Generalized Mixed Effect Models for Personalizing Job Search

Ankan Saha
Linkedin Corporation
asaha@linkedin.com

Dhruv Arya
Linkedin Corporation
darya@linkedin.com

ABSTRACT

Job Search is a core product at LinkedIn which makes it essential to generate highly relevant search results when a user searches for jobs on Linkedin. Historically job results were ranked using linear models consisting of a combination of user, job and query features. This paper talks about a new generalized mixed effect models introduced in the context of ranking candidate job results for a job search performed on LinkedIn. We build a per-query model which is populated with coefficients corresponding to job-features in addition to the existing global model features. We describe the details of the new method along with the challenges faced in launching such a model into production and making it efficient at a very large scale. Our experiments show improvement over previous baseline ranking models, in terms of offline metrics (both AUC and NDCG@K metrics) as well as online metrics in production (Job Applies) which are of interest to us. The resulting method is more powerful and has also been adopted in other applications at LinkedIn successfully.

CCS CONCEPTS

•**Statistical Machine Learning** →**Generalized Linear Models;** •**Information Retrieval and Data Mining** →*Web Search and Ranking;*

KEYWORDS

Generalized Linear Mixed Effect Models; Information Retrieval and Search Ranking

ACM Reference format:
Ankan Saha and Dhruv Arya. 2017. Generalized Mixed Effect Models for Personalizing Job Search. In *Proceedings of SIGIR '17, Shinjuku, Tokyo, Japan, August 07-11, 2017,* 4 pages.
DOI: http://dx.doi.org/10.1145/3077136.3080739

1 INTRODUCTION

LinkedIn has been a continuously developing and evolving ecosystem over the last decade, now encompassing more than 467 million members [6] and 7M+ active jobs. Job search, in particular, has been one of the principle product offerings at LinkedIn. With an unprecedented increase in the number of sources and volumes of job opportunities available via Linkedin, the problem of serving the most appropriate and relevant jobs with minimal delay becomes significantly challenging. Given a search query provided by a user, we try to offer the most relevant and personalized ranking of jobs that best fit the combination of the user and the query. LinkedIn's importance as one of the premier global professional networks makes it essential to generate results that are highly relevant and capture the most appropriate interests of the job-seeker. One of the standard examples where personalization makes a huge difference is when a user searches for a query like *software engineer*. Depending on the skills, background, experience, location and other factors available to LinkedIn, the ranked list of results can be drastically different. Thus a person skilled in *machine learning* would see a very different set of jobs compared to someone specializing in *hardware* or *computer networks*.

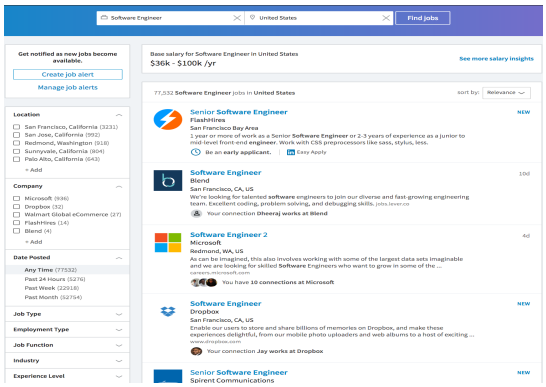

Figure 1: Example of a job search at LinkedIn

In order to further expand personalization beyond initial job-search approaches at LinkedIn [2, 3, 7], we introduce a new kind of model in the job search ecosystem called Generalized Linear Mixed effect modeling, GLMix [9]. We introduce a per-query model which is trained on the space of job-features in addition to the global model. As a result, we are able to capture finer signals in the training data thus allowing us to better differentiate how the presence of a job skill like *Hadoop* should generate ranking results for a query like *software engineer* as opposed to a skill like *IPV6*.

Introducing the per-query coefficients on top of the global features gives us significant improvement in offline metrics including ROC-AUC on the test dataset and offline NDCG, which is the key metric of interest for ranking applications. We launched an A/B test to compare this model over the previously existing model with global features and got healthy improvements in job application rates and total job applies which are the foremost metrics in job-search.

SIGIR '17, Shinjuku, Tokyo, Japan

978-1-4503-5022-8/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3077136.3080739

2 PROBLEM OVERVIEW

Job search is a very pertinent activity for a job seeker. A job seeker visits LinkedIn and enters a query expressing an explicit intent to find jobs that match that query. The search engine provides a set of results best matching to the query and personalized to the user if she is logged on Linkedin. For a job seeker the next step requires reading through the job descriptions, understanding the job responsibilities, skills required and learning about the company that has posted the job.

Historically our algorithms would heavily utilize text and entity based features extracted from the query and jobs to come up with a global ranking [3]. However learning such a global ranking model would improve certain queries and degrade others dependent on the representation of the queries and jobs in the training data. As an example let's consider a popular query, namely *software engineer*. Given the number of job seekers issuing this query we would always want to ensure that we maintain high relevance even if our trained global ranking model cannot generalize well for this query.

3 MIXED EFFECT MODELS

Generalized Linear Mixed Effect Models (GLMix) has been successfully used on large scale machine learning applications [9] to build per-user models in the past. In this section, we describe the GLMix models and how they apply in the context of job-search.

In the context of job-search application, the key is to show the *best* jobs to the user based on his query according to some measure. We quantify this measure in the traditional way as the likelihood of the member m to apply for the job j if it is impressed upon her when she enters the query q, measured by the binary response y_{mjq}. $\mathbf{s}_j$ denotes the feature vector of job j, which includes features extracted from the job post, e.g. the job title, summary, location, desired skills and experiences while $\mathbf{x}_{mjq}$ represents the overall feature vector for the (m, j, q) triple, which can include member, job, query and associated context features and any combination thereof.

3.1 Motivation behind a mixed-effect model

The previous baseline job-search model ranked jobs based on optimization of a Learning To Rank (LTR) metric [1, 4, 8] using a set of global features similar to $\mathbf{x}_{mjq}$ using coordinate ascent. However these features do not capture relationships between individual queries and jobs. Consider the following scenario: A member on LinkedIn searches for jobs with the query "software engineer". Suppose skill required for the job is one of the job features in $\mathbf{s}_j$. Now let us consider two jobs, Job1, which has skill features corresponding to "java" and "hadoop" and Job2 which has skill features corresponding to "capacitors" and "hardware design". Clearly the nature of the query should attract Job1 at a higher rank than Job2. This can be realized by obtaining a notion of affinity of the query string with the job-features associated with the job but this kind of signal cannot be exploited directly by using the model based solely on global features. Although this purpose can be achieved by introducing interaction features between each query string and job feature, that would make the feature space prohibitively expensive and training a model very difficult. Instead this can be achieved by introducing a new class of models called the mixed-effect models which can exploit the interaction of each query with the job features explicitly.

3.2 Generalized Linear Mixed Effect Models for Job Search

The GLMix model that we adopt for predicting the probability of member m applying for job j based on query q using logistic regression is

$$g(\mathbb{E}[y_{mjq}]) = \mathbf{x}'_{mjq}\mathbf{b} + \mathbf{s}'_j\boldsymbol{\beta}_q \tag{1}$$

where $g(\mathbb{E}[y_{mjq}]) = \log \frac{\mathbb{E}[y_{mjq}]}{1-\mathbb{E}[y_{mjq}]}$ is the link function, $\mathbf{b}$ is the global coefficient vector (also called fixed effect coefficients in the statistical literature) and $\boldsymbol{\beta}_q$ are the coefficient vectors specific to query q, called random effects coefficients, which capture query q's association or relationship with different job features.

Note that it is also possible to have similar random effects coefficients $\boldsymbol{\alpha}_m$ or $\boldsymbol{\gamma}_j$ on a per-member or per-job basis which can then be combined with features on the job-query or member-query spaces respectively. However given that the number of members or jobs in the LinkedIn ecosystem is of the order of millions (compared to possibly tens of thousands of popular queries), this can make such a model prohibitively expensive to be applied in production, as the final model would have a different set of coefficients for each member and each job and would incur severe latency while generating the scores for a (m, j, q) triple at production time. We notice that applying the random effects via a per-query model on the job-features in conjunction with the global model already allows us to improve upon the baseline model significantly in terms of both offline metrics as well as application rates in production. The member features tend to be static and do not contribute much additionally on being added into the per-query model.

The corresponding model in equation (1) is optimized via alternating optimization using parallelized coordinate descent. We alternately optimize for the global features and the per-query features for each query while holding all other variables fixed. For more details about the optimization algorithm, we refer the readers to [9] as well to the open source implementation of the algorithm called Photon-ML (link). It was demonstrated in this paper that GLMix has good scalability properties -scaling up almost linearly with the number of executors as long as the amount of data processed per executor remains consistent.

4 OFFLINE EXPERIMENTS

In this section, we describe the details of the experiments that were performed to compare the GLMix models with the previous linear feature based models for information retrieval trained using coordinate-ascent as part of the RankLib library [4].

4.1 Setup

The baseline models running RankLib were performed on the LinkedIn Hadoop cluster. All the experiments for GLMix models were implemented in Apache Spark. The experiments were conducted on a cluster consisting of 135 nodes managed by Apache YARN. Each node has 24 Intel Xeon(R) CPU E5-2640 processors with 6 cores at 2.50GHz each, and every node has 250GB memory. In the following experiments, Spark is running on top of YARN as a client, and the memory for each Spark executor is configured to 10GB, where each executor corresponds to a container (e.g., a core of the node) in YARN.

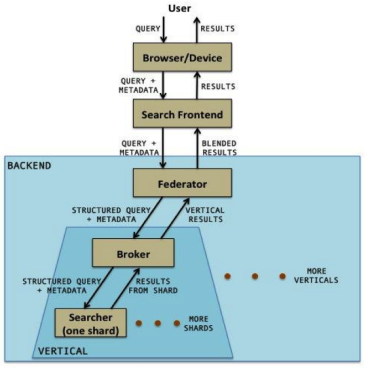

Figure 2: Job Search Scoring Infrastructure at LinkedIn

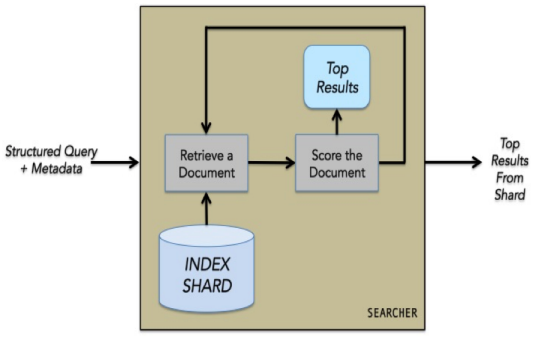

Figure 3: Producing results for a query

4.2 Data for offline experiments

Our offline experiments are conducted using a sample of six weeks of data spanning 09/16-10/16 of members' interactions on the *Job Search* module which is navigable from the LinkedIn homepage. We split the data by time into training (4 weeks), validation (1 week) and test (1 week). The positive responses consisted of impressions that resulted in applications clicks and impressions that did not result in application clicks provide the negative responses. Note that the detail job views are not considered as a signal in this experiment as we don't explicitly optimize for that metric. We subsample the negative examples so that the ratio of positive to negative examples is not too low.

There are around 10K job features for $\mathbf{s}_j$, e.g. the job title, key words in the description, required skills and qualifications, seniority, experience and functions. There are also about 100 global features consisting of various sort of interactions between member, job, query, location and industry features - including features like distance between member and job location, keyword match in job title and query, skill match between member and query among others. We compare the following three models:

- Baseline model with about 100 global features (**Baseline**).
- Baseline model + job position as a feature (**Model 2**)
- Model 2 + per-query x (job features **s**) (**Model 3**).
- Model 2 + per-query x (job features + position features) (**Model 4**).
- Model 2 + per-query x (member features + job features + global features + position features) (**Model 5**)

Since we wanted to build on top of the baseline model, we kept the baseline model consistent for all experiments and used the offset obtained from this model to train the random effect (per-query) part of the model. As shown in the results, we noticed that the addition of global features and member features does not significantly improve the model performance, while making the model much larger and more challenging to be pushed into production. As a result, we decided to proceed with only job features and position features in the per-query model. The position features were added to explain the presentation bias of jobs impressed at a higher position having a high CTR via their coefficients from the training model. During the scoring phase, these features are not available and therefore do not contribute to the final score for each candidate job while ranking.

Model	Offline AUC (Test Dataset)	Offline NDCG@25
Baseline	0.55	0.6261
Model 2	0.64	0.6309
Model 3	0.6904	0.6492
Model 4	0.7025	0.6493
Model 5	0.7049	0.6493

Table 1: Offline metrics for the different models

5 ONLINE SCORING TO GENERATE RANKING RESULTS

In this section, we provide details about how we deployed the model online in our job search system. The search system at LinkedIn is an in house developed system called **Galene** [5]. The search system utilizes Lucene to assist in building the index and retrieve matching entities from the index. The query begins at the browser/device and is passed down to our search backend after some preprocessing. Our search backend is a sharded system consisting of a single broker and multiple searchers (Figure 2). The role of the broker is to understand the user query, scoring metadata such as per query model features and coefficients, searcher data and build an annotated request for the searcher to execute. Specifically for the per-query model the broker fetches the model coefficients and hashed features (for compact storage and efficient matching during retrieval) from a key-value store with the key being a combination of the model name and the query. The annotated request thus built is then broadcasted to all searcher nodes. The searchers hold the sharded index over which the query is executed to get the matched results (Figure 3).

Searchers have a multipass scoring pipeline. A lightweight model is first used to narrow down the candidate jobs and then the global and per query model are applied to the set of candidates. The scorer at the searcher goes through the per-query coefficients and for each job adds the coefficient weights for the hashed features present. Each searcher takes a local view and computes the top k jobs as requested by the broker. These jobs are then passed back to the

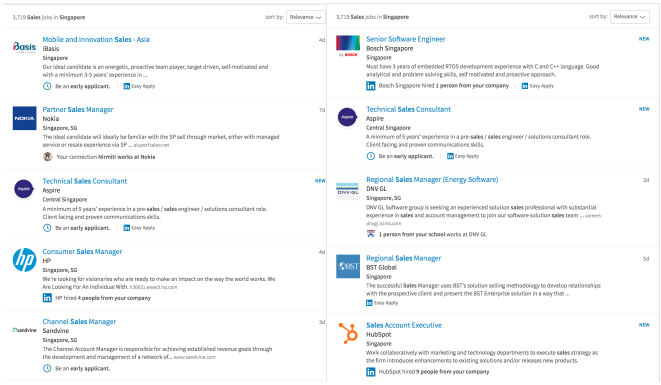

Figure 4: GLMix(left) vs Base Model results for query "Sales"

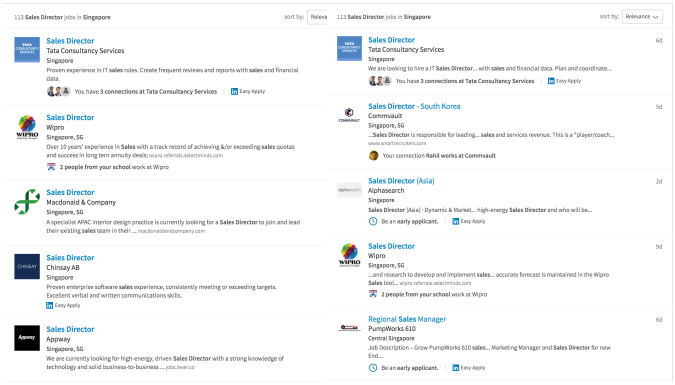

Figure 5: GLMix(left) vs Base Model results for query "Sales Director"

broker from all the searchers. The broker merges the set of returned jobs and may optionally apply a set of re-rankers to improve the global ranking. The final ranked jobs are then returned back to the frontend system which are then decorated and shown to the users.

The candidate selection helps narrow down the jobs to a few thousand which are then ranked by the per query model. All of this happens with extremely low latencies and at the current moment powers all of job search at LinkedIn.

6 ONLINE EXPERIMENTS

After launching the GLMix model (**Model 4**) and ensuring its healthy performance along with acceptable scoring latency, the ramp was finally increased to 30%. The corresponding results are illustrated in table 2. For comparison, we illustrate the results

Model	CTR@Position1	Total Job Apply Clicks
Model 4	+6.4%	+5.8%

Table 2: Lift over baseline model for online metrics

shown by the GLMix model vs the baseline model for the same user based in Singapore. Note that when we search for "Sales" (Figure 4), the GLMix model is more accurate (the topmost result of the baseline model is not very relevant) and also stays close to the skill set of the user, thus showing jobs mainly between "Sales Manager" and "Sales Consultant". On the other hand, when we are more specific and provide "Sales Director" as the query (Figure 5), the results are more similar although the GLMix model stays more accurate based on the skill-set of the user, which makes the model understand that (s)he is senior enough and would not want to see a job-posting about a sales manager (5th result for global model). Thus the GLMix model also helps us address potential mismatch in seniority/positions if the corresponding information is provided by the user to Linkedin.

7 CONCLUSION AND FUTURE WORK

This paper looks at the problem of ranking job search results given a query entered by the user. By using a new mixed effect model, we can obtain a greater level of personalization by developing a per-query model over the space of job features. Besides being better, our model can also be scaled to millions of examples and can be trained in a distributed manner efficiently. We demonstrate improvements in both offline metrics like AUC and NDCG as well as online metrics of interest via A/B tests.

We can potentially scale this model further to train per-member and per-job model to obtain further levels of personalization and capture the signal hidden in these aspects of the data as well. Another new line of work involves training latent representations for entities like jobs, members and queries in a supervised fashion and train them using deep learning. We are continuing to look into these areas to further improve the quality of job search at Linkedin.

REFERENCES

[1] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to Rank Using Gradient Descent. In *Proceedings of the 22nd International Conference on Machine Learning (ICML '05)*. ACM, New York, NY, USA, 89–96. DOI: http://dx.doi.org/10.1145/1102351.1102363

[2] Viet Ha-Thuc and Shakti Sinha. 2016. Learning to Rank Personalized Search Results in Professional Networks. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*. 461–462. DOI: http://dx.doi.org/10.1145/2911451.2927018

[3] Jia Li, Dhruv Arya, Viet Ha-Thuc, and Shakti Sinha. 2016. How to Get Them a Dream Job?: Entity-Aware Features for Personalized Job Search Ranking. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 501–510. DOI: http://dx.doi.org/10.1145/2939672.2939721

[4] Donald Metzler and W. Bruce Croft. 2007. Linear Feature-based Models for Information Retrieval. *Inf. Retr.* 10, 3 (June 2007), 257–274. DOI: http://dx.doi.org/10.1007/s10791-006-9019-z

[5] Sriram Shankar. 2014. Did you mean "Galene"? https://engineering.linkedin.com/search/did-you-mean-galene. (2014).

[6] LinkedIn Corporate Communications Team. 2016. LinkedIn Announces Third Quarter 2016 Results. https://press.linkedin.com/site-resources/news-releases/2016/linkedin-announces-third-quarter-2016-results. (2016).

[7] Ganesh Venkataraman, Abhimanyu Lad, Lin Guo, and Shakti Sinha. 2016. Fast, lenient and accurate: Building personalized instant search experience at LinkedIn. In *2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*. 1502–1511. DOI: http://dx.doi.org/10.1109/BigData.2016.7840758

[8] Qiang Wu, Christopher J. Burges, Krysta M. Svore, and Jianfeng Gao. 2010. Adapting Boosting for Information Retrieval Measures. *Inf. Retr.* 13, 3 (June 2010), 254–270. DOI: http://dx.doi.org/10.1007/s10791-009-9112-1

[9] XianXing Zhang, Yitong Zhou, Yiming Ma, Bee-Chung Chen, Liang Zhang, and Deepak Agarwal. 2016. GLMix: Generalized Linear Mixed Models For Large-Scale Response Prediction. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 363–372. DOI: http://dx.doi.org/10.1145/2939672.2939684