Personalized Job Recommendation System at LinkedIn: Practical Challenges and Lessons Learned

Krishnaram Kenthapadi
LinkedIn Corporation, USA
kkenthapadi@linkedin.com

Benjamin Le
LinkedIn Corporation, USA
ble@linkedin.com

Ganesh Venkataraman
LinkedIn Corporation, USA
ghvenkataraman@linkedin.com

ACM Reference format:
Krishnaram Kenthapadi, Benjamin Le, and Ganesh Venkataraman. 2017. Personalized Job Recommendation System at LinkedIn: Practical Challenges and Lessons Learned. In *Proceedings of RecSys'17, August 27–31, 2017, Como, Italy.*, , 2 pages.
DOI: http://dx.doi.org/10.1145/3109859.3109921

1 PERSONALIZED JOB RECOMMENDATION: PRACTICAL CHALLENGES

Online professional social networks such as LinkedIn play a key role in helping job seekers find right career opportunities and job providers reach out to potential candidates. LinkedIn's job ecosystem has been designed to serve as a marketplace for efficient matching between potential candidates and job postings, and to provide tools to connect job seekers and job providers. LinkedIn's job recommendations product is a crucial mechanism to help achieve these goals, wherein personalized sets of recommended job postings are presented for members based on the structured, context data present in their profiles.

The problem of recommending jobs to LinkedIn members poses several unique information retrieval, system, and modeling challenges. (1) Personalized recommendations need to be computed in real-time by scoring millions of structured candidate job documents associated with each query while providing a high degree of data freshness and meeting strict latency requirements. Here, the query incorporates the member context / interests expressed through member profile, and hence can become very large, comprising of thousands of Boolean clauses over hundreds of document attributes. Consequently, candidate selection techniques need to be applied since it is infeasible to retrieve and score all matching jobs from the underlying inverted index. (2) Typically, the underlying retrieval systems may use content-based recommendation models, which are primarily based on the explicit member context / interests obtained from the profile, and may not take into account the implicit context in the form of user interactions. Hence, we may face the challenge of incorporating different types of user interaction signals as part of the relevance model. (3) The job recommendation problem is fundamentally different from traditional recommendation system problems such as recommending books, products, or movies to users. While all of the above have a common objective to maximize the engagement rate of the users, one key difference is that a job posting is typically meant to hire one or a few employees only, whereas the same book, product, or movie could be potentially recommended to hundreds of thousands of users for consumption.

RecSys'17, August 27–31, 2017, Como, Italy.

DOI: http://dx.doi.org/10.1145/3109859.3109921

As LinkedIn's job posters pay to post jobs on the site, on the one hand it is critical to deliver sufficient number of applications from qualified candidates to each job, such that the job poster can interview the top candidates and finally hire one or a few of them. On the other hand, it is not desirable either for the system to deliver too many applications to any posted jobs with one or a few openings, as the amount of effort for the job poster to interview would become much greater than expected. Also, if too many job seekers compete for the same job posting, each job seeker's chances of getting that job will be dramatically reduced.

In this talk, we will present how we formulated and addressed the above problems, the overall system design and architecture, the challenges encountered in practice, and the lessons learned from the production deployment of these systems at LinkedIn. By presenting our experiences of applying techniques at the intersection of recommender systems, information retrieval, machine learning, and statistical modeling in a large-scale industrial setting and highlighting the open problems, we hope to stimulate further research and collaborations with the RecSys community.

2 OVERVIEW OF SYSTEMS DEVELOPED AND DEPLOYED AT LINKEDIN

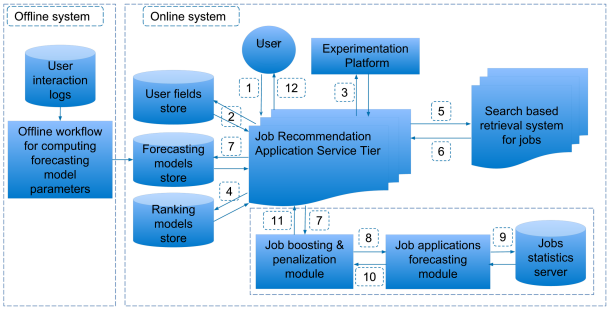

Figure 1: Architecture of LinkedIn's job recommendation engine.

The overall architecture of LinkedIn's job recommendation engine is shown in Figure 1. Our system can be subdivided into an online system for serving job recommendations and an offline workflow for updating different machine learned and statistical forecasting models (not fully shown).

We now provide a description of the overall flow for how a client request (query) is processed in our online job serving system. It makes use of a multi-tier service architecture, wherein the recommendation request queries are handled in a distributed fashion across hundreds of production servers. The online query processing proceeds as follows:

- Whenever a job recommendation needs to be shown to a user, the client application issues a query containing the

userId and other metadata to the backend distributed job recommendation application service tier (step 1).
- The job recommendation application service then retrieves relevant structured user data from the user fields store (step 2), determines the appropriate A/B testing experimental treatment that the user falls under (step 3), and fetches the necessary ranking models based on the experimental treatment (step 4). Then, it creates a request object containing the user data and the models, which is issued to the distributed search service tier (step 5).
- Our search based retrieval system is built on top of LinkedIn's Galene search platform [5]. This system is responsible for applying the candidate selection model [1] [4] to the user data to generate a corresponding Galene query, issuing this query to the distributed search index service, scoring the resulting jobs using the final-pass GLMix ranking model [7], and returning back the top ranked job recommendation results (step 6).
- The job recommendation application service then performs post-processing (such as applying filters and certain business rules), and in particular, invokes the job boosting and penalization module [2] (steps 7–11).

We will focus on three sub-systems in this talk:

Candidate Selection: Inspired by our prior work in *CaSMoS* [1] , we present a new machine learning framework to construct effective queries as part of candidate selection in personalized search and recommendation systems via decision trees [4]. Query construction is done by converting the branch paths in the decision tree to corresponding Weighted AND (WAND) clauses [3]. The deployment of the prior candidate selection system resulted in up to 25% reduction in latency without sacrificing recommendation quality. Iterating on top of this system by introducing decision trees and early termination with static rank further reduced p99 latency by 56%. These system efficiency improvements paved the way for usage of more sophisticated scoring models, including GLMix.

Personalized Relevance Models: *Dionysius* was our first work for incorporating user interactions to personalize job recommendations in a system originally built as a content-based recommender [6]. Designed to leverage existing recommendation infrastructure, we learned the hidden fields for each user by considering the hierarchy of interaction signals and replaced the user profile-based fields with them to use in our ranking model. Through improved modeling and ranking infrastructure, we have replaced *Dionysius* with large-scale generalized linear mixed models (GLMix) containing millions of parameters [7]. In order to capture user's personal preferences on items and the item's specific attraction, we introduce member and job level regression coefficients in addition to the global regression coefficients. The global regression coefficients act as our prior prediction given a member profile and job posting. The posterior is estimated through fitting the per-member and per-job regression coefficients, which becomes more effective as a member interacts with more jobs and a job receives more attention. The end result is tailored job recommendations for a member given their profile, activity, and activity of similar members. Deploying the GLMix model increased job applications by 20% to 40% per day.

LiJAR: *LiJAR* is LinkedIn's job applications forecasting and redistribution system, designed with the goal of ensuring that job postings do not receive too many or too few applications, while providing job recommendations to users with *the same level* of relevance as the system that purely optimizes for user engagement [2]. This system uses a dynamic forecasting model to estimate the expected number of applications at the job expiration date, and algorithms to either promote or penalize jobs based on the output of the forecasting model in real time. Our online A/B testing experiments demonstrate the effectiveness of our approach in increasing the engagement of users for underserved jobs by 6.5%, while keeping the user engagement in terms of the total number of job applications the same.

3 AUTHOR BIOS

Krishnaram Kenthapadi is a Staff Software Engineer and Applied Researcher at LinkedIn. He shaped the technical roadmap and led the privacy/modeling efforts for LinkedIn Salary product, and prior to that, served as the relevance lead for the LinkedIn Careers and Talent Solutions Relevance team, which powers search/recommendation products at the intersection of members, recruiters, and career opportunities. Previously, he was a Researcher at Microsoft Research Silicon Valley. He received his PhD degree in Computer Science from Stanford University in 2006. He received the SODA best student paper award and the WWW best paper award nomination. He has published 35+ papers, with 2500+ citations and filed 110+ patents.

Benjamin Le is a Senior Software Engineer at LinkedIn. His contributions to jobs relevance include candidate selection in job search through multi-pass ranking and deploying generalized linear mixed models for reducing poor job recommendation impressions. Currently, he is focusing on the deployment of deep and wide models for job recommendations. He is a recent graduate of the University of California, Berkeley, where he received both his Master of Engineering and Bachelor of Science Degrees.

Ganesh Venkataraman currently leads all AI powering jobs relevance at LinkedIn. His contributions at LinkedIn include, leading a multi group effort that led to 50% improvement in job applications at LinkedIn, leading end to end re-architecture of job search, machine learned ranking for people search typeahead (system that allows members to search for 400MM+ users via instant results), introducing machine learned ranking towards skills search at LinkedIn (ex: searching for people skilled at "information retrieval"). He co-authored a paper on personalized ranking which won the best paper award at the IEEE Big Data Conference 2015. He holds a Ph.D. from Texas A&M in Electrical & Computer Engineering where he was the recipient of the Dean's graduate merit scholarship. He has several publications with 200+ citations including one fundamental contribution to graph theory.

REFERENCES

[1] F. Borisyuk, K. Kenthapadi, D. Stein, and B. Zhao. CaSMoS: A framework for learning candidate selection models over structured queries and documents. In *KDD*, 2016.
[2] F. Borisyuk, L. Zhang, and K. Kenthapadi. LiJAR: A system for job application redistribution towards efficient career marketplace. In *KDD*, 2017.
[3] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Zien. Efficient query evaluation using a two-level retrieval process. In *CIKM*, 2003.
[4] A. Grover, D. Arya, and G. Venkataraman. Latency reduction via decision tree based query construction. LinkedIn Technical Report, 2017.
[5] S. Sriram and A. Makhani. LinkedIn's Galene Search engine, 2014. http://engineering.linkedin.com/search/did-you-mean-galene.
[6] J. Wang, K. Kenthapadi, K. Rangadurai, and D. Hardtke. Dionysius: A framework for modeling hierarchical user interactions in recommender systems. LinkedIn Technical Report, available at https://arxiv.org/abs/1706.03849, 2016.
[7] X. Zhang, Y. Zhou, Y. Ma, B.-C. Chen, L. Zhang, and D. Agarwal. GLMix: Generalized linear mixed models for large-scale response prediction. In *KDD*, 2016.