Model Threshold Optimization for Segmented Job-Jobseeker Recommendation System

Yichao Jin, Anirudh Alampally, Dheeraj Toshniwal, Zhiming Xu and Ankush Girdhar

Indeed.com
{jinyichao, aalampally, dtoshniwal, zxu, ankush}@indeed.com

Abstract

Recently, job-jobseeker recommendation system has played an important role in helping people get more timely and suitable jobs in the domain of HR technology. Most existing recommender systems proposed an unified model to serve all the job and jobseekers from different backgrounds. While very limited work, if not none, had paid attention to the possible performance gap among different segments. In this work, we use the occupation data to define the job segment, and study the segment-level performance comparison from an existing recommendation system within our organization. We then try to identify the possible causes, and make multiple attempts to deal with the problem. Finally, we adopt the most feasible approach to conduct the per-segment level model threshold optimization. In particular, we properly formulate a constrained optimization problem, and propose an efficient algorithm to speed up the threshold optimization process. Our prototype implementation enables the online A/B tests. The experimental results from real online products indicate significant performance improvement in terms of both recommendation quality and coverage on a list of selected segments.

Keywords

Job-jobseeker Recommendation, Segmentation, Threshold Optimization

1. Introduction

Nowadays, online job marketplaces such as Indeed.com, CareerBuilder and LinkedIn, are serving hundreds of millions jobseekers by connecting them to the right job opportunities. The target jobseekers of such recommender systems should not limit to any specific segments or groups. Instead, we should try to help all the jobseekers with a variety of profiles to get their best jobs in an efficient and scalable manner.

The job-jobseeker recommendation platform is one of the most important engines that we are using to help people get jobs within our organization. There are multiple ways that we recommend either jobs or jobseekers to the other side throughout different surfaces. Specifically, on the jobseeker-facing side, we sent invite-to-apply emails or app notifications to the jobseekers. We also display a list of recommended jobs on the homepage. On the employer-facing side, we provide instant candidate recommendations to the employers as soon as they publish a new job post.

Underneath the recommendation platform, we have multiple match providers, where each provider has its own way to retrieve and rank matches. In this paper, we will mainly focus on the ranking stage for our longest-lived probabilistic-based match provider using Logistic Regression models. Specifically, we have a set of Logistic Regression models to predict each steps along with the application funnel for every single job-jobseeker pair. In particular, we have three models in a tandem. The first model predicts the probability of receiving a response (either positive or negative) from the jobseeker , given the recommendation is made. The second one predicts the probability of getting a positive response (e.g., apply or enquiry), given receiving the jobseeker response. The third one predicts the probability of having a positive employer response (e.g., interview schedule or hire decision), given the application is made from the jobseeker. Each model has its own threshold to filter out certain matches with low scores, and the product of all the model scores will be used to rank the remaining matches.

Currently we only have one set of models for all types of jobs and jobseekers, while we found the performance gap is huge across different job segments in terms of their occupation. Although we already use a few segment-specific data (e.g., job title, industry, etc.) as the model features, the data didn't seem to be good enough to represent all the explicit or implicit features that are associated with the segment. There are certainly many ways to improve the per-segment performance, including adding more segment-specific features, and training dedicated models for each segment. But choosing the best cut-off threshold score per segment turned out to be the most practicable and effective way to achieve the goal.

In this work, we propose an efficient approach to optimize the segmented job-jobseeker recommendation performance by tuning the per-segment model thresholds. Specifically, we formulate a constrained optimization to identify the potential improvement space per segment.

RecSys in HR'22: The 2nd Workshop on Recommender Systems for Human Resources, in conjunction with the 16th ACM Conference on Recommender Systems, September 18–23, 2022, Seattle, USA.

Three attempts are discussed, and the most feasible one is adopted in the production. We also apply a greedy search algorithm to speed up the segment-specific threshold tuning process. Our prototype implementation and the corresponding AB tests on selected segments had suggested considerable improvements in terms of better recommendation quality and higher volume on both applies and positive outcomes from the job applications.

In summary, the main contributions of this paper are as follows. We hope our work can provide a reference on similar problems in the industry.

- We report the occupation-based segment-level investigation using the real-world data from our organization.
- We formulate a constrained optimization problem to facilitate our segmentation work in the job-jobseeker recommendation system.
- We propose an effective way to optimize the per-segment performance by tuning the thresholds on different models.
- We implement an automated model threshold tuning module into the pipeline, and the online experimental results from the real products indicate promising performance improvement on both recommendation quality and coverage.

The rest of the paper is organized in the following ways. In Section 2, we discuss and review a few related works. In Section 3, we provide an overview of our existing recommendation platform. In section 4, we describe the segment-level model threshold tuning that we use to optimize the recommendation performance. In Section 5, we illustrate the evaluation results on three selected segments. And finally, Section 6 concludes this work.

2. Related Work

Many existing works had studied the overall framework of efficient job recommendation systems. Kenthapadi et al. [1] discussed the candidate selection, personalized relevance model, and match redistribution, as three main sub-systems in the job recommendation system at Linkedin. Lu et al. [2] presented a hybrid-ranking system by combining the interaction-based and content-based features from both job and jobseekers, and calculate a ranking score accordingly. Shalaby et al. [3] built a graph-based job recommendation framework at CareerBuilder.com, using a similar hybrid approach by combining the behavior-based and content-based data together into weighted scores for the ranking purpose. Diaby et al. [4] proposed a taxonomy-based job recommender system that segmented both job and jobseekers into a taxonomy system using their occupation data.

There were also works focusing on jointly examining the resumes from jobseekers and the job descriptions from the job side, mostly for the high-tech job profiles. Malinowski et al. [5] presented a probabilistic-based CV and job recommender, that relied extensively on the structured resume data from a limited number (i.e., 100) of high-skilled jobseekers. Javad et al. [6] used named entity recognition (NER) to explicitly extract the skills from resume, and further used them to facilitate the recommender system. Qin et al. [7, 8] proposed a neural network based representation to embed the skills from resume and job descriptions, and ranking the matches based on the vector similarities. Luo et al. [9] introduced adversarial learning to learn more expressive representation from similar sources. However, the majority of jobseekers in the labor market (e.g., truck driver, retail sellers, etc.), do not have properly written resumes, if not completely without a resume. Consequently, such methods might not work well for these jobseekers.

While most existing job recommendation systems [10, 11, 12] tried to have one model worked for all different job profiles, very limited work noticed the significant difference among these job and jobseeker profiles. This work, on the contrary, attempts to identify such differences and make operational optimization correspondingly, with the objective to improve the overall recommendation performance.

3. Overview of Our Match Recommendation Platform

This section presents an overview of the match recommendation platform, and justifies the segment-level optimization is needed. In particular, we first present our probabilistic-based models that are still driving a significant number of recommendations within our organization. We then study the feature distribution from both the job and jobseeker side, and identify the performance gap across a variety of segments.

3.1. Probabilistic-based Models

Our recommendation match provider is built on top of a series of probabilistic-based models. Each of them takes care of a single step along the application funnel. In particular, as depicted in figure 1, each model takes a subset of features from job, employer, jobseekers' contents (e.g., resumes, questionnaires, etc.) and behaviors (e.g., apply history, feedback from previous applications, responses to previous recommendations, and inferred interests, etc.) as the input features. And the model outputs the probability for its own step.

More specifically, the first model focuses on if the jobseeker responds to the recommendation, given that the

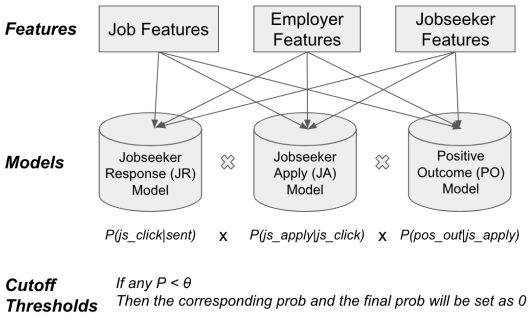

Figure 1: There are three models in tandem to construct the probabilistic-based filtering and ranking module for job-jobseeker recommendation.

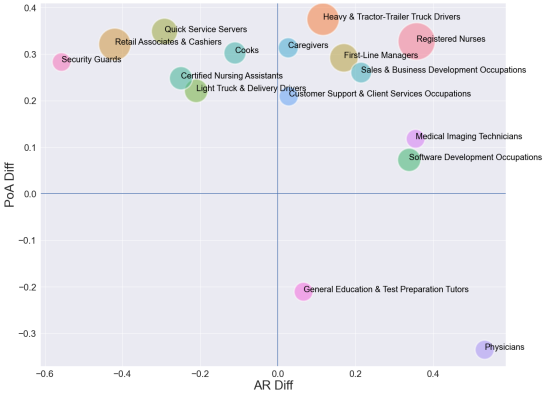

Figure 2: Performance difference between recommendation and other organic channels across different segments.

recommendation had been sent out. It can be any response, such as clicking the "apply job" or "not interested" button, unsubscribing, replying, or giving out a rating. The second model focuses on if the jobseeker actually applied for the job, given he/she had made any kind of responses to our recommendation. The third model deals with the probability further on the employer side, focusing on if the employer sends any positive outcome to the submitted application, such as follow-up conversations to further understand the applicant, interview arrangements, or even making an offer.

$$\begin{aligned} p(posOut|sent) = p(jsClick|sent) \\ *p(jsApply|jsClick) \\ *p(posOut|jsApply) \end{aligned} \tag{1}$$

We conduct both the scoring as shown in Eq.1, and filtering as shown in Eq.2, based on this model chain. In particular, each logistic regression model follows the sigmod function to generate the probability output $p(s|g)$, where g is the ground truth, s refers to the stage that the model is dealing with. $f(x_n) = \sum w_n x_n$, and x_n refers to the input feature vector, w_n refers to the weights to be trained for each feature. At the same time, there is a customized threshold θ for each model, to filter out the matches having low probability at that stage. Eventually, only the matches that pass all the three cutoff threshold at each stage will be assigned a non-zero score to represent the probability of having a positive outcome, given a sent recommendation $P(posOut|sent)$. Finally, this score will be passed into the ranking and aggregation module as the next step. It is easy to see that the multiplication leads to higher precision but lower recall, because it will be filtered out as long as the job-jobseeker pair gets a low score at any stage in the chain.

$$p(s|g) = \begin{cases} 1/(1+e^{-f(x_n)}), & \text{if } 1/(1+e^{-f(x_n)}) > \theta \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

3.2. Performance Gap among Segments

From our historical data, we found there are significant performance gaps across different segments in terms of their occupation, as we used to have one unified set of models with the same threshold values, to serve all the jobs and jobseekers from different backgrounds. This observation is based on a segment-level comparison on the recommendation performance with other organic channels, where jobseekers search and find jobs by themselves. Ideally, we expected the recommender system would consistently perform better, because it should provide better matches with higher accuracy. However, such an assumption is not always true.

Figure 2 studies the performance gap in terms of Apply start Rate (AR) and Positive outcome over Apply (PoA) of the biggest 16 occupations from our real-world data. The AR metric indicates the quality of the jobseeker engagement, while the PoA metric indicates the quality of the employer engagement. It is clear that a bunch of segments (mostly blue-collar jobs) are with low AR but okay PoA, indicating the model gets higher precision but lower recall there. While a few other segments (mostly white-collar jobs) are suffering from low PoA but okay AR, indicating the model gets higher recall but lower precision. After all, we believe all these segments could have considerable improvement spaces, but through different initial locations and different directions towards the top right corner as shown in the figure 2.

Note that, we cannot directly compare the absolute metrics across different segments, because the performance can be affected by the segment nature, instead of the recommendation quality.

The examination motivates us to further check if these segments are big enough to try the segment-level optimization. And if so, we also would like to understand the reasons that lead to such performance gaps.

Figure 3: There are multiple top-level occupations in the job markets. Each occupation accounts for a certain percentage, but no one clearly dominates the whole population.

3.3. Segment-level Investigation

Figure 3 shows the segment distribution in terms of the number of active jobs based on their top-level occupation from our organization in 2022H1. We clearly serve a full spectrum of jobs and jobseekers from a variety of occupations, without any single occupation clearly dominating the whole population. Every occupation-based segment occupies a certain portion in the job market. As a result, the segment-level optimization could have a reasonable expectation to benefit the overall performance.

We next want to examine if the performance gap originates from the different feature distribution among different segments. Specifically, we look at a mixture of blue-collar and white-collar jobs, on both the job and jobseekers side. As expected, the blue-collar jobseekers (e.g., delivery drivers, retail sellers, etc.) tend to have a much shorter resume, which in turn makes the skill and experiences extraction, or even resume embedding less representative than the white-collar jobseekers (e.g., software development, technical managers, etc.). Similar pattern can be observed on the job side too, where the white-collar jobs tend to list more job requirements in terms of hard skills and experiences, while blue-collar jobs tend to focus more on licences and soft skills.

The observations lead us to reconsider if our existing approach using the same model set with the same threshold setting is good enough to handle all these cases. Although we already use a few segment-specific data (e.g., job title, industry) as the model features, we suspect they might not be representative enough to properly differentiate the specific requirements. As a result, we work on a few different approaches on the segment-level optimization, and discuss the feasibility based on our real-world experiences in the next section.

4. Segment-level Optimization

There are a number of possible ways to do the segment-level optimization for our probabilistic-based recommendation system. In particular, we report three different attempts that we have tried in this section. For each attempt, we evaluate not only its effectiveness, but also its scalability in the long run.

4.1. First Attempt: dedicated models per segment

The most intuitive solution that first came to us was to build dedicated sets of models for each segment. We selected a list of low-performed occupation-based segments (i.e., Security Guard, Retail Store Manager, and Quick Service Server) according to Figure 2, and trained a dedicated set of models for each segment. As a result, every segment got three different models as shown in Figure 1, and they were trained by only using the historical dataset from that specific segment.

Surprisingly, the initial experimental results did not align well with our expectation, showing mixed signals in terms of the recommendation quality and volume. In particular, for all the three experimented segments, we observed significant decreases in terms of the Applystart Rate (AR) or Positive outcome over Apply (PoA) ranging from -9.8% to -15.6%, while an improvement in terms of the number of apply starts and positive outcomes ranging from 6.0% to 13.8%. However, our expectation on the dedicated models was to have considerable improvements on all the key metrics at the same time.

After a close examination on the approach and the corresponding models, we found three major issues that lead to the disappointing results. First, we did not setup a formulation to properly represent the overall objective. Consequently, we even did not have a clearly defined expectation and target for the optimization at the very beginning. Second, we over-emphasized the model training part, whereas we missed the fact that the cutoff thresholds play even more important roles to trade-off precision and recall. Therefore, we believe the dedicated models still need careful threshold tuning, to maximize its benefit. Lastly, we noticed that we were ongoing many other initiatives (such as an alternative way of embedding features, or even adding new features, etc.) that kept improving the baseline models from other members within our organization, while our treatment models were kept unchanged during the experiment. This made the experimental comparison inconsistent over time. More importantly, we could not fix this easily, because the large-scale model auto-updates together with the parameter fine-tuning could be too expensive in terms of both the initial engineering efforts and the following infrastructural maintenance.

4.2. Second Attempt: online reinforcement learning with multi-armed bandit

By learning the lessons from our first attempt, we would like to formulate an optimization problem to appropriately capture our task and the objective. In particular, we want to simultaneously improve both recommendation quality and volume, on all the key metrics including applystart volume, positive outcome volume, applystart rate, and positive outcome over apply. While we can focus slightly more on AR for those low-AR segments, or positive outcomes for those low-PoA segments. And the control variables that we can operate here are the thresholds for each model per segment.

$$\max_{\vec{\theta}} \quad \sum_{i\in\{a,ar,p,poa\}} \lambda_i \Delta_i(\vec{\theta}) \tag{3}$$

$$\text{s.t.} \quad 0 < \lambda_i < 1, i \in \{a, ar, p, poa\} \tag{4}$$

$$\sum_{i\in\{a,ar,p,poa\}} \lambda_i = 1 \tag{5}$$

$$\Delta_i(\vec{\theta}) = \frac{new_i(\vec{\theta}) - base_i}{base_i} > 0, i \in \{a, ar, p, poa\} \tag{6}$$

$$slo_j(\vec{\theta}) < slo_j - \epsilon_j, j \in \{unsub, neg\} \tag{7}$$

As a result, we formulate a constrained optimization problem as shown in Equation 3 to 7. Specifically, the objective function aims to maximize the weighted combination of all the key metrics, including apply start volume a, apply start rate ar, positive outcome volume p, and positive outcome over apply poa. For each metric, λ_i represents the weight for us to shift focus between low-AR and low-PoA segments, and Δ_i indicates the corresponding performance improvement. In the meanwhile, there are a few Service Level Agreements (SLOs) that we must meet, including the unsubscription rate must lower than 0.05%, and negative feedback ratio from jobseekers must be lower than 25% among all the feedback. These SLOs are hard requirements, so that we even want to add a marginal buffer ϵ to the constraint. Both Δ and slo metrics can be affected by the threshold setting $\vec{\theta}$. With these definitions, our task is to find out the optimal model thresholds for each segment that could maximize the objective function, while fulfilling all the constraints.

With the clearly defined objective (or reward) function and constraints, one possible way is to adopt multi-armed bandits as a reinforcement learning approach to find out the optimal solution in the online environment. Specifically, we can setup multiple test groups in the production, each group has different threshold settings. Then we keep monitoring the performance on the objective value and constraints, and adjust the traffic allocation towards the better performing variances gradually. Some sampling methods such as Thompson sampling could speed up the convergence rate to some extent.

However, there were still a list of issues that prevented us from doing efficient multi-armed bandit tests for our segmented threshold optimization. First, the underlying baseline models were being iterated in parallel, resulting in the inconsistent and unreliable comparison among different treatment groups with fixed threshold settings. Second, the online reinforcement learning could take a long time to get converged, especially when a few target segments are with small sample size. Last but not least, we also suffered from the delayed data issue from the up-streaming data sources, considering the signals from employer-side (e.g., interview schedule and results, etc.) could take up to a few weeks to come back after the application had been made. Consequently, this attempt is unfortunately also impractical to our problem.

4.3. Third Attempt: offline threshold tuning per segment

By learning from the previous two failed attempts, we confirmed that fine-tuning the thresholds for each segment could be the feasible solution to optimize the performance. But it was not practical to find out the optimal solution throughout the reinforcement learning approach over the online iterations. As a result, we came up with our third attempt by using a proper offline evaluation algorithm based on the historical job and jobseeker interaction data from all the channels.

Algorithm 1 describes the proposed greedy searching process to find out the optimal threshold settings per segment in an efficient manner. Specifically, the algorithm takes a few different inputs, including the models (i.e., Jobseeker Response JR model, Jobseeker Apply JA model, and Positive Outcome PO model) as discussed in Section 1, the historical data for model performance evaluation per segment, and a default threshold setting. We select an upper u and a lower bound l for each model respectively, by plus-minus a range over the default value. We then follow a greedy searching process to find out the optimal settings, that can achieve the best performance $\vec{P} = (O, a, ar, p, pos)$ in terms of the objective value O and the four key metrics as defined in Equation 3 to 7. The expected outputs are the optimal threshold settings $\vec{\theta_o}$ per segment, which can achieve no worse performance than the default ones.

The greedy part originates from the fact that JA model threshold correlates well with the applystart rate, and the same pattern applies to PO model threshold and the positive outcome over apply. On the other hand, when we increase any model threshold, the applystart and positive outcome volume can only go down or at most flat. As a result, if we want to improve both quality and volume as

Algorithm 1 Greedy Threshold Searching Algorithm

Require: Models: JR, JA, PO
Require: Historical job-jobseeker interactions
Require: Default threshold set $\vec{\theta}_d$

function greedySearch($\vec{\theta}$, $\vec{P}$, $m1$, $m2$)
  for θ_{m1} **in** $range(l_{m1}, \vec{\theta}_d(m1), s_{m1})$ **do**
    $\vec{\theta}_t \leftarrow (\theta_{jr}, \theta_{m1}, \vec{\theta}_d(m2))$
    if $getVol(\vec{\theta}_t)(m1) < \vec{P}(m1)$ **then**
      break
    end if
    for θ_{m2} **in** $range(u_{m2}, \vec{\theta}_d(m2), -s_{m2})$ **do**
      $\vec{\theta}_t \leftarrow (\theta_{jr}, \theta_{m1}, \theta_{m2})$
      if $getRatio(\vec{\theta}_t)(m2) < \vec{P}(m2)$ **then**
        break
      end if
      if $meetSlo(\vec{\theta}_t)$ **and** $getObj(\vec{\theta}_t) > \vec{P}(O)$ **then**
        $\vec{\theta} \leftarrow \vec{\theta}_t$
        $\vec{P} \leftarrow getPerf(\vec{\theta})$
      end if
    end for
  end for
  return $\vec{\theta}$, $\vec{P}$
end function
$\vec{\theta}_o \leftarrow \vec{\theta}_d$
$a \leftarrow getVol(\vec{\theta}_d)(JA)$, $ar \leftarrow getRatio(\vec{\theta}_d)(JA)$
$p \leftarrow getVol(\vec{\theta}_d)(PO)$, $poa \leftarrow getRatio(\vec{\theta}_d)(PO)$
$O \leftarrow getObj(\vec{\theta}_d)$
$\vec{P} \leftarrow (O, a, ar, p, poa)$
for θ_{jr} **in** $range(l_{jr}, u_{jr}, s_{jr})$ **do**
  $\vec{\theta}_o$, $\vec{P}$ ←greedySearch($\vec{\theta}_o$, $\vec{P}$, JA, PO)
  $\vec{\theta}_o$, $\vec{P}$ ←greedySearch($\vec{\theta}_o$, $\vec{P}$, PO, JA)
end for
return optimal threshold set $\vec{\theta}_o$ per segment

the key metrics at the same time, we need to search the JA and PO model threshold in different directions starting from the default value. In addition, once we reach the boundary, by either observing a volume that is smaller than default when increasing the threshold, or a ratio that is smaller than the default when decreasing threshold, we do not need to further down the same direction. However, the JR model does not display a clear relationship with our targeted key metrics, therefore, we still do a full grid search on the JR model in the outer loop.

Figure 4 illustrates an example of searching the optimal threshold settings for Security Guard over three dimensions (i.e., JA and PO model threshold serve x-axis and y-axis, while the objective value is set along z-axis). It is obvious that we only need to check the areas with non-zero objective values. Whereas most of the areas with zero value can be ignored due to the negative volume

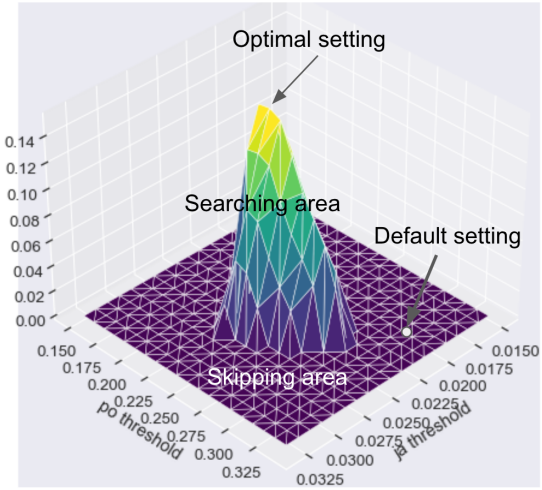

Figure 4: A 3d illustration of the greedy threshold searching regions on the Security Guard segment, with the JA and PO model threshold as x and y axis, and the objective value as the z-axis. It is clear that we can only search the region with non-zero objective value, and skip all the non-zero regions.

or ratio change, or SLO violations. In this way, the proposed algorithm can be 10x to 15x faster than the full grid search, by skipping these regions. The speed-up factor could be even bigger when we have limited knowledge about a new segment, thus requiring a wider searching boundary with smaller steps. The greedy algorithm enables us to have the optimal threshold searching process running more frequently and efficiently for every model update, but also to scale up the optimization process to all the segments.

5. Performance Evaluation

By following the third attempt as discussed in the previous Section, we further integrate the algorithm into our model training pipeline as a prototype implementation. The performance is evaluated throughout proper online A/B testing from real recommendation products. The experimental results demonstrate promising signals for all the three selected segments. Moreover, the approach is generally applicable for all the segments.

5.1. Prototype Implementation

Under our current model pipeline, the unified model set is retrained upon either the regular daily update or a production release on various other model improvement initiatives. Previously, the retained models would be put into the production model storage, thus they can be directly invoked by the online recommendation system

Segment	Positive Outcomes	POs/Applies	Apply Starts	ASs/Sends
Security Guard	+17.29%	+20.61%	+2.46%	+14.87% ↑
Retail Store Manager	+86.49% ↑	+8.16%	+95.51% ↑	+38.05% ↑
Quick Service Server	+1.02%	+4.30%	-11.54%	+30.72% ↑

Table 1
Final performance evaluation via online A/B experiment on selected segments via threshold optimization, where ↑ denotes a statistically significant increase ($\alpha = 0.05$ with a two-sided t-test). Almost all segments showed promising improvements by comparing with the baseline group, which uses a default threshold for all segments.

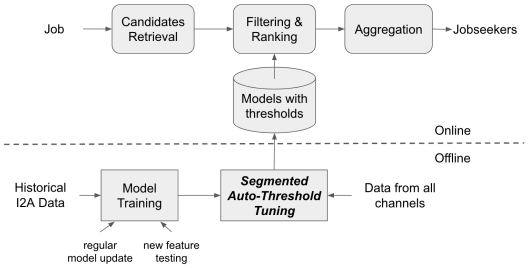

Figure 5: The functional workflow indicates the way we integrate our segmented auto-threshold tuning module into the model training and adaptation pipeline.

by using a fixed set of default threshold values.

Figure 5 describes the conceptual design of our prototype implementation, on top of the existing pipeline. Specifically, we introduce a new stand-alone segment auto-threshold tuning module to host our algorithm, and plug it into the offline stage. The module is able to generate the optimal cutoff threshold settings per segment for the newly trained models, before the online recommender system can actually use them. As a result, we make sure the online matches are always proceeding with the optimal thresholds for each segment. In the experiment, we set the weight λ equally at 0.25 for all the four key metric changes. Different λ settings could have various impacts on the objective values, and thus the optimal threshold settings, but due to the page limitation, we are not going to dig deeper into this point. Note that, we take the offline historical interactions for all the job-jobseeker pairs from all other channels as the inputs for our algorithm, because this would allow us to not only check the potential impact if we increase the threshold, but also enable us to see the estimated impact if we decrease the threshold for certain models.

5.2. Online AB Test Results

We continue to focus on the same three low-performed segments (i.e., Security Guard, Retail Store Manager, and Quick Service Server), but with a more rigorous online A/B testing plan. In particular, we run the online A/B experiments for two weeks with the power analysis indicating at least apply start volume or ratio should be able to reach the statistical significance given 15% estimated effect size within this period.

Table 1 elaborates the experimental results from the online A/B testing, with ↑denotes the statistical change for that metric. In particular, we find all the three segments show significant improvement in terms of positive outcome volume, positive outcome over applies, and applystart ratio. Such observation aligns with our offline evaluation. However, it does note us that while Security Guard and Retail Store Manager segments also have considerable improvements on apply start volume, the Quick Service Server segment display a negative signal at around -11%. Although we believe our offline evaluation is overall reliable enough for most segments, there are still the difference between online and offline data due to the usage of the historical data from other sources.

While low-performed segments are more likely to have bigger improvement spaces as we observed in the experiment, the proposed approach is generally applicable to all the segments. In particular, first, the optimization framework can figure out if a specific segment can be improved by the threshold tuning. In the worst case, the existing threshold settings are already in the optimal range, and our proposed algorithm can quickly confirm this. But if there is an opportunity, we can also accurately identify it, and further find out the optimal settings accordingly.

6. Conclusion

In this work, we presented an effective solution to improve the performance of our job-jobseeker recommendation system. Specifically, we started by identifying the performance gap among different segments, followed by segment-level investigations. We then reported three different attempts, and came up with the most feasible approach by tuning the model thresholds per segment. The detailed solution was presented, including a proper problem formulation as a constrained optimization problem, an efficient algorithm to speed up the threshold optimization process, and the prototype implementation. Finally, online A/B tests from real products proved the performance improvement in terms of both recommendation quality and quantity.

Our future work will mainly follow three venues. First, we are going to scale up the auto threshold optimization to more segments, and also figure out the way to minimize the difference between offline evaluation and the actual online performance. Second, we will evaluate if similar segmentation work can benefit other match providers that are based on more sophisticated models (e.g., neural networks, or deep collaborative filtering). Third, we will extend our segmentation optimization work for the same match provider into our international markets, where the user behaviors and job requirements can become different even for the same occupation across different countries and markets.

References

[1] K. Kenthapadi, B. Le, G. Venkataraman, Personalized job recommendation system at linkedin: Practical challenges and lessons learned, in: Proceedings of the eleventh ACM conference on recommender systems, 2017, pp. 346–347.

[2] Y. Lu, S. El Helou, D. Gillet, A recommender system for job seeking and recruiting website, in: Proceedings of the 22nd International Conference on World Wide Web, 2013, pp. 963–966.

[3] W. Shalaby, B. AlAila, M. Korayem, L. Pournajaf, K. AlJadda, S. Quinn, W. Zadrozny, Help me find a job: A graph-based approach for job recommendation at scale, in: 2017 IEEE international conference on big data (big data), IEEE, 2017, pp. 1544–1553.

[4] M. Diaby, E. Viennet, Taxonomy-based job recommender systems on facebook and linkedin profiles, in: 2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS), IEEE, 2014, pp. 1–6.

[5] J. Malinowski, T. Keim, O. Wendt, T. Weitzel, Matching people and jobs: A bilateral recommendation approach, in: Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06), volume 6, IEEE, 2006, pp. 137–145.

[6] F. Javed, P. Hoang, T. Mahoney, M. McNair, Large-scale occupational skills normalization for online recruitment, in: Twenty-ninth IAAI conference, 2017, pp. 4627–4634.

[7] C. Qin, H. Zhu, T. Xu, C. Zhu, L. Jiang, E. Chen, H. Xiong, Enhancing person-job fit for talent recruitment: An ability-aware neural network approach, in: The 41st international ACM SIGIR conference on research & development in information retrieval, 2018, pp. 25–34.

[8] C. Qin, H. Zhu, T. Xu, C. Zhu, C. Ma, E. Chen, H. Xiong, An enhanced neural network approach to person-job fit in talent recruitment, ACM Transactions on Information Systems 38 (2020) 1–33.

[9] Y. Luo, H. Zhang, Y. Wen, X. Zhang, Resumegan: an optimized deep representation learning framework for talent-job fit via adversarial learning, in: Proceedings of the 28th ACM international conference on information and knowledge management, 2019, pp. 1101–1110.

[10] T. A.-O. Shaha, Y. Mourad, A survey of job recommender systems, International Journal of Physical Sciences 7 (2012) 5127–5142.

[11] F. Abel, A. Benczúr, D. Kohlsdorf, M. Larson, R. Pálovics, Recsys challenge 2016: Job recommendations, in: Proceedings of the 10th ACM conference on recommender systems, 2016, pp. 425–426.

[12] J. Dhameliya, N. Desai, Job recommender systems: A survey, in: 2019 innovations in power and advanced computing technologies (i-PACT), volume 1, IEEE, 2019, pp. 1–5.