Multi-Labeling Service for Auto-Targeting Pipeline in Job Recommendation

Li Ji*
Jingyi Zhu*
Yu Sun
Chang Li
Xuefei Yang
{rji,jingyiz,sunyu,changl,xyang}@indeed.com
Indeed
Sunnyvale, CA, USA

ABSTRACT

Ads targeting aims to learn the relevance ranking among potential audiences for each ad. It boosts advertising campaigns' performance by reaching prospective job seekers in time. In most existing online advertising systems, the targeting rules of ad campaigns are set up manually by either ad-ops or advertisers. This self-service scenarios may be sub-optimal, especially when the ad-ops or the advertisers are inexperienced.

This work puts forward and experiments a minimum viable product in solving automatic audience targeting for sponsored jobs. We build a knowledge graph embedding (KGE) of several entities, which incorporates various features such as job seeker user profile signals. We then leverage the KGE to map each ad campaign to one or multiple targeting labels (a.k.a. segments). This KGE-based multi-labeling approach reduces manual labeling errors, alleviates the computational burden on relevance ranking, and supports the subsequent task of audience expansion. This method demonstrates its strength in improved recall and precision, in comparison with the baseline methods in Indeed's offline and online experiments.

CCS CONCEPTS

• **Computational Advertising** → **Recommendation Systems**; • **Multi-Label Classification**; • **Knowledge Graph Embedding**;

KEYWORDS

automatic targeting, knowledge graph embedding, approximate nearest neighbor

ACM Reference Format:
Li Ji, Jingyi Zhu, Yu Sun, Chang Li, and Xuefei Yang. 2022. Multi-Labeling Service for Auto-Targeting Pipeline in Job Recommendation . In *Proceedings of 1st International Workshop on Computational Job Marketplace (WSDM '22).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Both authors contributed equally to this research.

WSDM '22, February 21–25, 2022, Tempe, Arizona

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Computational marketplaces, such as Indeed.com, CareerBuilder and LinkedIn Inc., are dedicated to delivering relevant job posts to potential job seekers [2]. Overall, targeting aims to extract interpretable and useful features from three integral components in the online marketplace: `context`, `user` (i.e., job seekers), and `ad` (i.e., job posts). Utilizing the *current* characteristics of users' access behaviors, geo-targeting, channel targeting, and contextual targeting are three essential techniques to construct `context` labels. Taking users' *historical* data into consideration, demographic targeting and behavioral targeting are two fundamental strategies to construct `user` labels. As for `ad` labeling, we may either perform manual labeling or directly use the advertisers, ad groups, keywords as labels, see [4, Chap. 12] for a comprehensive overview. Aside from labeling these three components individually, there are techniques to construct labels for `user` and `ad` jointly–as in re-targeting and look-alike methods–by integrating the attributes from these two aspects. This work puts forward a novel `ads`-labeling approach that leverages knowledge graph embedding (KGE). We will use ads[1] and job posts interchangeably.

1.1 Literature Review

The granularity of `ad`-based audience targeting ranges from the finest keywords to the coarsest job posts. There are primarily two ways to mine the job posts.

(i) We can define a set of hand-crafted self-explanatory labels (e.g., advertisers, ad groups, keywords) in advance, and then map job posts to one or more labels in the established set via supervised learning.
(ii) We can obtain a set of labels via unsupervised learning (e.g., clustering) while enforcing a hard threshold to the number of labels. In such a way, we may obtain the mapping function from the job posts to the set of labels obtained from unsupervised learning.

Choosing between (i) and (ii) depends on how ads labeling will be utilized in the advertising system. If the ads labeling only serves

[1]It is possible that multiple jobs are attached to one `ad`. This paper discusses labeling the jobs only.

for feature extraction subsequently, then either supervised or unsupervised learning paradigms are acceptable. However, if it serves as a labeling system for the advertisers, then supervised learning is preferred due to the interpretable labels that are conveniently defined in advance.

For the latter approach (ii), we can tackle it via either multi-label classification or supervised topic modeling, such as supervised Latent Dirichlet Allocation (LDA) [1], hierarchically supervised LDA [5], and other methods reviewed in [3]. This work puts forward a KGE-based approach to handle approach (ii).

1.2 Our Contribution

We propose a KGE-model-based procedure to map a job post to one or multiple labels, along with its system design. This auto-targeting back-end service has these functionalities and features:

- For every requested job post, it returns a list of labels. More importantly, this service is capable of handling large-scale traffic from clients.
- It uses Non-Metric Space Library (*NMSLIB*) to explore the approximate nearest neighbor (ANN) for relevant labels. Specifically, it uses the cosine similarity metric to score the label against a job.
- It mitigates the cold-start problem that arises for either new jobs or new labels.

Benefits. The newly-proposed KGE-based multi-labeling for ad posts avoids human labeling errors and supports the subsequent task of audience expansion. It seamlessly bridges the gap between targeted ads and search ads [4, Sect. 13.2], see further discussion in Sect. 2.5. Besides, *without* manual labeling, our model-based multi-labeling for job posts mitigates the computation burden on relevance ranking, so that relevance ranking in later stages can place more emphasis on precision over recall. Last but not least, Algorithm 1 to appear can further incorporate various additional features such as job seeker user profile signals.

2 MULTI-LABELING SERVICE (MLS)

This section explains the challenges and presents our approach of multi-labeling service for job posts. Under the hood, MLS loads the model trained offline based on ad features, including job metadata, job title, job location, etc.

2.1 Challenge

To build an ad multi-labeling model, the greatest challenge comes from the shortage of labeled data. Reconstructing the label(s) for each job ad based on historical data may be demanding as one ad could be mapped to multiple labels *subjectively*, and sometimes may be misleading as the labels *may* be *time-varying*. Even if we manage to recover the labels from the history, we will not be able to support cold-start problems which arises when either new labels or new jobs come in.

2.2 Our Approach

2.2.1 Normalization of Job Titles and Queries. Normalization is the process for converting data–raw job titles "`rawTitles`" and raw queries "`rawQueries`" in our discussion–that have more than one possible representation into one of a known set of *normal* or *canonical* titles. These canonical titles and queries are called `normTitles` and `normQueries`. The canonical title is the best job title among a group of job titles that are synonyms. After you choose a canonical title, all related raw titles will be normalized to that canonical title, regardless of its original formatting and/or its extra information. Figure 1 gives some examples of how raw job titles are normalized to normalized job titles.

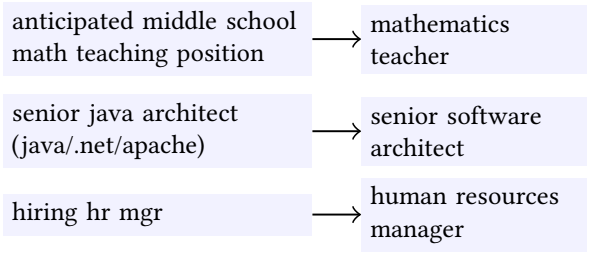

Figure 1: Examples of `rawTitles` being normalized to `normTitle`

2.2.2 Constructing Labels via KGE. Given that we do not have labeled data and do not plan to label the gigantic raw data, we resort to KGE for rapid iteration and model refresh.

Knowledge graphs are collections of triplets, each triplet (*head*, *relation*, *tail*) among which represents a *relation* between a *head* entity and a *tail* entity. Specifically, we apply *RotatE* algorithm introduced in [6], to obtain a KGE over the pre-selected relevant nodes and relations. The nodes of the graph include the user id (`userId`), the normalized query (`normQuery`), and the normalized job title (`normTitle`). The two relations of our interest are "`normQuery` was queried by `userId`" and "`normTitle` was applied by `userId`".

With this trained KGE, we construct labels that embrace both the normalized queries and the normalized job titles. In detail, from a sub-graph in the knowledge graph training dataset, we start from a *head* node that contains one user id (`userId`). After that, from the corresponding *tail* node which contains either one normalized job title (`normTitle`) or one normalized search query (`normQuery`), we *recursively* locate more similar titles and queries by leveraging the trained KGE. Note that the recursion goes on until the similarity score reaches a fixed threshold. After several repetitions of both auto and manual quality checks, the resulting labels can be stored in a database with other machine-generated or human-selected labels. This approach of ads-labeling is best suitable for user-targeting by real-time monitoring if one user should be targeted based on his recent queries, clicked titles, and resume titles. We will cover the details more in Sect. 2.5.

For illustration, Fig. 2 includes a snippet of one label (HR) obtained through the aforementioned KGE. The fields `clickedTitle` and `resumeTitle` come from the *normalized* job title embedding, while the field `recentQuery` uses the *normalized* query embedding. The remaining machine-generated labels can be extracted from the KGE similarly. Note that these machine-generated labels *may* be harder to interpret compared to the hand-crafted labels by ad-ops or advertisers. Here we use embeddings built on the following entities:

- `resumeTitle` is the *normalized* titles appearing in the job seeker's resume and represents his past experience.
- `clickedTitle` is the *normalized* titles that the job seeker recently clicked and reflects his/her desired future career title
- `recentQuery` is the *normalized* search queries that the job seeker input and encodes his current interests.

With these background information in mind, the example label in Fig. 2 is "human resources based on their recent resume title similarity and search query similarity".

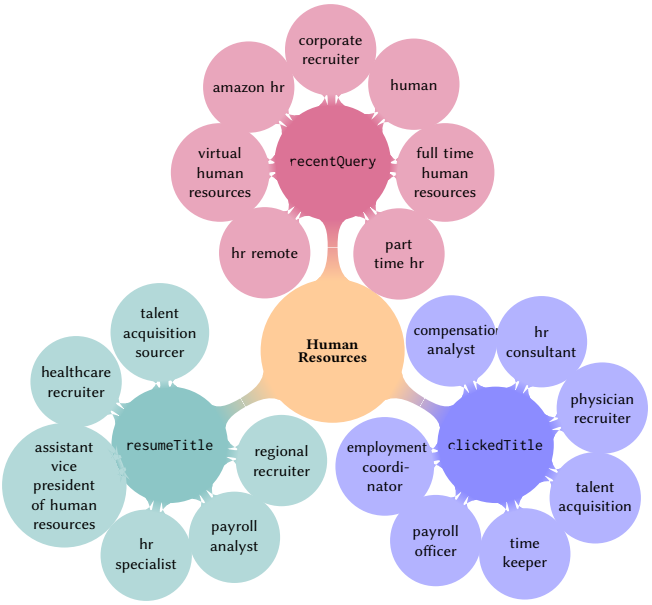

Figure 2: One Example Label (HR) Obtained via KGE

Last but not least, the foregoing time-varyingness problem in Sect. 2.1 can be partially addressed if this KGE model gets refreshed periodically.

2.2.3 Retrieving Labels via MLS. Sect. 2.2.2 explains how we recursively obtain labels using the KGE built from the normalized titles and normalized queries. The set of KGE-based labels then serves as the core component of MLS, since these labels are composed of normalized titles and normalized queries. Besides, each job has other metadata other than the normalized job title.

MLS consumes the labels definition database periodically and performs mean-pooling (taking the mean of the individual embedding) over the item embedding of each section into three vectors. Note that the missing terms are omitted. Afterwards, we index the corresponding three vectors with an id by library *NMSLIB*. This step can be viewed as k-nearest-neighbor (k-NN) with $k = 1$. Currently, the approximate nearest neighbor (ANN) search is applied in Indeed's current production. The output is three indices under these names appearing in Fig. 2: `clickedTitle`, `resumeTitle`, and `relevantQuery`. Note that the first two fields use normalized title embedding and the last uses the query embedding.

When one single job is input into MLS, we call the indices to obtain three lists of scores, in order to generate its corresponding multi-labeling lists. Under the hood, each prediction method calls the indices to get three lists of scores. Precisely, the normalized title embedding of job title calls the indices `clickedTitle` and `resumeTitle`. Meanwhile, the query embedding of job title (viewing job title as a query) calls the index `recentQuery`. A list of labels is subsequently returned to the client after integrating the three lists of scores. See Algorithm. 1 for a summary.

Algorithm 1 KGE-based Multi-Labeling Service

Require: metadata of a single job, including `rawTitle`, `normTitle`, `jobType`, `category`, `companyInfo`, and etc.
Ensure: a list of recommended labels for the input job
  initialize label list *list* and target title *title*
  if `normTitle` of input job is empty **then**
    set *title* to be the most plausible `normTitle` from the set of all the `normTitle` by comparing with `rawTitle` based on fuzzy string matching
  end if
  if target title t is empty **then**
    set t by calling fall-back solution in Sect. 2.3.1 with transformer model
  end if
  for (`clickedTitle` index, `resumeTitle` index, `recentQuery` index) **do**
    if *score* is larger than a threshold **then**
      if *label* not in *list* **then**
        append the matched labels by label name and similarity score pair $(label, score)$ to *list*
      else
        add *score* to the existing *label* in *list*
      end if
    end if
  end for
  sort *list* by *score* descendingly

2.3 Fall-Back Solution

2.3.1 Missing Normalized Job Titles. One caveat for Sect. 2.2 arises in the edge case when the incoming job has no normalized job title. Note that the trained KGE is static (though it gets refreshed periodically), and the set of embedded terms cannot comprehensively cover all the upcoming terms. For example, we encountered "tattoo artist" and "tattoo shop" before any tattoo-related terms are incorporated into the pre-trained embeddings. To better manage the unseen cases, we can utilize *Transformer* [7] to generate word embeddings as a fall-back solution.

Note that we need the sentence and token embeddings to run ANN in *NMSLIB*. To enable mapping normalized titles and queries into fixed-length representations, we resort to the sentence transformer that utilizes *BERT*/*RoBERTa*/*DistilBERT*/*ALBERT*/*XLNet* in *PyTorch*.

To streamline the model, we pick *DistilBERT* whose number of hidden units is 768. Besides, we reduce the dimension down to 200 by adding a dense layer after the pooling. In short, *DistilBERT* is selected ultimately, since we need a lighter model than other

Transformer models and less dimension to index in order to balance online serving response time and model performance.

Following this route, we can embed the normalized titles and queries into fixed-length vectors and then index all of these vectors. In detail, to support this fall-back solution with ANN search, three more similar but different source-data indices (`clickedTitle`, `resumeTitle`, `recentQuery`) are prepared, in addition to the KGE-based ones in Sect. 2.2.3. For incoming job requests with its own normalized title or raw title, the 200 dimension word embedding will be generated by the same *DistilBERT* model with the dense layer before calling these indices to retrieve labels.

2.3.2 Broad Terms. Another tricky scenario arises when the desired normalized job titles and/or search queries are abnormally broad. Take "technician" as an example. After the embedding training, this label is associated with the following titles and queries.

- `clickedTitles`: "hvac", "electrician", "field technician", "maintenance professional", "mechanic", "technician", "machine operator", "service technician", "maintenance technician", "hvac technician", "installer", "construction", "hvac tech", "maintenance"
- `appliedTitles`: "hvac", "electrician", "field technician", "maintenance professional", "mechanic", "machine operator", "service technician", "maintenance technician", "hvac technician", "installer", "construction", "hvac tech", "maintenance"
- `relevantQueries`: "hvac", "electrician", "mechanic", "electrical", "technician", "service technician", "maintenance technician", "hvac technician", "building maintenance", "installation", "manufacturing", "construction", "apartment maintenance", "hvac tech", "automotive", "maintenance"

Because of these broad terms, another fall-back solution is to use *RapidFuzz*[2] with a matching score of 95–somewhat higher than the recommended score of 85–to ensure delivering relevant job posts to job seekers.

2.4 Other Relevant Components

In between clients (who send in multi-labeling request) and MLS, we also built a service that automatically selects labels for any incoming job called Label Selector Service (LSS). In short, the client input/send a list of jobs, and then LSS output/return a list of labels for each job. Note that it is the very component that bridges the gap between search ads and targeted ads.

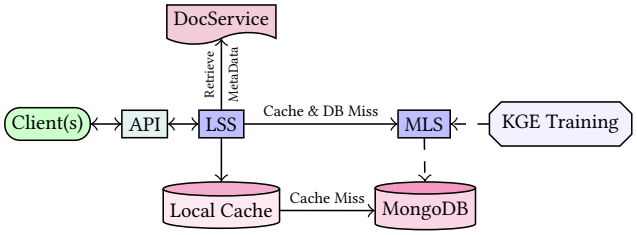

Figure 3: Architecture for Label Selector Service and Multi-Labeling Service. Solid/Dotted lines mean online/offline communications.

[2] *RapidFuzz* is a Python library that is used for string matching.

LSS starts by loading labeling data, which is written by MLS, into the cache from *MongoDB*. Afterward, LSS maintains a background thread/process that listens to MongoDB change stream and then updates the cache whenever data in MongoDB changes. When LSS receives a request from the client, LSS searches the correct labels in the cache to return to the client. In case of cache miss, LSS has to retrieve job metadata from DocService, call MLS to get the list of labels, and write the data to the local cache. Last but not least, an offline batch job can be set up to update the cache.

2.5 Auto-Targeting Pipeline

Other than categorizing targeting methods by the feature extraction sources (`context`, `user`, and `ad`) as in Sect. 1, targeting can also be categorized into explicit targeting (a.k.a. manual targeting) and implicit targeting (a.k.a. automatic targeting). Automatic (implicit) targeting refers to the scenario where targeting criteria are set up automatically for advertisers or ad-ops after campaign creation. Now that the KGE incorporates job titles, queries, and other information, and can be used as meaningful labels (e.g. Fig. 2), the MLS enables the fully automatic job ad targeting, as ad-ops and/or advertisers do not need to manually pick targeting labels (segments) in advance.

Now that Sect. 2.2 lays out MLS that obtains KGE-based labels, we can outline an ads auto-targeting pipeline in Figure 4. The optional block "audience expansion" usually facilitates expanding ads reach beyond the MLS. The automatic targeting pipeline such as Fig. 4 provides convenience when the advertiser is uncertain of its targeting audience. It allows for a scalable system handling a large number of ads and reduces human errors in tons of operational manipulations entailed in manual targeting.

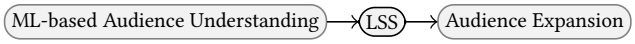

Figure 4: Automatic targeting may extract information from `context` (e.g., geo-targeting, channel targeting, contextual targeting), `user` (e.g., demographic targeting, behavioral targeting), and `ad` (e.g., MLS).

For the user matching side, the key fields (`recentQuery`, `resumeTitle`, and `clickedTitle` in Fig. 2) in the targeting labels are used to classify users into pre-selected segmentation. This task can be done either via a separate online service or an offline batch job. Besides, LSS can link jobs and labels as we proposed above. These two components comprise the auto-targeting pipeline.

Note that the auto-targeting pipeline in Fig. 4 does not entail manual labeling, this KGE-model-based multi-labeling for job posts mitigates the computation burden on ranking, which performs user response prediction by taking ads quality, contextual information, and bidding into account. Thanks to MLS and LSS, ranking in later stages can place more emphasis on precision over recall and the ads ecosystem in Fig. 5 gets healthier.

3 NUMERICAL EXPERIMENT

This section presents the performance of Fig. 3 inside Indeed.

Datasets. The dataset of our interest is around 588 million job posts, 539 million of which have normalized job titles. There are

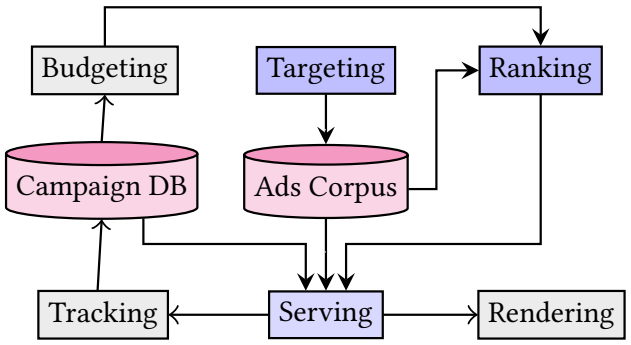

Figure 5: Ads targeting with LSS alleviates computational burden in ranking.

around 7 thousand normalized job titles. Note that nearly 19 million job posts do not have corresponding normalized titles, so the fallback solution in Sect. 2.3 experiments here.

Baseline and Implementation. To compare against the newly-proposed KGE-based method in Sect. 2, *RapidFuzz* is applied to label ad using `normTitle` *only*. We follow these criteria to return a list of matching labels for a requested job with normalized title and metadata.

- We set the score threshold in *RapidFuzz* to be 85 (recommended score) to increase ad reaching down the stream.
- For each `normTitle`, the labels with the highest matching score will be assigned, with a total cap of 6 labels.

Recall Improvement. Compared to the aforementioned baseline, the work discussed in 2 brings up both the recall on job and the recall on normalized norm title, see Table 1.

	Recall on Job	Recall on Title
RapidFuzz	86.83%	79.26%
KGE-based method	89.82%	81.64%

Table 1: Performance lift of Algorithm 1 compared with baseline: 3% lift on job recall and 2.4% lift on title recall.

Accuracy Improvement. Other than recall improvement on these unlabeled 588 million job posts, we also witness an accuracy lift from 60.97% to 74.79% on 246 normalized titles with human/manual labeling–a decent 13.82% lift on accuracy.

4 CONCLUDING REMARK

We put forward and experiment a minimum viable product (MVP) for an auto-targeting system, whose benefits are summarized in Sect. 1.2. Subsequent versions can be developed with several improvements.

- Currently, the set of labels is obtained through the embedding that utilizes `normTitle` and `normQuery` history. However, there exist some human-labels (labeling manually) that leverage further aspects, such as resume information (including `recentTitles` and `graduationDate`) and company information (including `recentCompanies`). Enriching our KGE is anticipated to facilitate profile customization for each company and other prospective objectives.
- For the moment, this service is limited to an "OR" operation of labels. Namely, Fig. 3 is capable of returning a list of labels for each requested job. We can extend this service to additional logical operations such as "AND", "NOT", and potentially nested operations, as discussed in [3].
- Although this work puts forward a complete model-based pipeline to label ads without human intervention, a supplementary matching and ranking layer may serve our entire computational advertisement better for *simultaneously* labeling both ad and user for relevance and reach purposes.

REFERENCES

[1] David M Blei and Jon D McAuliffe. 2007. Supervised Topic Models. In *NIPS*.
[2] Andrei Z. Broder. 2008. Computational advertising and recommender systems. In *ACM Conference on Recommender Systems*. 1–2.
[3] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.
[4] Peng Liu and Wang Chao. 2020. *Computational Advertising: Market and Technologies for Internet Commercial Monetization*. CRC Press.
[5] Adler Perotte, Frank Wood, Noemie Elhadad, and Nicholas Bartlett. 2011. Hierarchically supervised latent Dirichlet allocation. *Advances in neural information processing systems* 24 (2011), 2609–2617.
[6] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2018. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *International Conference on Learning Representations*.
[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.