Candidate Selection for Large Scale Personalized Search and Recommender Systems

Dhruv Arya
LinkedIn Corporation
darya@linkedin.com

Ganesh Venkataraman
LinkedIn Corporation
ghvenkat@linkedin.com

Aman Grover
LinkedIn Corporation
agrover@linkedin.com

Krishnaram Kenthapadi
LinkedIn Corporation
kkenthapadi@linkedin.com

ABSTRACT

Modern day social media search and recommender systems require complex query formulation that incorporates both user context and their explicit search queries. Users expect these systems to be fast and provide relevant results to their query and context. With millions of documents to choose from, these systems utilize a multi-pass scoring function to narrow the results and provide the most relevant ones to users. Candidate selection is required to sift through all the documents in the index and select a relevant few to be ranked by subsequent scoring functions. It becomes crucial to narrow down the document set while maintaining relevant ones in resulting set. In this tutorial we survey various candidate selection techniques and deep dive into case studies on a large scale social media platform. In the later half we provide hands-on tutorial where we explore building these candidate selection models on a real world dataset and see how to balance the tradeoff between relevance and latency.

CCS CONCEPTS

• **Information systems → Web searching and information discovery**; **Information retrieval**; **Information retrieval query processing**; **Query representation**; **Query intent**; *Query languages*; • **Software and its engineering → Software design tradeoffs**; **Search-based software engineering**;

KEYWORDS

Candidate Selection; Search; Recommender Systems; Personalization; Information Retrieval

ACM Reference format:
Dhruv Arya, Ganesh Venkataraman, Aman Grover, and Krishnaram Kenthapadi. 2017. Candidate Selection for Large Scale Personalized Search and Recommender Systems. In *Proceedings of SIGIR '17, Shinjuku, Tokyo, Japan, August 07-11, 2017,* 3 pages.
https://doi.org/10.1145/3077136.3082066

SIGIR '17, August 07-11, 2017, Shinjuku, Tokyo, Japan

ACM ISBN 978-1-4503-5022-8/17/08…$15.00
https://doi.org/10.1145/3077136.3082066

1 MOTIVATION

User experience at social media and web platforms is heavily dependent on the performance and scalability of its products. Applications such as personalized search and recommendations require real-time scoring of millions of structured candidate documents associated with each query, with strict latency constraints. In such applications, the query incorporates the context of the user and session (in addition to search keywords if present), and hence can become very large, comprising of thousands of boolean clauses representing cross features between the user, their explicit query and over hundreds of document attributes. Additionally a typical scoring function for both search and recommendations involves evaluating a function that matches various fields in the document with various fields in the query. This in turn translates to evaluating a function with several thousands of features to get the most personalized ranking. Consequently, candidate selection techniques need to be applied since it is infeasible to retrieve and score all matching documents from the underlying inverted index and bring down latency. On the other hand, reducing the candidate set could potentially involve loss of relevant documents. The tutorial helps provide a hands on experience to understand how to balance this tradeoff and delivering a relevant and fast user experience at scale. The tutorial consists of three parts. The first part presents a survey of candidate selection techniques used for search and recommender systems. The second part consists of case study on LinkedIn job search and recommendations and how we used candidate selection to bring down latencies by over 66%. The third part consists of a hands-on session where we use open source tools to explore candidate selection queries and its impact on relevance and latencies using a publicly available dataset. This part builds on the codebase used in the "Instant Search Tutorial" [10] at SIGIR 2016. This tutorial is aimed at both researchers interested in knowing about candidate selection techniques as well as practitioners interested in deploying a search and recommender system at scale.

2 OBJECTIVES

Large scale search and recommender systems attempt at matching the right documents from a corpus of hundreds of millions to billions of documents. Typically, the number of documents shown to the user is far lower than the number of documents that match a user query and profile. The matched documents could run in millions whereas the number of documents shown could be in tens, which has a significant overhead in terms of latency for online

search and recommender systems. Moreover, the latency requirements for end-to-end retrieval, i.e. candidate selection and ranking for candidate documents, need to factor the computation time for ranking by a nonlinear model document To run a complicated model and maintain low latency requires a sophisticated candidate selection model. Such a model at massive scale would involve the following components.

2.1 Indexing and Static Rank

The index is ordered in such a way that the documents with highest probability of interaction (independent of query and user) are retrieved first. This enables stopping retrieval when sufficient number of high quality documents are matched.

2.2 Query Understanding

We can achieve significant reduction in number of documents scored by understanding the type and intent of the query. For example, if a user types in "oracle" into LinkedIn job search [7], we could assume that the user meant to search for jobs from company "Oracle" and rewrite the query as: +COMPANY:oracle Above reformulation significantly reduces the number of documents scored. This of course requires us to understand that "oracle" in fact represents a company.

2.3 Query Operators

We look into the use of complex operators like WAND [5] go beyond regular MUST/SHOULD/NOT and ensure at least certain type of documents are retrieved. In the "oracle" example shown above may be some users meant to search for "oracle db" in which case "oracle" could in fact represent a skill. By rewriting it based on company, we could lose on recall. WAND operators can be tuned to address these use cases. For example, WAND operator can be used to ensure that at least 1% of the documents retrieved MUST have a skill match for the term "oracle".

2.4 Offline Replay and Evaluation

Once index is ordered and query reformulated, one needs to evaluate the impact of changes efficiently. Traditionally, we rely on A/B testing. However, this may be inefficient, take a lot of time and potentially involve subjecting a small percentage of users to reduced user experience. Hence, it would be more prudent to use an offline replay framework [8] which replays real production queries to evaluate potential loss of recall.

2.5 Query Construction Algorithm

We make use of historical interactions with our dataset for a given baseline candidate selection query. As we explore new candidate selection queries, our objective is to maximize retention of documents with favorable actions and reduction of documents with an unfavorable action. With the inherent position and presentation bias in the training data from the previous candidate selection query model, we will make use of top-k randomized result set (collected by randomizing top-k results for certain sessions) or a golden dataset with human annotated quality labels. In summary objectives of the tutorial are as follows.

- Present a survey of techniques involved for solving all of the challenges mentioned above - indexing, query rewriting, replay framework and query construction
- Present a real use case of LinkedIn job search and recommendations and detail how we achieved very significant reduction in latency (over 66%) by employing candidate selection techniques
- Present open source tools and a real use case (along with code in Github) about how to solve this problem at big scale.

3 RELATED WORK

In Information retrieval (IR) systems inverted indexes are used to support keyword queries, and only top-k results need to be returned. Candidate selection enables use of more complex ranking functions while scoring a fewer number of candidate documents. In the space of candidate selection Broder et al. [5] formally introduced the concept of WAND (weighted AND) operator which utilizes tf-idf based heuristics to tune the weights for each of the clauses to facilitate retrieval at a large scale.

For inverted index based search systems early termination is a technique used to speed up retrieval. Use of early termination requires ordering of the posting list in such a way that the highest quality document as determined through a "static rank" is placed first. The rank is computed per document and is independent of the user or the query. Persin et al. [9] proposed use of within document term frequencies and Brin et al. [4] proposed use of document popularity as the static rank.

Agarwal et al. [2] propose a two pass approach to ranking. The first pass retrieves the top-k documents using approximate ranking. e second pass uses a more accurate model. The document and items are represented in a vector space. The first state ranking leverages this vector space representation to retrieve a limited set of documents. This gives more time for the second pass to work on a more accurate model.

More recently Covington et al. [6] propose a neural network based candidate selection for YouTube recommendations. In this work, the authors reduce the number of videos to be ranked to a few hundreds by converting the recommendation problem into an extreme multi class classification problem. A high dimensional embedding of each document and item (in this case user and videos) is learned using neural networks. A nearest neighbor search is executed in real time in this high dimensional space to retrieve the top N documents.

Aphinyanaphongs et al. [3] propose a way to learn Boolean queries in such a way that they are manageable by humans. Such queries are then used as filters.

In terms of the open source technologies elasticsearch [1] is a search server based on Apache Lucene that provides capabilities to build search engines. It also provides different query building apis to allow exploration of different candidate selection techniques to find the one that best serves the intended use case.

In this tutorial we aim to provide an overview of the candidate selection methodologies and explore how to build candidate selection queries to serve fast and relevant search results and recommendations at scale.

4 FORMAT AND DETAILED SCHEDULE

We provide details on the format and hands-on part of the tutorial. The hands-on part builds on the open source codebase used in Instant Search tutorial in SIGIR 2016.

4.1 Introduction and Problem Setting

- Motivate the necessity and importance of candidate selection in large scale personalized search and recommender systems, and describe the problem setting.
- Showcase examples from LinkedIn search and recommendations

4.2 Literature Survey

- WAND Operator
- Early Termination and Static Rank
- Two phase ranking as proposed by [2]
- NN based candidate selection at youtube
- Learning weights for boolean clauses

4.3 Candidate Selection

- Building static rank
- Query understanding and query operators
- Query construction methodology
- Offline replay and evaluation

4.4 Walking through a real world example

- Utilize a public dataset to explore relevance/latency tradeoff by building different candidate selection queries.
- Evaluation methodology of candidate selection models

5 SUPPORTING MATERIAL

Supported material and setup instructions will be provided prior to the tutorial. The material will also be accompanied with detailed READMEs and guidance will be provided to attendees during the tutorial.

REFERENCES

[1] Elasticsearch. https://www.elastic.co/products/elasticsearch.

[2] Agarwal, D., and Gurevich, M. Fast top-k retrieval for model based recommendation. In *Proceedings of the fifth ACM international conference on Web search and data mining* (2012), ACM, pp. 483–492.

[3] Aphinyanaphongs, Y., and Aliferis, C. Learning boolean queries for article quality filtering. *Medinfo 11*, 1 (2004), 263–267.

[4] Brin, S., and Page, L. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems 30*, 1 (1998), 107–117.

[5] Broder, A. Z., Carmel, D., Herscovici, M., Soffer, A., and Zien, J. Efficient query evaluation using a two-level retrieval process. In *Proceedings of the twelfth international conference on Information and knowledge management* (2003), ACM, pp. 426–434.

[6] Covington, P., Adams, J., and Sargin, E. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* (2016), ACM, pp. 191–198.

[7] Li, J., Arya, D., Ha-Thuc, V., and Sinha, S. 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* (2016), pp. 501–510.

[8] Li, L., Chu, W., Langford, J., and Wang, X. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM international conference on Web search and data mining* (2011), ACM, pp. 297–306.

[9] Persin, M., Zobel, J., and Sacks-Davis, R. Filtered document retrieval with frequency-sorted indexes. *JASIS 47*, 10 (1996), 749–764.

[10] Venkataraman, G., Lad, A., Guo, L., and Sinha, S. 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* (2016), pp. 1502–1511.