresponding ones in the job description. We consider two types of entity-matching: hard matching and soft matching. The former checks if the two entities (in the query and in the document) have the same identifier. This matching is able to capture synonymy relationship amongst different textual forms of the same entities, e.g., "software engineer" and "software developer".

We also consider the semantic similarity (soft matching) between two different but related standardized entities, such as, between title "software engineer" and title "software architect" or between skill "information retrieval" and skill "Web search". To measure such similarity, we use an approach leveraging the uniqueness of LinkedIn data collection including entity co-occurrence and career trajectory of more than 400 million members on LinkedIn. Intuitively, if two skills tend to co-occur in similar groups of members, they are likely to be related. Likewise, if there are a significant number of employees transferring between two companies, they are also likely to be similar.

4. SEARCHER-DOCUMENT EXPERTISE HOMOPHILY

As briefly presented before, for a query like "software engineer Microsoft" the searcher is not equally interested in every software engineer job at the company. Instead, if the searcher happens to be a machine learning expert, he or she is much more likely to be interested in software engineer jobs related to this field rather than software engineer jobs focusing on other domains. Thus, in many cases, user queries are not enough to represent user information need and interest. To complement the query, we exploit an idea of expertise homophily that captures the similarity between searcher's expertise and job expertise requirements to make job search results more personally relevant.

Homophily has been extensively studied in the context of social networks analysis [17, 14]. The main idea is that in a social network, a node tends to be connected or interact with other nodes that are similar to it. In the context of job search, we hypothesize that a job searcher tends to be interested in the jobs requiring similar expertise as his or hers. In this section, we first present how we represent expertise of searchers and expertise requirement of jobs via standardized skills. Then, we conduct an analysis to verify the hypothesis. Finally, we propose features capturing searcher-document expertise homophily.

LinkedIn allows members to add *skills* to their profiles. Typical example of skills for a software engineer would be - "Algorithm", "Data Mining", "Python", etc. On LinkedIn, there are about 35 thousand standardized skills. Members can also *endorse* skills of other members in their network. Thus, skills are an integral part of members' profiles to help them showcase their professional expertise (see Figure 2). On document side, as described in Section 3, each job description also associates with a set of standardized skills. Thus, in this paper, we use skills to represent searcher's expertise and job expertise requirement.

We conduct an analysis to verify the hypothesis that a job searcher tends to be interested in the jobs similar to his or her expertise using search logs. To avoid confounding factors, e.g., a feature in the original ranking function produc-

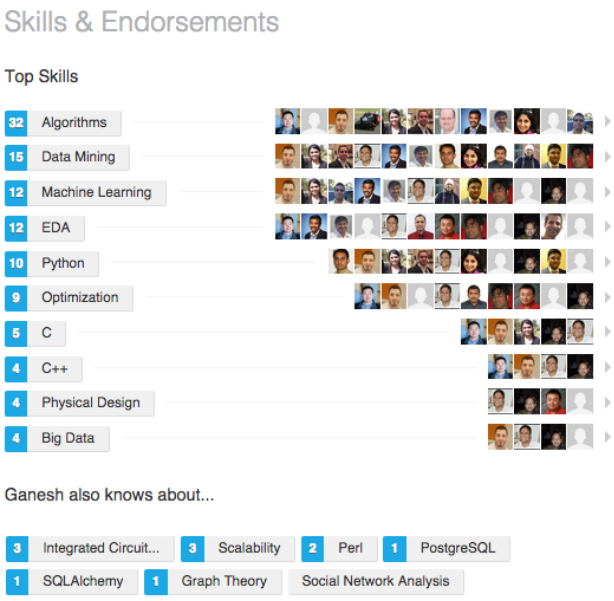

Figure 2: Example of Skill Section with Endorsements.

	Difference between overlapping bucket and non-overlapping bucket
CTR	+ 55%
Apply Rate	+ >100%

Table 1: CTRs and Apply Rates of job results with overlapping skills and job results without overlapping skills.

ing the logs that is correlated with searcher-result skill homophily and also impacts searcher's actions on results (since it impacts result rankings and searchers are biased towards top results), we use randomized data as described in details in Section 6.2. The data contains about 200 thousand searches from tens of thousands unique searchers. The results of the searches are divided into two buckets. The first bucket contains job results that their required skills overlap with the searcher's skills. The second bucket includes job results that do not require any skills the searcher has. The overlapping bucket ends up with about 16% and the non-overlapping bucket has 84% of the total number of results. For each bucket, we compute *CTR* and *apply rate*, which are defined as ratios of the number of job results that the searcher clicks (views) or applies, respectively, to the total number of results shown (the number of result impressions). As shown in Table 1, CTR and apply rate of the jobs with overlapping skills are 55% and more than 100%, respectively, higher than the jobs without overlapping skills. Thus, this analysis confirms the importance of skill homophily between searcher and results.

A key challenge of generating the expertise homophily feature is that searchers may not explicitly list all the skills they have on profiles. On the other hand, some of their skills might not be relevant to their core expertise. For instance, a machine learning researcher could have "nonprofit fundraising" skill [11]. To overcome these challenges, we estimate expertise scores of a member on the explicit skills and the ones he might have. Figure 3 describes the offline process to estimate the expertise scores. In the first step, we use a supervised learning algorithm combining various signals on LinkedIn such as skill-endorsement graph page rank, skill-profile textual similarity, member's seniority, etc. to estimate the expertise score, i.e., $p(expert|member, skill)$. After this step, the expertise matrix (E_0) is very sparse since we can be certain only for a small percentage of the pairs. In the second step, we factorize the matrix into member and skill matrices in K-dimensional latent space. Then, we compute the dot-product of the matrices to fill in the "unknown" cells. The intuition is that if a member have "machine learning" and "information retrieval" skills, based on skill co-occurrence patterns from all of our member base, we could infer that the member is likely to also know "learning-to-rank".

Since the dot-product results in a large number of non-zero scores of each member on the 35K skills, the scores are then thresholded. If a member's score on a skill is less than a threshold, the member is assumed not to know the skill and the score is zeroed out. Thus, outlier skills are removed. At the same time, the final expertise matrix (E_1) is still relatively denser than E_0 since it includes scores of inferred skills. We refer interested readers to our recent work [10] for more details.

Since matrix factorization is computationally complex, to guarantee efficiency, we apply a two-phase approach. An offline process periodically runs on distributed computing platforms like Hadoop to infer member skills. The online phase then simply consumes the latest version of the data at ranking time. Given a set of skills that a searcher has and a set of skills that a job requires, we compute Jaccard similarity between the the two sets. One future direction is to use weighted Jaccard similarity in which the weights are determined by searcher's expertise scores on the skills.

5. ENTITY-FACETED HISTORICAL CLICK-THROUGH-RATE

In this section, we propose entity-faceted document-popularity features based on user past behaviors. These features aim to indicate the quality of documents and are independent of query and searcher. The idea of using user's historical actions on documents in Web search ranking functions has been shown effective in the literature [3]. However, a key challenge when deploying these features is the sparseness of data. The challenge is even more serious for job search since the lifetime of a job document is much shorter. To overcome this, instead of computing a single historical CTR separately for every job in the corpus, we estimate historical CTRs of a job on different entity-facets, such as, title, company or location. Please also note that in this paper we use historical CTR as an example, the idea is also applicable to other user-action based popularity features where data sparseness is an issue.

5.1 Feature Definition

Given a job corpus, we compute multiple historical CTRs on different facets of the jobs. These facets are defined based on standardized entities mentioned in the jobs, such as, ti-

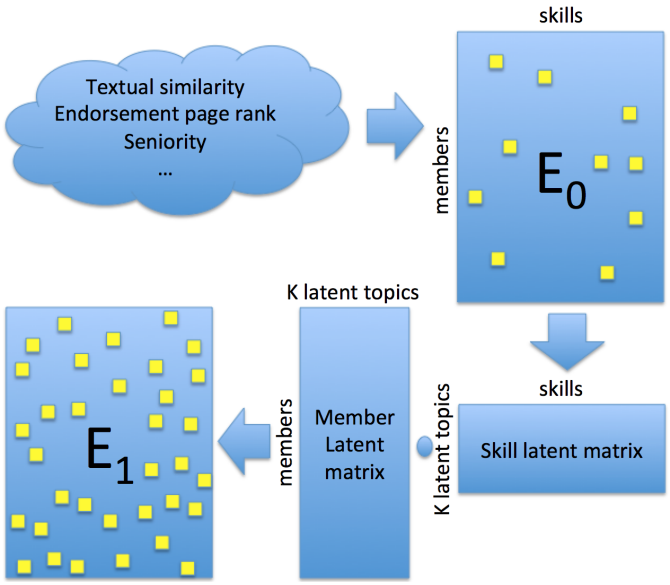

Figure 3: Member-skill matrix factorization to infer skills that members might have but do not explicit list on profiles.

tle, company or location. Specifically, we estimate historical CTR of jobs with a certain title like "data scientist", historical CTR of jobs at a company or historical CTR of jobs at some location. Then, given a specific job, for instance, a data scientist job at company LinkedIn in the city of Mountain View (California - USA), we use applicable entity-faceted historical CTRs including $CTR_{title}(data\ scientist)$, $CTR_{company}(LinkedIn)$ and $CTR_{location}(Mountain\ View\ (California\ -\ USA))$ as features to indicate the popularity of the jobs on the corresponding aspects.

It is worth noting that when computing historical CTRs, the jobs are grouped at entity level (e.g., company Ids), not at text level (e.g., company names in job postings). Thus jobs listing different names of the same company such as "International Business Machines", "IBM Corporation", "IBM corp." or "IBM" are grouped together in the same bucket. Similarly, jobs with different title variations of the same title entity like "software engineer", "software developer", "software development engineer" or "SDE" are also grouped together. Grouping the jobs at entity level does not only make the facet more meaningful, but also further reduces the issue of data sparseness. One future direction is to consider related entities, such as subsidiary companies (e.g., "Google", "Youtube" and "Alphabet") or neighboring cities (like "Mountain View" and "Sunnyvale" - both are in Silicon Valley) when estimating historical CTRs.

The entity-faceted historical CTRs are computed based on the number of times the jobs belonging to the entity got clicked and the number of times these jobs were shown on search results (See Equation 1). Smoothing factor N and default value λ are parameters. For entity that the jobs belonging to it have zero or a small number of impressions compared to N, historical CTR corresponding to it will have value equal or closed to the default value λ. Once again, for efficiency purpose, we build an offline process running on a distributed system to estimate the entity-faceted historical CTRs at a large scale. At ranking time, the online process takes the latest data to compute features in realtime.

$$CTR = \frac{(\#clicks + \lambda * N)}{(\#impressions + N)} \quad (1)$$

5.2 Parameter Tuning

The default value λ controls an interesting trade off for the cases where the number of impressions are low, e.g., jobs from a new company. If we use a too small default value, the new jobs that were not previously shown have little chance to be shown. Thus, we could not *explore* these results. However, many of these jobs could be good results and could get clicked if they are shown to the users. On the other hand, if we assign λ to a too high value, historical CTRs become too smooth and we do not *exploit* much of the insight from user historical actions.

$$\lambda = \underset{\lambda}{\text{argmax}} \quad Correlation(CTR, label) \quad (2)$$

We tune the values of λ on validation set. Each instance in the data set, which corresponds to a search result, contains a feature vector and a graded relevance label. The label indicates how relevant the result is to the query and the searcher. The details on how the data set is generated will be described in Section 6.2. For each entity-faceted historical CTR (e.g., company-faceted historical CTR), we select the value for λ that maximizes the correlation between the CTR and the labels (See Equation 2).

To illustrate the effect of parameter tuning, Figures 4 and 5 show company-faceted historical CTRs with λ equaling to zero and the optimal value. In each figure, the upper plot shows the relationship between the CTR feature and labels and the lower plot reveals company distribution over CTR values. On Figure 4, when the default value is zero, there are a significant number of companies having the historical CTR of zero because they have zero or few impressions in the period before we collect the training data. However, the average label of these results is not low (the first data point in Figure 4(a)), i.e., many of them are actually good results. Thus, as shown in the figure, the CTR does not strongly predict result relevance. When we assign λ to the optimal value (Figure 5), however, the company-faceted historical CTR becomes strongly correlated with the label. Quantitatively, between the two values for λ, the later improves Pearson correlation coefficient between the feature and the label by 55%.

6. MODEL TRAINING

This section describes how the proposed features are integrated into the job search ranking function. Specifically, we apply learning-to-rank approach to learn a new ranking function combining the entity-aware features with the existing ones. We first give a high level description on the existing features. Then, we present an approach for extracting personalized training data from search log and the learning-to-rank algorithm used to learn a new ranking function from the training data.

6.1 Existing Features

The existing features are generally divided into the following categories.

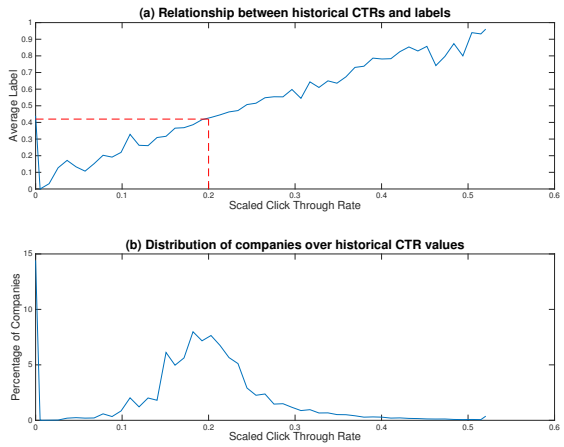

Figure 4: Company-Faceted Historical CTR with $\lambda = 0$. With this setting, the historical CTR does not predict result relevance well.

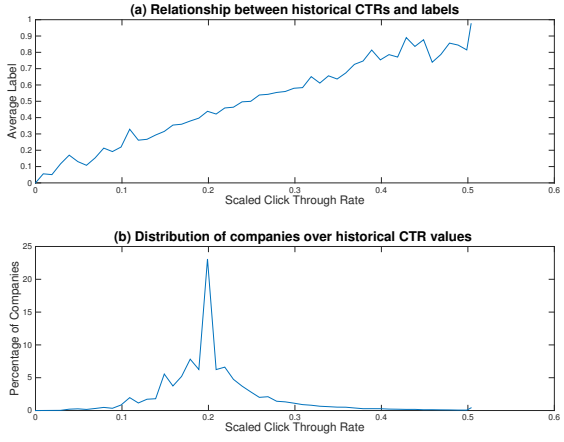

Figure 5: Company-Faceted Historical CTR with the optimal value for λ. With this setting, the historical CTR is strongly correlated with labels.

Textual features The most traditional type of features in information retrieval is textual features. These features match the keywords in queries with different sections of job descriptions, such as, title, company, etc. The key difference between these features and entity-aware matching features in Section 3 is that the former does not take into account standardized entity information when matching queries and documents.

Geographic features (personalized features) Job search on LinkedIn is highly personalized. For instance, a query like "software developer" from a job seeker will produce very different results if the searcher is in New York City, USA as opposed to (say) Montreal, Canada. Location plays an important role in personalizing the results. We create multiple features capturing this.

Social features (personalized features) Another important aspect of personalization is to capture how the results socially relate to the searcher. We leverage a variety of the signals on LinkedIn, such as, how the searcher socially connects with the company posting the job, e.g., if he or she follows the company or he or she has friends working at the company, etc. to generate the features in this category. We refer the readers to [10] for a more detailed description of the existing features.

6.2 Training Data and Learning Algorithm

A traditional way to obtain training data is to use human experts to label the results. However, as we are dealing with a large amount training data for personalized search, it is expensive to use human experts. At the same time, it is very hard for people other than the searcher to know the true relevance judgment of the results. For example, for the same query "software engineer", a new college graduate in the US and an experienced candidate in Canada could be interested in very different results. Thus, similar to [13], we use log data as implicit feedback from searchers to generate training data.

One problem with the log data is position bias as the users tend to interact (e.g., click) on top results. Thus, labels inferred from the user actions are biased towards the ranking function generating the data. In order to counter the position bias, we randomize the ranking of search results and show them to a small percentage of traffic. Then, we collect user actions on the results. For instance, as shown in Figure 6, a user issues query "software engineer", and six results are shown. There are different actions the user can take on the results and the corresponding ones will have different graded relevance labels based on the importance of the actions. In the example shown, the searcher clicks on the second result and decides to apply to the job at the fourth position. Since applying to a job is a stronger signal of relevance than clicking on a job, we assign a higher label to applied results (label=3, i.e., considered as perfect results) and a lower label to clicked results (label=1, i.e., good results). For the first and third results, the searcher scans through them but chooses not to take any action. Thus, we consider that these results are irrelevant (label=0, i.e., bad results). For the results ranked below the last interacted one (below result 4 in the Figure), we cannot be certain about the relevance judgement of them since the searcher may not have looked at them. Therefore, we discard these results.

It is worth noting that when collecting the data, we count every query issued as a unique search. For instance, if the

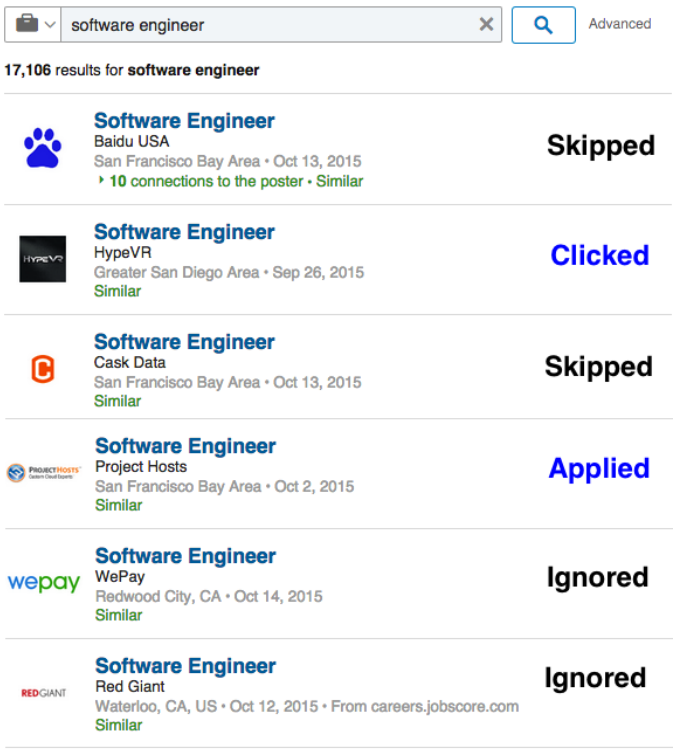

Figure 6: Example of how the labels are generated from user actions. Applied results are considered as perfect, clicked results are good and skipped results are bad. The results under the last interacted ones are ignored since the searcher may not have looked at them.

query "software engineer" is issued five times, they are considered as five distinct searches. The reason for this is that even though the query being searched for is identical across the five searches, they might have different search contexts when considering other dimensions such as searcher, location or time, etc. Moreover, this keeps the query distribution in the labeled data the same as in our live search traffic. We run the randomization bucket as mentioned above for two full weeks. The reason we keep the period in order of weeks is to iron out strong weekly patterns. For example, if the labeled data is collected only during weekdays, it will be biased towards weekday search patterns and will not represent weekend search traffic. After two weeks, the randomization bucket ends up with about two hundred thousand searches with at least one searcher's action. The labeled data is then divided into training, validation and test sets.

Given the training dataset, we apply Coordinate Ascent [18], a listwise learning-to-rank algorithm, to search for an optimal model. For efficiency purpose, we use linear models in our work. Coordinate Ascent algorithm has also been shown to be effective for learning linear ranking functions in some other search domains [18]. One key benefit of listwise learning-to-rank approach over pointwise and pairwise ones is that the listwise approach can optimize ranking-based metrics directly [15, 16]. The objective function we optimize in the learning process is normalized discounted accumulative gain (NDCG@K) defined on the graded relevance labels as described above. Parameter K is set to *25*, which is the number of results shown in the first result page of LinkedIn job search.

7. EVALUATION

In this section, we evaluate the proposed features in both offline testing and online A/B testing.

7.1 Offline Experiment

7.1.1 Models to Compare

Baseline: The baseline model uses all of the existing features described in Section 6.1 and does not contain any entity-aware features.

Entity-Aware Document-Query Matching (DQM): This model uses the existing features and entity-aware document-query matching features discussed in Section 3.

Document-Searcher Expertise Homophily (DSH): This model uses the existing features and searcher-document expertise homophily features described in Section 4.

Entity-Faceted Historical CTR (ECTR): This model uses the existing features and entity-facted histotical CTRs described in Section 5. In this paper, we experiment with company and title facets.

All of Entity-Aware Features (ALL): This model uses the existing features and all of the entity-aware features.

All of the models are trained on the same training set and are tuned extensively on the validation set to get the optimal parameter setting for each of them.

7.1.2 Offline Results

The Figure 7 shows performance lifts of the proposed models over the baseline on the test set in terms of Precision at 1 (P@1), Mean Reciprocal Rank (MRR), NDCG@15 and NDCG@25. As shown in the figure, all of the models with entity-aware features (DQM, DSH and ECTR) are consistently better than the baseline across all of the metrics. This confirms the benefit of using entity-aware features in job search. For instance, on NDCG@25, entity-aware document-query matching features, document-searcher expertise homophily features and entity-faceted historical CTR features can improve the baseline by 1.2%, 5.4% and 4.8% respectively.

Interestingly, amongst the three feature categories, document-query matching features yield the least improvement and document-searcher matching features achieve the highest improvement. This emphasizes the importance of using standardized entities to personalize search results. Finally, when combining all of the features together, not so surprisingly ALL model has the best performance. Compared to the baseline, ALL model is 20%, 12.1%, 8.3% and 7.9% better in terms of P@1, MRR, NDCG@15 and NDCG@25.

7.2 Online Experiment

In our online experiment, we take the model with the best offline performance, which is the one using all of the entity-aware features and compare it with the model currently in production. The model in production uses the features in Section 6.1 and some simple entity-aware document-query matching features. The later could be viewed as a simplified version of the features described in Section 3. The model

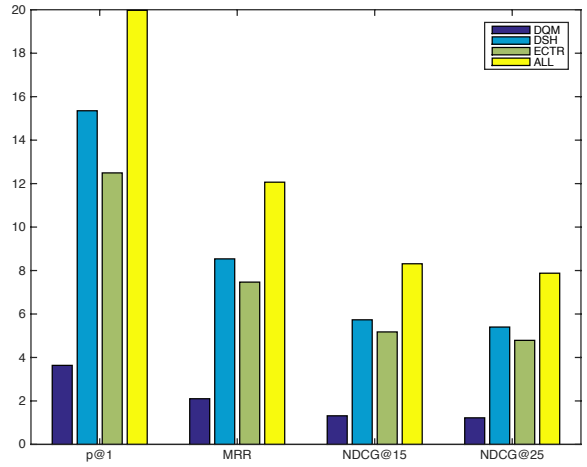

Figure 7: Percentage improvements over the baseline model on offline experiment. All of the improvements are statistically significant.

is also linear and trained by the same approach. Thus, the difference between the new model and the current one essentially demonstrates the effectiveness of the proposed features.

We take a random portion of LinkedIn job search live traffic and randomly split the searchers in the experiment traffic into control and treatment buckets. To guarantee consistent experience, each of the searchers is either control or treatment bucket (not both) throughout the whole experiment period. The control bucket is served by the current ranking model and the treatment bucket is served by the new model. We run the A/B test on LinkedIn experiment platform [26] for a reasonably long period of time (4 weeks) and ignore results in the first week to eliminate the novelty effect. During the three-week period, each bucket ends up with more than one million unique searchers and tens of millions of searches.

We evaluate the buckets by two metrics: click-through-rate and apply rate as defined in Section 4. These are the most important metrics from our business perspective. It is worth emphasizing that these are ranking-based metrics, not set-based metrics. In *online* experiments, even if two ranking functions produce exactly the same set of results on first result page (top 25 results), the one that ranks relevant results at higher positions on the page will be likely to achieve higher numbers of job views and applications. Thus, these metrics discount document relevance by positions, just like the NDCG metric that we use as the objective function in the offline training. In fact, we find that the NDCG metric in offline experiments is directionally inline with these metrics in online experiments, i.e., if a model A is better than a model B in terms of the NDCG metric in offline experiments, model A is likely to perform better than model B on click-through-rate and apply rate in online experiments.

Table 2 shows online A/B test results. Because of business sensitivity, we only report the relative improvements. On click-through-rate and apply rate, the new model with the proposed entity-aware features is 11.3% and 5.3% better than the current ranking model in production. Both of the improvements are statistically significant (details of the statistical testing is described in [26]) This re-confirms the advantage of entity-aware features in personalized job search ranking.

Metric	Improvement of Treatment over Control
Click-Through-Rate	11.3%
Apply Rate	5.3%

Table 2: Online A/B test results on click-through-rate and apply rate metrics. Both of the improvements are statistically significant.

8. CONCLUSION

In this paper, we propose an approach to applying standardized entity data to improve job search quality and to personalize search results. To match queries and job documents, we structure the queries and the documents in a schema specific to job search domain including critical entity types in the domain such as job title, skill, company and location, etc. Then, we construct various features semantically matching the queries and the job documents based on the structured schema. To close the gap between a query and the searcher's information need and also to personalize search results, we propose a concept of expertise homphily capturing the similarity between the searcher's expertise and the requirement of the job. We use standardized skills to represent searcher's expertise and job expertise requirement and measure the similarity based on these skills. Finally, to model document popularity, we introduce an idea of entity-faceted historical CTRs. These capture quality of each job on different aspects such as job title, company and location, etc. Grouping job documents based on the entities does not only make the groups meaningful but also alleviates the data sparseness issue when estimating historical CTRs.

Throughout the paper, we also detail how we resolve practical challenges when deploying standardized entities in job search ranking, including: ambiguity, missing values (such as members' skills), scalability and online efficiency (especially for sophisticated signals such as skill expertise scores), sparseness of historical searchers' actions (particularly in job search domain where job lifetime is typically short), parameter tuning for feature engineering, confounding factors and biases in log data and how to optimally combine the new features with the current ones in the existing ranking function.

We conduct offline experiment on randomized data and online A/B test to understand the effectiveness of the entity-aware features. In offline experiment, a new model with the features improves over the baseline +20%, +12.1% and +8.3% in terms of Precision@1, MRR and NDCG@25, respectively. Online A/B test shows that the features can improve click-through-rate and apply rate, the two most important metrics of the product, +11.3% and +5.3%, respectively. The results confirm the benefits of the proposed features in job search. As of this writing, the new model serves all live traffic on LinkedIn job search.

ACKNOWLEDGMENT: We would like to thank Deepak Agarwal for his valuable feedback during the course of this work.

9. REFERENCES

[1] D. Arya, V. Ha-Thuc, and S. Sinha. Personalized federated search at linkedin. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*, pages 1699–1702, 2015.

[2] D. Buscaldi and P. Rosso. Using geowordnet for geographical information retrieval. In *Evaluating Systems for Multilingual and Multimodal Information Access*, pages 863–866, 2008.

[3] O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview. In *Proceedings of the Yahoo Learning to Rank Challenge, held at ICML*, pages 1–24, 2011.

[4] O. Corby, R. Dieng-Kuntz, and C. Faron-Zucker. Querying the semantic web with corese search engine. In *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 705–709, 2005.

[5] J. Dalton, L. Dietz, and J. Allan. Entity query feature expansion using knowledge base links. In *The 37th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 365–374, 2014.

[6] B. Fazzinga and T. Lukasiewicz. Semantic search on the web. *Semantic Web*, 1(1-2):89–96, 2010.

[7] M. Fernandez, V. Lopez, M. Sabou, V. Uren, D. Vallet, E. Motta, and P. Castells. Semantic search meets the web. In *2008 IEEE International Conference on Semantic Computing (ICSC)*, pages 253–260, 2008.

[8] T. W. Finin, L. Ding, R. Pan, A. Joshi, P. Kolari, A. Java, and Y. Peng. Swoogle: Searching for knowledge on the semantic web. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, pages 1682–1683, 2005.

[9] R. Guha, R. McCool, and E. Miller. Semantic search. In *Proceedings of the 12th ACM International Conference on World Wide Web (WWW)*, pages 700–709, 2003.

[10] V. Ha-Thuc, G. Venkataraman, M. Rodriguez, S. Sinha, S. Sundaram, and L. Guo. Personalized expertise search at linkedin. In *Proceedings of the 4th IEEE International Conference Big Data*, 2015.

[11] V. Ha-Thuc, Y. Xu, S. P. Kanduri, X. Wu, V. Dialani, Y. Yan, A. Gupta, and S. Sinha. Search by ideal candidates: Next generation of talent search at linkedin. In *Proceedings of the 25th International Conference on World Wide Web (WWW)*, pages 195–198, 2016.

[12] J. Heflin, J. A. Hendler, and S. Luke. SHOE: A blueprint for the semantic web. In *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*, pages 29–63, 2003.

[13] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142, 2002.

[14] P. F. Lazarsfeld and R. K. Merton. Friendship as a social process: A substantive and methodological analysis. *Freedom and control in modern society*, 18:18–66, 1954.

[15] H. Li. A short introduction to learning to rank. *IEICE Transactions*, 94-D(10):1854–1862, 2011.

[16] T. Liu, T. Joachims, H. Li, and C. Zhai. Introduction to special issue on learning to rank for information retrieval. *Information Retrieval*, 13(3):197–200, 2010.

[17] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[18] D. Metzler and W. B. Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274, 2007.

[19] E. Prud'Hommeaux, A. Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.

[20] X. Shen, B. Tan, and C. Zhai. Implicit user modeling for personalized search. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 824–831, 2005.

[21] D. Sontag, K. Collins-Thompson, P. N. Bennett, R. W. White, S. Dumais, and B. Billerbeck. Probabilistic models for personalizing web search. In *Proceedings of the fifth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 433–442, 2012.

[22] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th ACM International Conference on World Wide Web (WWW)*, pages 675–684, 2004.

[23] B. Tan and F. Peng. Unsupervised query segmentation using generative language models and wikipedia. In *Proceedings of the 17th ACM International Conference on World Wide Web (WWW)*, pages 347–356, 2008.

[24] V. S. Uren, Y. Lei, and E. Motta. Semsearch: Refining semantic search. In *The Semantic Web: Research and Applications, 5th European Semantic Web Conference*, pages 874–878, 2008.

[25] R. W. White, P. N. Bennett, and S. T. Dumais. Predicting short-term interests using activity-based search context. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1009–1018, 2010.

[26] Y. Xu, N. Chen, A. Fernandez, O. Sinno, and A. Bhasin. From infrastructure to culture: A/B testing challenges in large scale social networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2227–2236, 2015.

[27] J. Yan, W. Chu, and R. W. White. Cohort modeling for enhanced personalized search. In *Proceedings of the 37th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 505–514, 2014.

[28] J. Zhong, H. Zhu, J. Li, and Y. Yu. Conceptual graph matching for semantic search. In *Conceptual Structures: Integration and Interfaces, 10th International Conference on Conceptual Structures*, pages 92–196, 2002.