Lessons Learned — Building ML Models to Remove Irrelevant Results in Job Search

Gabriel Womark, Ritvik Kharkar, Ishan Shrivastrava
Ziprecruiter, Inc
604 Arizona Ave
Santa Monica, California 90401 USA
{gabrielw, ritvikk, ishans}@ziprecruiter.com

Abstract

The presence of irrelevant search results is an important problem to solve for any modern search engine. For job search engines, this problem manifests when a job seeker receives an irrelevant job for a query. Such results can occur due to insufficient measurement of relevance within the system or due to competing objectives such as monetization. The occurrence of such results worsens the job seeker experience, harms revenue, and can affect long-term brand image. Therefore, it is crucial to identify and address the issue of irrelevant search results. We propose a supervised machine learning approach for removing irrelevant search results. We share offline and online results and lessons learned along the way.

1 Introduction

Search engines are crucial in the online job search ecosystem. At ZipRecruiter, our keyword search engine drives job seeker applications to both paid job listings and organically scraped jobs, the latter providing additional value by expanding job options for job seekers.

Figure 1: a “border patrol agent” job that appears for a “recruiter” query due to a reference to “Recruitment” in the job title and high employer bid

Our search engine aims to balance relevance for job seekers with delivery for paying employers. However, this balancing act can result in irrelevant results for job seeker queries such as the result shown in Figure 1. Given the growing level of internal feedback on the incidence of such irrelevant results for key job seeker queries and with the goal of building a more robust product, we wanted to engineer a sufficient solution.

To address irrelevant results, we first adopted a manual approach: identifying problematic queries, diagnosing root causes, devising targeted solutions, and implementing them in production. For instance, the query “Human Resources” often returned unrelated jobs matching the term only in descriptions, in clauses such as “speak to a Human Resources Rep”. We resolved this by filtering out jobs that match “Human Resources” *only* in their descriptions.

We quickly encountered several problems with this manual approach:

- Targeted solutions impacted too little traffic to measure changes in engagement with statistical significance.
- We lacked a clear method to assess the severity of relevance issues or identify problematic queries. While engagement data offered some relevance signals, it was limited by insufficient data for infrequent queries and influences from non-relevance factors like job distance, posting age, etc.

A broader solution was needed to address relevance issues across many queries, capable of estimating the relevance of each candidate result and filtering out irrelevant ones. Our solution involved:

- Defining relevance grades to measure query-job relevance.
- Creating a labeled data set of query-job pairs.
- Training a supervised machine learning model to predict result relevance.
- Setting a relevance threshold to determine whether results should be removed or kept.

We evaluated the model offline and through an online experiment. Although it did not meet the metrics needed for deployment to production, the process provided the following valuable insights for removing irrelevant jobs with machine learning:

- Measurement of relevance improvements through engagement signals can be error-prone.
- Sufficiently descriptive semantic signals are required to prevent the removal of relevant jobs.
- The quality of relevance labels is imperative for training an accurate supervised machine learning model.

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

2 ML Based Approach

We aimed to create a data set of triplets $D = (q_i, j_i, l_i)$ where (q_i, j_i) is the ith query-job pair and l_i was its corresponding relevance label. From there we would generate a feature vector x_i for each query-job pair in the data set. This feature vector x_i would consist of features that could easily be computed at inference time. With this data set and the corresponding feature vectors, we could train a supervised learning algorithm to predict the relevance of a given query-job pair.

2.1 Defining Relevance

To measure the relevance of a query-job pair, we defined five grades of relevance as shown in Table 1. Crowd-sourced annotators were asked a prompt containing these relevance definitions and asked to label a few thousand query-job pairs accordingly. Consistent with the language in these definitions, we are trying to understand how likely a job seeker would be to apply to a job if they had searched a given query.

Grade	Description
0	Horrible match and I definitely would not apply—**extremely irrelevant**.
1	Bad match and there's a very low chance I would apply—**irrelevant**.
2	I understand why I see this job but am hesitant to apply—**somewhat relevant**.
3	I will actually apply to this job even though it's not an absolute perfect match—**relevant**.
4	This job is perfect and I will certainly apply—**extremely relevant**.

Table 1: relevance grades are based on users willingness to apply

2.2 Collecting Relevance Labels

The process outlined in Section 2.1 where we send query-job pairs for labeling to crowd-sourced annotators eventually proved to be a bottleneck in two ways. First, the velocity at which we were getting labels was slow, delaying progress on training our relevance predictor. Second, the dollar cost of obtaining these labels from annotators was getting prohibitively high.

Given recent successes in using Large Language Models for labeling tasks (Rahmani et al. 2024), we decided to try and save time and money by engineering a prompt for GPT 4o which would label query-job pairs in as similar a way as possible to how annotators would label them. To quantitatively measure this agreement, we used a commonly used metric called Cohen's κ (McHugh 2012) to measure the level of agreement between two raters beyond random chance.

After several rounds of prompt engineering, we eventually landed on a prompt which uses the labeling guidelines in Table 1 at its core. The full prompt is omitted for confidentiality.

2.3 Model Features

In order to predict relevance, we used a combination of aggregated lexical and semantic features. The aggregated lexical features were statistical aggregations (minimums, maximums, averages, etc.) of term frequency (TF) and inverse document frequency (IDF) (Qin and Liu 2013) aggregated over all terms in a job seeker's search query. We included one such set of aggregated lexical features per field in the job; job fields included job title, job description, job company name, etc.

We also included one semantic matching feature to attempt to fill in gaps that could not be addressed using lexical features. For example, a job with title "nurse" is relevant for a job seeker query "medical" even though such a job may never explicitly mention the word "medical". This semantic matching feature was the product of an existing model, was a function of the job seeker's query and canonicalized job title, and was bounded between 0 and 1.

2.4 Model

After collecting relevance labels, the size of our data set D was about 27000 query-job-relevance label triplets. The size of the data set led us to believe that a very sophisticated model (such as a deep-learning model) would overfit to the data. On the other hand, an overly simple model (such as a logistic regression model) would likely underfit the data as it would be unable to capture the non-linear relationships between the features we planned to use. Thus we went forward with a Gradient Boosted Decision Tree Classifier (GBDTC) to predict relevance on binary relevance labels.

2.5 Training

Since our goal was to remove only the most irrelevant jobs, we binarized the quinary relevance labels such that a binary relevance label $b_i = \begin{cases} 0 & l_i < 2 \\ 1 & l_i \geq 2 \end{cases}$

Furthermore, before training, we split D into training and test sets using a $70/30$ split.

We used the LightGBM GBDT API to train the GBDTC. To increase robustness to data set class imbalance we set the training parameter *is_unbalance* to true. To prevent overfitting we set the training parameter *early_stopping_rounds* to 5 such that if *auc* on the test set did not improve over 5 training epochs, we'd select the best performing model on *training_auc* before those 5 epochs. The result was a GBDTC with 16 trees and an average tree depth of 10.

3 Offline Evaluation

The model produces a probability that a given query-job pair is relevant, so we need to threshold that probability to create binary predictions. We evaluated thresholds by plotting a receiving operator characteristic (ROC) curve. The sensitivity on the curve represents the probability we would keep a job for a given search if it was truly relevant; the specificity represents the probability we would remove a job given that it was truly irrelevant. So as to create a minimal impact on online engagement our goal was to find a model threshold such that sensitivity was equal to 0.9.

We could tell if a model m' improved upon a model m if specificity(m') $>$ specificity(m) when sensitivity$(m') =$ sensitivity$(m) = 0.9$.

In Figure 2 we see an example of how adding a semantic feature to the model can increase specificity by 5 percentage points while keeping the sensitivity at 0.9.

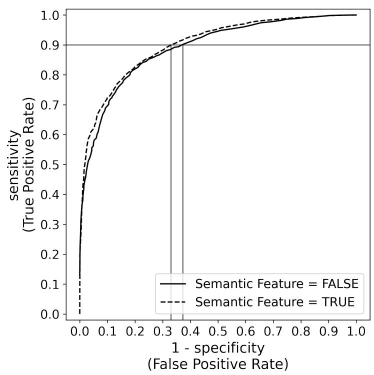

Figure 2: improvement on false positive rate from including a semantic feature in classification

4 Online Evaluation

To evaluate the model online, we would run our relevance classifier in-between the retrieval and monetization reranking steps of a search. The classifier would accept up to 1000 job listings gathered by the retrieval step, predict the probability of relevance on each job, filter out jobs based on a threshold we determined, and then send the remaining jobs to the reranking step. For our experiment we A/B tested six different thresholds on predicted probability of relevance using a single trained model. That model included all the lexical features described in Section 2.3 as well as the semantic feature. The thresholds were selected to balance between removing as many irrelevant jobs as possible while keeping a sufficient number of jobs in the set for job seekers to browse.

Given the volume of data we collect in an online A/B test and our limited LLM labeling budget, there was no way for us to assess the quality of our classifier at different thresholds by labeling *all* the query-job pairs for each test variant in the experiment. Thus we had to rely on online engagement metrics and revenue metrics to assess the quality of our classifier. More specifically, our search engine result page has two panes in its user interface. The left pane presents a list of jobs with minimal details such as the title, company name, location, etc. A click on a job in this left pane exposes a more detailed description of that job in a right pane. At the top of the right pane is a button that allows the job seeker to begin the process of applying to the displayed job. We call clicks on this button “apply-starts”.

We believed that apply-starts would have a strong correlation with relevance (as defined in Section 2.1) as a job seeker’s willingness to apply to a job was a strong indication of its relevance. Clicks on the left pane seemed to be a weaker indication of relevance as they don’t account for the fact that the job seeker had observed a job description. On the other hand, completed applies and hiring signals are often too sparse for us to measure with statistical significance.

Therefore, we assumed that if our model improved the relevance of search results, we would see an increase in apply-starts per job seeker while preserving the amount of revenue we generate from listings that pay us on a per-apply-start basis.

Table 2 displays metrics from our online experiment. The leftmost column displays the threshold applied to predicted model probabilities to determine whether a job should be filtered out of search results. The middle column shows the relative performance of each test variant on apply-starts per job seeker when compared to the control variant. As we can see, 99% credible intervals show that there was no statistically significant improvement or reduction in apply-starts per job seeker. The only exception is the variant with a threshold of 0.74 which showed a statistically significant reduction in apply-starts per job seeker.

Threshold	**%-Relative Lift Apply-Starts Per Job Seeker (99% CI)**	**%-Relative Lift Revenue Per Job Seeker (99% CI)**
0.34	[-3.46%, 1.59%]	[-6.26%, -2.28%]
0.40	[-2.35%, 1.79%]	[-7.36%, -3.49%]
0.42	[-2.35%, 2.73%]	[-6.97%, -2.92%]
0.49	[-2.08%, 3.17%]	[-9.96%, -6.17%]
0.51	[-4.66%, 0.25%]	[-9.83%, -5.93%]
0.74	[-7.27%, -2.45%]	[-11.60%, -7.76%]

Table 2: relative lift in apply-starts and revenue by threshold show no significant losses in clicks for all thresholds aside from *0.74*, while all test threshold show significant loss in revenue

The rightmost column of Table 2 shows the relative performance of each test variant on revenue per job seeker when compared to the control variant. As we can see, 99% credible intervals show a statistically significant decrease in revenue per job seeker on all test variants. This can likely be attributed to the fact that all test variants reduced the number of jobs that job seekers could engage with. Interestingly we do not see a linear decrease in revenue as the threshold for relevance gets more strict.

Based on this table, we can infer that for test variants where we saw an insignificant change in apply-starts per job seeker but a significant decrease in revenue, we were likely removing jobs that would have yielded revenue on an apply-start (“paid apply-starts”) and replacing them with jobs further down the result set that did not pay for engagement.

While we had anticipated the possibility that there would be no increase in apply-starts per job seeker, due to the fact that we were simply removing job listings and replacing them with results further down in the result set, we did not expect to see a loss in revenue from these apply-starts.

Based on these results we decided not to deploy the model to production with any of the thresholds we tested.

5 Post - Mortem

Although we could not release the model to production due to a decrease in paid apply-starts, we still had many questions about the online performance of the classifier:

- What was the distribution of relevance grades for paid apply-starts that were removed by the classifier? Was it possible that the paid apply-starts we removed were truly irrelevant, despite receiving engagement from job seekers?
- If the paid apply-starts were labeled relevant, how could we recover those removed by the classifier?
- Were the labels generated by our LLM labeler of high quality? If the paid apply-starts we removed were labeled in the middle relevance category (quinary grade 2), was it possible these labels were incorrect?

5.1 Equating Apply-Starts and Relevance

It is natural to believe that a job seeker's willingness to apply to a job served to them in search results is strongly correlated with its relevance for the query. This is an assumption we had made in Section 4. However, we did not take the time to verify this assumption before running our online experiment.

To verify this assumption, we took one month of historical jobs served in search results that had received paid apply-starts. We ran the classifier used during the online experiment with a threshold of 0.49 to classify whether these paid apply-starts would have been removed by our classifier. This threshold was selected because it had the least negative impact on user engagement compared to other thresholds. We then sampled 5000 predicted-removed paid apply-starts and the respective queries they were served for and applied the labeling process we discussed in Section 2.2. We used this labeled sample to produce a distribution of $P(l|A \cap R)$ where l is a quinary relevance label and A indicates membership in the set of paid apply-started jobs and R indicates membership in the set of jobs removed by our classifier.

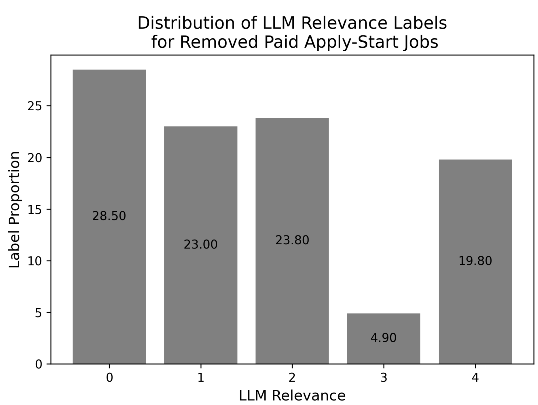

Figure 3: distribution of true relevance grades for paid apply-starts which were removed by the classifier

Assuming our labeler is correct, Figure 3 shows that roughly 50% of the removed paid apply-start sample has a relevance label of 0 or 1, which are considered irrelevant using the binary scheme proposed in Section 2.5. Certain queries exhibit this pattern more than others. For example, the query "work from home" has nearly 75% of its removed paid apply-starts labeled as irrelevant. Many of these paid apply-starts are jobs requiring presence in the office, which seems to counter the intention of the query.

We realize that this analysis doesn't provide a direct measurement of the probability *any* apply-start would be labeled irrelevant. This is due to the fact that we only generated labels for a very specific segment of apply-starts, namely those that would be removed by our classifier and generated revenue from engagement. However, this analysis does show us that even if we improve the accuracy the of the classifier, we would still see at least 50% of the loss in paid apply-starts that we observed in the online experiment. In the future we will have to decide whether this inherent potential revenue and engagement loss is an acceptable cost for improving the relevance of search results.

5.2 Recovering Relevant Apply-Starts

In spite of the fact that 50% of removed paid apply-starts are truly irrelevant, the fact remains that the other 50% are on truly relevant jobs. In order to launch the model into production we'd need to reduce this quantity.

To identify why these relevant paid apply-starts were removed, we analyzed our model's training data to discover what feature combinations lead to incorrect predictions by the classifier.

We discovered that the hardest examples for the classifier to predict correctly were ones in which there was a missing value for the semantic feature described in Section 2.3. Concretely, we find that the probability of having a missing value for our semantic feature is about 20% for truly relevant jobs which are removed by the classifier and is about 75% for truly relevant jobs which are kept by the classifier. This indicates a possible gap in our semantic feature such that when this feature is absent, the classifier mistakenly deems the corresponding job as irrelevant.

This implies the need for more descriptive features that could effectively capture portions of a job that semantically match the query.

5.3 Assessing Quality of LLM Relevance Labels

Finally, we investigate whether the relevance labels generated by the LLM are of high quality. To do this, we sample a set of 100 pairs of queries and paid apply-started jobs. These jobs are assigned an LLM quinary relevance label of 2 yet removed by our classifier. The idea is to only consider those jobs whose quinary relevance grade is on the cutoff between being relevant and being irrelevant which might therefore be the most susceptible to mislabeling.

Given these 100 query-job pairs, we assign three of our internal team members to independently label them using the quinary relevance scale. We find that 33% of these pairs have no majority label and 67% do have a majority label. This finding was very insightful for our future work as it showed that even three independent humans find it challenging to align on a quinary relevance grade for those jobs which the LLM labeled on the boundary between being relevant and being irrelevant. For those 67% of pairs where there *was* a majority quinary relevance grade, 85% of them were labeled as having a quinary grade of 2 or higher, meaning that we admitted a 15% false negative rate even when majority agreement was reached.

All together, this exercise highlighted the fact that data labeling is not fully objective and even two different annotators can disagree on what the "correct" label should be.

6 Future Work

Having identified important barriers to pushing out the relevance classifier in production, our next steps will primarily focus on addressing these issues. Given that we stand to remove irrelevant jobs from search results that generate revenue from job seeker engagement (Section 5.1), we'll need to estimate the amount of revenue lost from using a perfect classifier (i.e. our LLM labeler) on a small sample of historical search data. This could allow us to set guardrails on online engagement and revenue metrics that account for the fact that not all engagement is relevant.

Furthermore, this misalignment between engagement and relevance may suggest that engagement metrics are not a good barometer for relevance improvements. A possible solution to this problem is to distill the knowledge of our LLM labeler into a much smaller model. That model could be used to generate relevance labels on the large volume of online experiment data. While this smaller model wouldn't be able to be used in production because it cannot meet our latency requirements for serving results to job seekers, it could still be feasible to label online experiment data and create online experimentation metrics based on relevance (Wang et al. 2024). This would also allow us to better measure the trade off between relevance, engagement, and revenue discussed in the previous paragraph.

To improve the recall of our classifier (Section 5.2), we can incorporate more semantic signals by using transformer models to generate query embeddings and job embeddings that are stored in our search index. We can use similarities from query and job embeddings as a feature for the model to capture additional semantic signals that our current classifier model does not capture (Haldar et al. 2020).

Lastly, we'd like to improve the quality of our labels by consulting with domain experts (Section 5.3). We believe domain experts should be able to devise more objective definitions of relevance that we can provide to our LLM Labeler. With more consistent labeling, we believe we can effectively reduce the confusion of our classifier model, especially in cases where the relevance of a job for a query might not be clear.

7 Conclusion

In this paper we discussed how the keyword search engine at ZipRecruiter tries to balance between showing relevant jobs and generating revenue from job seeker engagement. Trying to strike that balance can lead to surfacing of irrelevant search results. We highlighted the difficulty of solving this problem using manual fixes and proposed a new approach that could generalize to fixing this problem on all search queries using both established and novel ML technology. Though the proposed approach did not yield significant improvements in key business metrics, our analysis revealed valuable insights. We discovered that even an accurate relevance classifier might decrease engagement as job seekers can interact with irrelevant results. We also discovered the important need for more descriptive semantic features to make up for weak lexical and only-basic semantic matching signals indicating relevance. Finally, we discovered that relevance labeling can be a highly subjective task, even for a human. Without providing more clear objectives for labeling, we don't believe model performance could be improved. We've shared these lessons learned with the hopes to contribute to others' efforts to improve their keyword search engines and provide job seekers a better search experience.

References

Haldar, M.; Abdool, M.; Ramanathan, P.; Sax, T.; Zhang, L.; Mansawala, A.; Yang, S.; Turnbull, B. C.; and Liao, J. 2020. Improving Deep Learning For Airbnb Search. *CoRR*, abs/2002.05515.

McHugh, M. L. 2012. Interrater reliability: the kappa statistic. *Biochem Med (Zagreb)*, 22(3): 276–282.

Qin, T.; and Liu, T. 2013. Introducing LETOR 4.0 Datasets. *CoRR*, abs/1306.2597.

Rahmani, H. A.; Yilmaz, E.; Craswell, N.; Mitra, B.; Thomas, P.; Clarke, C. L. A.; Aliannejadi, M.; Siro, C.; and Faggioli, G. 2024. LLMJudge: LLMs for Relevance Judgments.

Wang, H.; Sundararaman, M. N.; Gungor, O.; Xu, Y.; Kamath, K.; Chalasani, R.; Hazra, K. S.; and Rao, J. 2024. Improving Pinterest Search Relevance Using Large Language Models. arXiv:2410.17152.