CEUR-WS.org/Vol-3589/paper_18.pdf

Behavior-driven Query Similarity Prediction based on Pre-trained Language Models for E-Commerce Search

Yupin Huang[1], Jiri Gesi[2], Xinyu Hong[1], Han Cheng[1], Kai Zhong[1], Vivek Mittal[1], Qingjun Cui[1] and Vamsi Salaka[1]

[1] *Amazon, Palo Alto, USA*

[2] *University of California, Irvine, Irvine, USA*

Abstract

Pre-trained language models (PLM) excel at capturing semantic similarity in language, while in e-commerce, customer shopping behavior data (e.g., clicks, add-to-cart, purchases) helps establish connections between similar queries based on behavior on products. This work addressed the challenges of using sparse behavior data to build a robust query-to-query similarity prediction model and apply it to a product search ranking system. Our contributions include a straightforward method for data generation, testing different model structures on both public PLMs and in-house PLMs fine-tuned with Amazon internal data. The fine-tuned in-house PLM model shows a 27.4% NDCG improvement compared with the BERT. And we designed an end-to-end pipeline that incorporates model outputs into prior feature. The prior scores can be used to impact ranking, matching, and recommendation systems. We tested the prior in an online experiment, which led to a significant improvement of 0.08% in the search click rate and a 0.03% reduction in the search reformulation rate. Overall, our approach has significant implications for improving search ranking and matching applications.

Keywords

Neural Networks, Large Language Model, Query-to-Query, Distillation, E-Commerce, Search Ranking

1. Introduction

Behavior features constructed based on user feedback towards corresponding queries are one of the most crucial features in ranking models [1]. They have been a key revenue driver in applications like search ranking and matching, click-through rate prediction, and sourcing. Though powerful, the user feedback signals are sparse. Taking Amazon's US website as an example, it receives several billion unique queries every year, but the majority of them do not have sufficient customer behavior signals (e.g., clicks, add-to-cart, and purchases) to build high-quality behavior features used in ranking. The same pattern happened on YouTube as well [2]. Thus, for long-tail unique queries, customer signals are too sparse to generate features. Although many queries are semantically similar [3] (for example, "acoustic noise-canceling panels", "soundproofing acoustic studio foam," and "sound-absorbing acoustic panels"), their corresponding signals differ significantly, resulting in unbalanced feature quality.

eCom'23: ACM SIGIR Workshop on eCommerce, July 27, 2023, Taipei, Taiwan

yupinhuang93@gmail.com (Y. Huang); fjiriges@uci.edu (J. Gesi); xinyuhong08@gmail.com (X. Hong); chenghan@amazon.com (H. Cheng); kaizhong@amazon.com (K. Zhong); vivekmit@amazon.com (V. Mittal); qingjunc@amazon.com (Q. Cui); vsalaka@amazon.com (V. Salaka)

© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

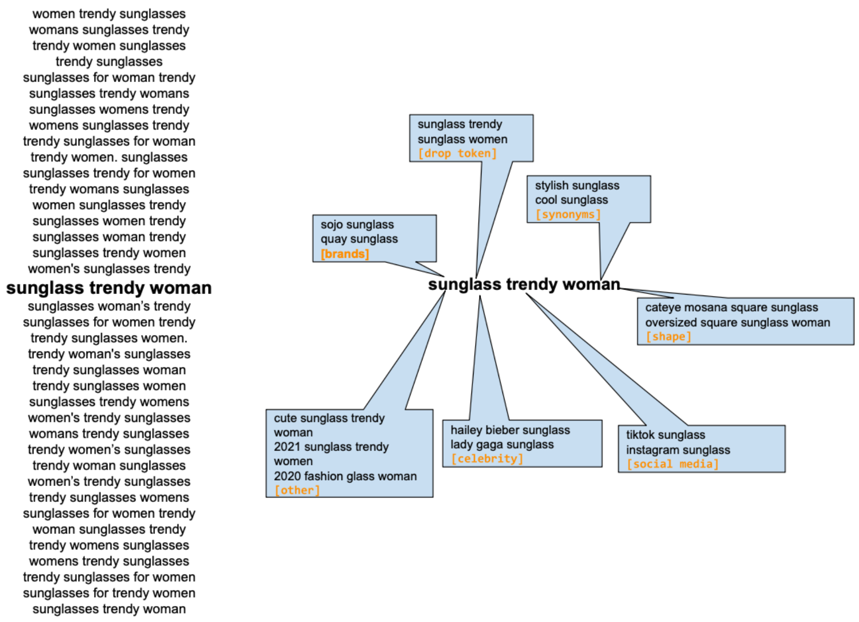
women trendy sunglasses
womans sunglasses trendy
trendy women sunglasses
trendy sunglasses
sunglasses for woman trendy
sunglasses trendy womans
sunglasses womens trendy
womens sunglasses trendy
trendy sunglasses for woman
trendy women. sunglasses
sunglasses trendy for women
trendy womans sunglasses
women sunglasses trendy
sunglasses women trendy
sunglasses woman trendy
sunglasses trendy women
women's sunglasses trendy
sunglass trendy woman
sunglasses woman's trendy
sunglasses for women trendy
trendy sunglasses women.
trendy woman's sunglasses
trendy sunglasses woman
trendy sunglasses women
sunglasses trendy womens
women's trendy sunglasses
womans trendy sunglasses
trendy women's sunglasses
trendy woman sunglasses
women's trendy sunglasses
trendy sunglasses womens
sunglasses for women trendy
woman sunglasses trendy
trendy womens sunglasses
womens trendy sunglasses
trendy sunglasses for women
sunglasses for trendy women
sunglasses trendy woman
sunglass trendy woman
sunglass trendy sunglass women [drop token]
stylish sunglass cool sunglass [synonyms]
sojo sunglass quay sunglass [brands]
cateye mosana square sunglass oversized square sunglass woman [shape]
cute sunglass trendy woman 2021 sunglass trendy women 2020 fashion glass woman [other]
hailey bieber sunglass lady gaga sunglass [celebrity]
tiktok sunglass instagram sunglass [social media]

Figure 1: Example of candidate queries for "sunglass trendy woman"

Pre-trained language models (PLMs) have a significant impact on query-related tasks. PLMs convert raw text into continuous, high-dimensional vectors that encode the semantic meaning of the text [4]. The distance (e.g., cosine similarity) between two vectors can measure whether the two queries are semantically related. We tested several PLMs, including BERT-base [5], Sentence-BERT [6] and InfoXLM [7] on a zero-shot setting. For the above mentioned query "acoustic noise-canceling panels", it introduces some defects like "noise canceling headphones" and "noise cancelling earbuds for sleep" which are undesirable. Some methods in e-commerce typically require several additional auxiliary tasks [8] to keep query intent or the creation of a query-product knowledge graph [9]. This is not practical to implement due to its complexity in generating training data with special requirements.

Our work leveraged the large amount of anonymized and aggregated customer behavior data to create training data and labels, then tested our model structures in several PLMs, including in-house PLMs trained with e-commerce data. We use this model to establish better query representation and ultimately improve feature coverage by mapping tail queries (usually longer, specific queries have low search volumes) to head queries (usually short, common queries have high search volumes) with better behavior signals. For production applications, we designed a two stage pipeline (retrieval and re-ranking) to incorporate the model outputs into behavior feature priors [10].

In this paper, we present three contributions:

- Introduced a straightforward yet effective method for collecting similar queries and creating labels from search logs only.
- Tested different model structures on several PLMs, including two in-house PLMs fine-tuned with Amazon internal data. Our experiments highlight the superior performance

and necessity of the in-house PLM, with a 27.4% improvement in offline NDCG.

- Designed a method to extract similar query behavioral scores into priors used for product search ranking models. The online A/B test conducted on 100M search sessions achieved significant improvement in both the revenue-aware metrics and user-engagement metrics, with the search click rate increasing by 0.08% and the search reformulation rate decreasing by 0.03%. By adopting similar queries' behavioral signals, we also observed a significant reduction in search defects in production.

2. Background and Related Work

2.1. Query Normalization

Query normalization is important in helping match user queries with relevant products when the query contains different forms or alternative expressions of the same concept. It consists of some common techniques [11] including tokenization, filtering, and stemming. In e-commerce, the user queries are short in length, so we found via experiments that sorting tokens alphabetically within a query could further reduce the number of unique queries while maintaining a high level of precision at Amazon. Figure 1 (left) shows 37 user queries like "women trendy sunglasses", "womans sunglasses trendy", and "trendy woman sunglass", etc. After query normalization like filtering (e.g., women's -> women), stemming (e.g., women -> woman) and sorting tokens, they can all use one query, "sunglass trendy woman" to represent. This significantly facilitates customer feedback signal sharing for building behavior features. Though powerful, it is limited at the lexicon level and thus cannot establish associations with queries that contain different tokens. Figure 1(right) shows some semantically related queries grouped by different types that our work aims to generate on top of query normalization.

2.2. Query Rewriting

Query rewriting is a crucial aspect of information retrieval and ranking, and it has been an active research field [12, 13, 14]. Three primary rewriting techniques exist: replacement-based, generation-based, and retrieval-based methods. Replacement-based methods employ synonym replacement [15, 16] or query term-dropping [17]. Figure 1 (right) showed some results in the [synonyms] and [token drop] sections that can be achieved using the replacement-based method. But these methods could lead to some poorly rewritten queries like "trendy woman" or "stylish sunglass women" that may not reflect the actual terms customers use in real-world scenarios. Generation-based methods typically utilize Seq2Seq models [18] or transformer models (BERT [5] and GPT [19]), which lead to significant improvements compared to traditional rule-based and statistical methods. Zhang et al. [8] developed a multi-task learning model that predicts the target query while also fulfilling query-matching, category classification, and product name prediction tasks to preserve the query's shopping intent. However, this increases the original task's complexity and makes label collection more challenging. Another challenge with generation-based models is the long inference time. Hofstatter's analysis [20] on the Fusion-in-Decoder model shows that the decoding latency is 10X of encoding thus is hard to meet the online latency requirement for our applications. To simplify the training task and better leverage the power of pre-training PLMs, we choose a retrieval-based method and build our work on top of PLMs. This ensures that similar queries come from the pre-defined candidate set that has good customer signals.

Collecting labeled data for query rewriting can be challenging and expensive. E-commerce queries are usually short in length and lack customer shopping context; thus, it's difficult to set judgment standards and train people to generate consistent labels. This made the weakly supervised technique of mining query pairs from search logs [21, 22] a widely accepted approach. The implementation varies in practice. Ozertem et al. [23] collect customer-rewritten queries within the same session; Fujita et al. [24] use co-click data to gather similar queries with higher click data rankings; Baeza-Yates et al. [25] create query clusters by extracting tokens from queries and clicked URLs, then identify similar queries within the input query's corresponding cluster. In this work, we designed a divide-and-conquer method to collect similar query pairs that can be effectively applied to large amounts of data.

3. Problem Formulation

We formally define the problem of predicting query similarity based on customer behavior data in e-commerce applications. We denote all input queries of all users by Q, which consist of a collection of queries q_i. Given an input query q_i, there could exist a set of candidate query set $C_i = \{c_1, c_2, ...\}$ from Q that are close to q_i. We train model to predict the similarity between q_i and each candidate query c_j, denoted as y_{ij}, ranging from 0 to 1. So the training data can be denoted as T = $\{(q_i, c_j, y_{ij})\}_{i\in\{1,2,...,|C_i|\}}^N$. In fact, it is not possible to generate a similarity score that precisely represents the relationship between two queries, as the actual score does not have a meaning in a real-world scenario. We tried to approximate this value by considering the co-purchase actions between them.

Traning Label Design To compute the similarity score y_{ij} between two queries (q_i, c_j), we define two types of similarity between queries: overlap similarity (OS) and jaccard similarity (JS). We use $S(q)$ to denote the unique purchased products from query q then OS is calculated by:

$$OS(q_i, c_j) = \frac{|S(q_i) \cap S(c_j)|}{\min(|S(q_i)|, |S(c_j)|)} \tag{1}$$

To prevent popular queries from dominating every query's candidate query list, we introduce JS. Popular queries are those that have many purchases of different products. For example, 'nike shoes' is a popular query that has many different products purchased. Since the denominator in JS uses the union of the purchased products from two queries, it tries to avoid every shoe- or Nike-related query having 'nike shoes' as the top similar candidate.

$$JS(q_i, c_{ij}) = \frac{|S(q_i) \cap S(c_j)|}{|S(q_i) \cup S(c_j)|} \tag{2}$$

We use the product of the two similarities as the label to represent the similarity between two queries.

$$QuerySimilarity(q_i, c_j) = OS(q_i, c_j) * JS(q_i, c_j) \tag{3}$$

We consider label y_{ij} to be close to 1 when queries q_i and c_j have the most co-purchased products overlap among other candidate queries, and close to 0 otherwise. We then create a model to fit on (q_i, c_j, y_{ij}) to score the similarity between the two queries.

Once generate a candidate query set C_i for a given query q_i, we could combine the signals from each query in C_i to generate a prior score (details in Section 4.5). We then combine prior with query q_i's behavioral signals to create a new ranking feature that is used in tree-based ranking models to improve ranking quality.

4. Methodology

Our proposed framework (in Figure 2) consists of three main components: the input layer, the encoder layer, and the similarity calculation layer. We tested both bi-encoder (BE) and cross-encoder (CE) structures in the encoder layer.

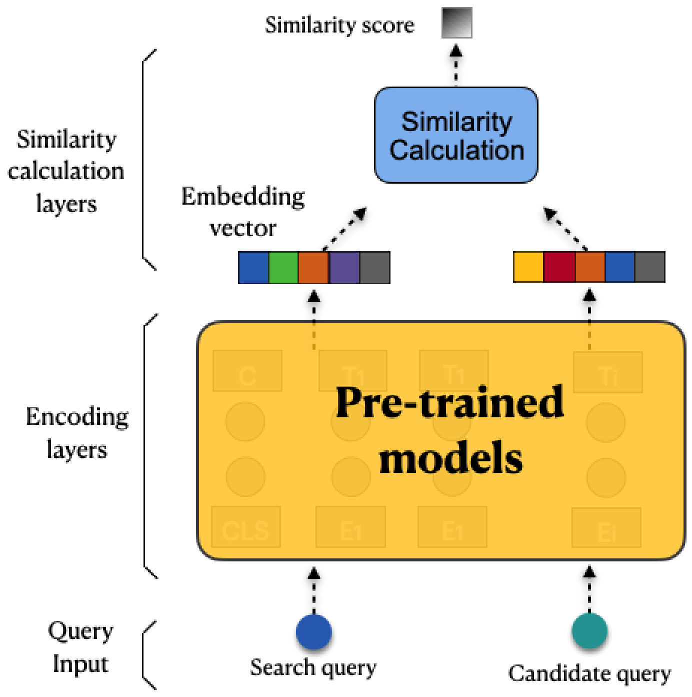

Figure 2: Q2Q model architecture

4.1. Input Layer

Input Data We use co-purchased products to bridge similar queries. After we retrieved a large pool of query pairs from the search log, we designed the following steps to filter and label the number of samples.

1. The two queries had at least three different co-purchased products within a year. This is to prevent weak query pairs from being generated.
2. For each query, we rank its similar queries based on a defined *QuerySimilarity* score.
3. For each query's ranked queries, we select the top 30[1] or top 60% queries. We chose the top 60% threshold here for very popular queries that could have thousands of similar queries.

Model Input For BE models, the search query and candidate query are fed separately into the pre-trained model, which outputs two separate embeddings. For CE models, we concatenate the two queries using the special tokens <CLS> and <SEP>. The input is like <CLS> Search Query <SEP> Candidate Query <SEP> and then feed it into pre-trained models.

4.2. Encoder Layer

4.2.1. Bi-encoder model

Under the bi-encoder structure (Figure 5 in Appendix A), each encoder uses a representation-based model that takes one query as input and outputs one feature embedding vector. Then, the generated embeddings of two queries can be fed into a simple dot product or perception to calculate the similarity. The advantage of a BE model is the low inference cost that enables online deployment with low latency requirements.

4.2.2. Cross-encoder Model

The CE model (Figure 6 in Appendix B) has multiple inputs, and they allow informational interactions at the early stage by leveraging their attention heads to exploit inter-query interactions. This interaction can be as simple as feeding two connected feature embeddings into a multi-layer perceptron (MLP) or be more complex, such as leveraging an attention mechanism between two input queries.

4.2.3. Pre-trained Language Model

PLMs are trained on massive multi-lingual corpora from the internet and have proven to be foundational game-changers for various natural language processing (NLP) and natural language understanding tasks. Amazon, which operates in over 20 countries worldwide, has a vast amount of product abd search log data. Thus, within Amazon, we also have PLMs fine-tuned for e-commerce. Table 1 summarizes the four different PLMs we tested in this project.

- BERT base [5] is transformer model pre-trained on a large corpus of English data in a self-supervised fashion using a masked language modeling (MLM) objective.
- Sentence-BERT [6] is a project that aims to train sentence embedding models on very large sentence level datasets using a self-supervised contrastive learning objective. It is pretrained on a dataset of 1 billion sentence pairs, i.e. given a pair of sentences, the

[1]We found that as we included more than 30 queries, the differences between the later ones were small and less relevant. We tested using the top 150 in model training and found the offline NDCG dropped by 1000 bps.

Table 1
PLM Comparisons

Pre-trained model name	Number of layers	Hidden size	Self-attention heads	Param- -eters	Training method	Training data
BERT base	12	768	12	110M	MLM	Book Corpus of 11,038 books and English Wikipedia
Sentence-BERT	12	384	12	33M	contrastive learning	1B sentence pairs dataset
A-PLMv1	6	768	12	158M	MLM + TLM	1B distinct queries + 266M parallel translations
A-PLMv2	24	1024	16	300M	MLM	Amazon internal data

model should predict which is the actual pair of sentences from other sentences randomly paired in the dataset. This pre-training method is similar to the Q2Q model task.

- Amazon in-house PLM V1 (A-PLMv1) model [26] is pre-trained using translation language modeling and MLM, which is trained on 1 billion distinct queries and 266 million parallel translations.
- Amazon in-house PLM V2 (A-PLMv2) model is obtained by continual pretraining public InfoXLM [7] on Amazon internal parallel data. InfoXLM is a multilingual and multimodal pre-trained model by Microsoft Research.

To generate the final embedding representation from different encoders, we explored using <CLS> special token embeddings and average pooling in our experiments.

4.3. Similarity Calculation Layer

We calculated the similarity between the two generated embeddings using three methods: cosine similarity, MLP, and a combination of the two. Details of these methods are provided in Appendix C Figure 7.

4.4. Supervised Learning

4.4.1. Pointwise Training

- $y_{ij} \to (0, 1]$, when q_i and c_{ij} have co-purchased products.
- $yij \to 0$, when q_i and c_{ij} have no co-purchased products.

During training, we artificially created negative pairs (q_i, c_{ij}, 0) where there was no co-purchase for q_i and c_{ij}, and mixed them with positive pairs (q_i, c_{ij}, y_{ij}), and trained a model on the label. During inference, we use the model to rank a list of candidates with the input form [(q_i, q_j)]. In this study, we use two different loss functions:

- Binary Cross Entropy Loss (BCE) [27]:

$$L_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N} y_{ij}log(\hat{y_{ij}}) + (1 - y_{ij}) \cdot log(1 - \hat{y_{ij}}) \tag{4}$$

- Mean Square Error Loss (MSE):

$$L_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_{ij} - \hat{y_{ij}})^2 \tag{5}$$

4.4.2. Negative Contrastive Learning

To mine hard negative training pairs, we adopted Approximate nearest neighbor Negative Contrastive Learning (ANCE) [28] on our representation model to generate a set of negative candidates for a given query q_i. We denote Q_i to include one positive candidate q_i^+ and $K - 1$ negative candidates $\{q_{i1}^-, q_{i2}^-, ...\}$. Then we use infoNCE [29] loss to optimize the negative log probability of identifying the positive sample q_i^+ amongst noise samples.

$$\mathcal{L}_{\text{NCE}} = -\frac{1}{N}\sum_{t=1}^{N} \log \frac{\exp(y_i^+)}{\sum_{t=1}^{K} \exp(y_{it})} \tag{6}$$

4.5. Application: Product Search Ranking

Two Stage Q2Q Pipeline CE models generally outperform BE models, but the computational cost increases significantly. For example, given 100 million search queries and 100 million historical candidate queries, the model will be called $100M^2$ times and will take 100 million GPU-days. With an optimal 10x inference speed and 1000 GPUs, it will take 10,000 days to finish. So it is not practical to use it on a very large set of query candidates like the Amazon search scenario, where the number of queries to be ranked is on a million scale. To design an end-to-end framework, we employ a two-stage procedure (Figure 3): retrieval and reranking.

For the first stage, we select the best representation-based model and run every non-tail query through it to generate embedding. We built an ANN graph for fast retrieval. Here we choose to use PECOS-HNSW [30], a graph-based ANN library for large-scale vector-similarity search that achieves state-of-the-art performance on ANN benchmark evaluations, to index all these embeddings. For any given input query embedding, the similarity computation is conducted using the inner product.

In the second stage, after each query retrieves its top k (k = 300 in our experiments) most similar queries from the previous stage, we deploy the best performing interaction-based model to rerank the retrieved candidates and output similarity scores.

Prior Calculation To improve search ranking using the Q2Q model output, we designed a method to augment tail queries where behavior signals are sparse with prior scores. Prior score measures the initial likelihood of an event before observing any data, which is a key concept in Bayesian methods that are commonly adopted in combination with observed data to make predictions. For a given query q, the Q2Q model finds q's similar queries $C_q = \{c_1, ..., c_m\}$

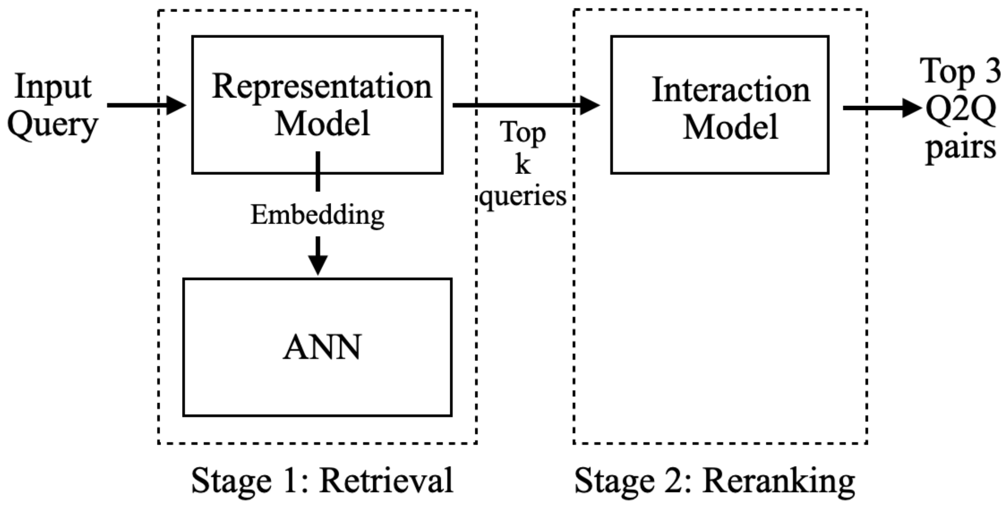

Figure 3: Retrieval-Reranking Pipeline

sharing similar customer actions, then build the $Prior$ with C_q's behavior signals. For query q and a related product p, we use $H(q,p)$ to represent their history signal score[2]. Then we have $Prior(q,p)$ defined as:

$$Prior(q,p) = \frac{1}{|C_q|} \sum_{i \in [1,m]} H(c_i, p) \tag{7}$$

To calculate the final feature $F(\cdot)$, we combine the $Prior(\cdot)$ with $H(\cdot)$:

$$F(q,p) = \alpha \cdot H(q,p) + (1-\alpha) \cdot Prior(q,p) \cdot \beta \tag{8}$$

where α is defined as:

$$\alpha = tanh \left[\frac{min(\gamma, I_{q,p})}{min(\gamma, \max_{p} I_{q,p})} \right] \tag{9}$$

$I_{q,p}$ denotes the number of impressions for product p under query q. γ is a constant to cap the very large numbers. We chose 10,000 here, and it depends on different application traffic. So α is computed from query-product impressions. The more query-product impressions, the higher α will be. It is used to balance the weight between observed historical signals and prior scores. We introduce β as the confidence rate used to further adjust the weight of prior.

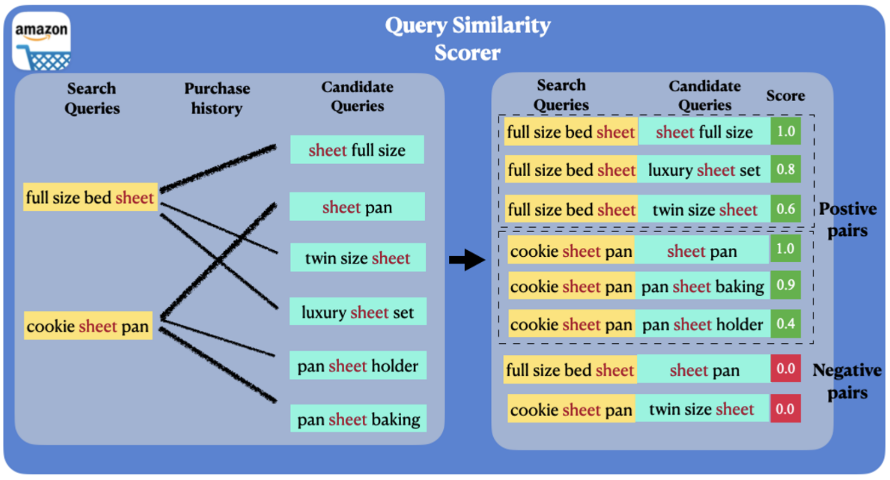
Query Similarity Scorer
Search Queries
Purchase history
Candidate Queries
full size bed sheet
cookie sheet pan
sheet full size
sheet pan
twin size sheet
luxury sheet set
pan sheet holder
pan sheet baking
Search Queries
Candidate Queries
Score
full size bed sheet | sheet full size | 1.0
full size bed sheet | luxury sheet set | 0.8
full size bed sheet | twin size sheet | 0.6
cookie sheet pan | sheet pan | 1.0
cookie sheet pan | pan sheet baking | 0.9
cookie sheet pan | pan sheet holder | 0.4
Postive pairs
full size bed sheet | sheet pan | 0.0
cookie sheet pan | twin size sheet | 0.0
Negative pairs

Figure 4: Overview of query pairs scoring schema

Table 2
Training Data Example of 'goya lady fingers'

input query	candidate query	Number of co_purchase	Number of union purchase	Number of minimum purchase	Jaccard similarity	Overlap similarity	Target
goya lady fingers	lady fingers for tiramisu prime	9	42	12	0.214	0.75	0.161
goya lady fingers	lady finger cookies for tiramisu	8	34	12	0.235	0.67	0.157
goya lady fingers	ladyfinger cookies	8	58	12	0.138	0.67	0.092
goya lady fingers	sponge fingers biscuit	4	18	10	0.222	0.4	0.088
goya lady fingers	lady fingers for trifle	4	18	10	0.222	0.4	0.088

5. Experiments and Evaluation

5.1. Data Preparation

We utilized the anonymized Amazon search logs for our experiments. Compared to click signals, purchase signals are more sparse but of higher quality. To reduce the training data noise, we considered query similarity based on the customer's purchase signals only. And we aggregated at the (query, product) tuple across all search sessions recorded over a one year period. For example in search log, if a user searches "harry potter", then all the returned products from the search would be listed out as separate rows. And each row contains the elementary metrics associated with a query-product impression and its subsequent actions (click, add-to-cart, purchase). The intuition is that if two different queries lead to purchases of the same product, these queries are likely to represent similar customer intentions.

Figure 4 illustrates an example of query-pairs scoring overview. We have "full size bed sheet" and "cookie sheet pan" as input queries with six candidate queries. For "full size bed sheet", three of them have the same product purchase history. The thicker the line, the more co-purchases there are between them. The connected queries are used as positive samples and are assigned a

[2] $H(q, p)$ is a weighted combination of clicks, adds and purchases of the (q, p) pair, normalized by the sum of its impressions and a query-level constant

similarity score. Similarly, for "cookie sheet pan", there are three queries with co-purchased products. Using this method, we build a group of positive samples for the query. For negative samples, we randomly pair two queries that do not have any co-purchased product history.

To improve the recall at the retrieval stage, we adopted noise contrastive learning to mine hard negative pairs from ANN retrieval results using our initial representation model.

Data Example

Table 2 presents the top queries that have similar purchase histories with the "goya lady fingers". "Lady finger" is a type of sponge cake biscuits and "goya" is the brand name. In the "candidate query" column, we observe that the top-ranked queries include "lady fingers for tiramisu prime", "ladyfinger cookies". It is not straightforward to consider "goya lady finger" and "tiramisu cookies" as similar queries at a lexical level. But rich behavior signals show the two queries share similar purchased products, thus teaching the model to learn this semantic level similarity.

Scalability Challenge

When the search logs have billions of unique query product pairs, using the Cartesian product directly on all the queries would yield over a trillion query pairs, resulting in an out-of-memory issue. To reduce enormous communications between nodes and fully leverage the parallel computation power in Spark, we aggregate the query-product pairs at the product level and then enumerate query pairs from all the corresponding products. The greater the number of related queries for a product, the longer the enumeration time. Thus, we implemented the divide-and-conquer strategy for this task. We bucketed the product based on the number of corresponding queries and then triggered the enumeration process. However, there are some very popular products with a high number of related queries. We first did a sampling of queries, then triggered this process to save computation time.

Final Dataset

Table 3 shows the details of the training, validation, and test data. For the training data, there were 1.67 billion query pairs with 27.9 million unique input queries. For validation data, there were 687,300 pair queries and 22,910 unique input queries. We have two test data sets. A small test set can be used to quickly test model performance as well as the best checkpoint selection. A large test set is needed to evaluate the performance of a large pool and generate stable scores for model comparisons.

Table 3
Datasets used in this project

	Number of rows	Unique query1
Training	1.67 billion	27.9 million
Validation	687,300	22,910
Test (Small)	7,083	284
Test (Large)	21 million	379,358

Table 4
Bi-encoder models. In Encoder finetuned column, T denotes True and F denotes False.

Model name	**Model config**		**NDCG@3**	**Gain**
	Encoder backbone	Encoder finetuned		
Q2Q	random init	T	0.5713	-
bBv1	BERT-base	F	0.6748	10.35%
bBv2		T	0.6912	11.99%
bSTv1	Sentence-BERT	F	0.6650	9.37%
bSTv2		T	**0.7275**	15.62%
bAv1	A-PLMv1	F	0.6861	11.48%

Table 5
Cross-encoder models. For interaction layers, all use "MLP+cos" means a concatenation of MLP output (a score) and cos-sim output, followed by a 2-to-1 layer.

Model name	**Model config**			**NDCG@3**	**Gain**
	Encoder backbone	Encoder finetuned	Interaction Layers		
cBv1	BERT-base	F	MLP	0.6319	-
cBv2		F	MLP+cos	**0.6395**	0.76%
cSTv1	Sentence-BERT	F	MLP	0.6780	4.61%
cSTv2		F	MLP+cos	**0.7286**	9.67%
cBv3	BERT-base	T	MLP+cos	0.7122	8.03%
cSTv3	Sentence-BERT	T	MLP+cos	0.7450	11.31%
cAv2	A-PLMv2	T	MLP+cos	**0.7968**	16.49%

Table 6
Comparison of cross-encoder teacher and the distilled bi-encoder student.

Note	**encoder**	**finetuned on**	**architecture**	**loss**	**train**	**inference**	**NDCG@3**	**row**
cAv2 (teacher)	A-PLMv2	hard label	CE	MSE	-	-	0.7968	1
benchmark1	Sentence-BERT	zero-shot	BE	-	-	cos	0.6395	2
benchmark2	Sentence-BERT	hard label	BE	BCE	cos	cos	0.6875	3
student1	A-PLMv2	cAv2 score	BE	BCE	MLP	cos	0.7240	4
student2	A-PLMv2	cAv2 score	BE	BCE	MLP	MLP	**0.7780**	5

5.2. Offline Evaluation Metric

In this study, we use Recall@100, 1000 and NDCG@3 (Normalized Discounted Cumulative Gain [31]) to evaluate the model's performance. Recall is to measure the retrieval stage performance for the representation-based model. NDCG is to measure the reranking stage performance, and we chose 3 here to align with our downstream applications for computing the prior score. In our task, the rank of the relevant queries is more important than the actual prediction score. So we choose NDCG as it accumulates gain from the top of the query list to the bottom, with the gain of each result discounted at lower ranks. The metric ranges from 0 to 1.

5.3. Offline Evaluation

We trained models using different combinations of model components, including BE and CE structure in the encoder layer, and MLP and Cosine similarity in the similarity calculation layer.

Table 7
Models Recall with different techniques

Note	Model name	Recall@100	Recall@1000
baseline	bi-q2q	0.59	0.92
+weight	w-bi-q2q	0.64	0.92
+ANCE	ANCE-w-bi-q2q	0.78	0.94

BE Model Comparisons Table 4 shows the NDCG@3 metric on BE models with cosine similarity for interaction. We found that 1) Using the Q2Q model train from scratch as a baseline, the models using PLMs as the backbone have lifted the NDCG range from 937 bps to 1562 bps. 2) Using our behavioral-driven query pairs to fine-tune PLMs improved the performance, ranging from 2.4% to 9.4% (compare bBv1 vs. bBv2 and bSTv1 vs. bSTv2). Specifically, the best BE model, bSTv2 outperformed the vanilla Q2Q model by 27.4%.

CE Model Comparisons Under the CE model (Table 5) structure, we first frozen the encoder and tested different interaction layers. We found combining MLP and cosine similarity, followed by a 2-to-1 layer, is better than MLP or cosine similarity individually. And the best CE model, cAv2 (using Amazon's in-house A-PLMv2), outperformed cBv1 (using BERT-base) with an improvement of 26%. In addition, our evaluation shows that the model training using A-PLMv2 is not sensitive to loss (MSE, BCE) or pooling choices (avg, cls).

Distillation We further evaluated whether we could improve BE model performance by distilling the best CE model, cAv2.Instead of using hard labels from the original training data, we use cAv2 to generate soft labels for student models. Table 6 shows that the BE student1 outperforms the benchmark2 fine-tuned with hard labels by 300 bps (rows 3, 4), but still trails the CE teacher by 700bps (rows 1, 4). With one more linear layer to allow more embedding interaction, the model narrows the gap and trails the CE teacher by 180 bps (rows 1, 5). As expected, fine-tuning public Sentence-BERT using our label improved the performance by 7.5%. And replacing cosine similarity with MLP brings a 7.5% gain (rows 4, 5) on the distilled student model.

Model Recall In Table 7, the recall@100 for the initial trained representation model is 0.59. To improve the model recall, we first add the label as weight in training, and the recall increases to 0.64. After we introduce the ANCE technique with the negative queries coming from the model's retrieval phase, the recall at 100 reaches 0.78 with 32% improvement compared with baseline.

5.4. Ablation Studies

To better understand the impact of the backbone models, pretraining, finetuning, and distillation. We conducted a series of evaluations on our models trained with BE architecture, measured by NDCG@3. We measured whether Amazon's in-house PLM provided additional benefits over its parent, InfoXLM. In Table 8, specifically, we observed that pretraining with Amazon

Table 8
Ablation Study

Results in this ablation study all use these configs: BE architecture, train with one MLP layer and BCE loss					row
Note	**encoder**	**finetuned on**	**interaction (inference)**	**NDCG@3**	1
initial backbone	InfoXLM	zero-shot	cos	0.63689	2
+ pretrain	A-PLMv2	zero-shot	cos	0.67044	3
+ finetune	A-PLMv2	hard label	cos	0.71784	4
+ distillation	A-PLMv2	cAv2	cos	0.72396	5
benchmark	Sentence-bert	zero-shot	cos	0.6395	6
benchmark	Sentence-bert	hard label	cos	0.6875	7
CE teacher	A-PLMv2	hard label	-	0.7968	8
other configs	A-PLMv2	cAv2	MLP	0.77803	9
	InfoXLM	hard label	cos	0.64157	10
	InfoXLM	hard label	MLP	0.68549	11
	A-PLMv2	hard label	MLP	**0.78599**	12

query/product datasets brings 340 bps lift (rows 2, 3) to 1000 bps lift (rows 11, 12). For Q2Q tasks, when find-tuned on Sentence-BERT and A-PLMv2 with hard label, A-PLMv2 has 303bps lift over Sentence-BERT. On the other hand, fine-tuning the behavior signal brings an additional 470 bps lift (rows 3, 4). Distillation brings an additional 60 bps lift (rows 4, 5). Surprisingly, we found the simple MLP layer on top of the BE layer could bring the model to similar performance as the best CE model with negligible differences of 11 bps (rows 8, 12).

5.5. Production Experiment Results

Using the two-stage Q2Q pipeline, we conducted an online A/B test on the Amazon US website for one week on 100 million search sessions at a 5% level of significance. We use the best BE model, bSTv2, in the retrieval stage and the best CE model, cAV2, in the reranking stage. The model yielded significant revenue wins and significantly reduced search defects, along with other search-related metrics improvements, including the number of searches increasing by 0.03%, search page clicks increasing by 0.08%, search reformulation rate decreasing by 0.03%, and the average click depth decreasing by 0.05%.

In particular, this experiment shows a stronger improvement in clothing and fashion-related shopping categories across all the metrics, than other categories. The stats show that these categories have a lower conversion rate than electronics and kitchens. We conjecture that these significant wins are due to the fact that the prior scores powered by the Q2Q model provide customers with more related choices to browse and select. With the Q2Q augmentation, we are able to improve this capacity and thus gain more customer purchases.

6. Conclusion

In this study, we present a query similarity prediction framework that leverages behavior data. We first mine the query pairs from the yearly aggregated logs and design the training labels that can approximate their similarities. These query pairs could go beyond the semantic level when fused with domain-specific knowledge. For example, the behavior data could link "cheap" with "amazon basic" and have better domain-specific token representations like "amazon", "prime", "gift card", and "brands". Then, we explored various model components and compared their performance on both public and Amazon in-house PLMs. The model fine-tuned on Amazon's in-house PLM has improved 27.4% over the BERT baseline. To improve ranking quality in e-commerce, we designed an end-to-end pipeline to utilize the model output to build prior behavior features. The online experiments conducted in the US showed significant improvements in search click rates and defect reduction.

Our work provides a practical solution to leverage similar queries to improve search ranking in e-commerce settings. This study emphasizes the value of combining customer behavior signals, which contain precise and up-to-date knowledge, with the general knowledge provided by PLMs. And we selectively combined them for different applications with different latency requirements. While CE models generally exhibited superior performance to BE models, we found that with distillation techniques and combining MLP on top of the best-performing BE model, we could achieve similar performance as a CE model. This combination not only leads to better precision in downstream applications but also facilitates online deployment.

7. Acknowledgements

This work builds on top of the work that has been done at Amazon Search. Special thanks to Wei-Cheng Chang, Jyun-Yu Jiang, Choon-Hui Teo, Saeedeh Salimianrizi, Christopher Fayoux, Mackie Hembrador, Anurag Shiv, Kang Wang, Ram Kandasamy, Ruirui Li, Haoming Jiang, Yifan Gao, Qingyu Yin, Cuize Han, Zhengyang Wang, Chen Luo, Xiaojie Wang, Ziqi Zhang and Hsiang-Fu Yu.

References

[1] E. Agichtein, E. Brill, S. Dumais, Improving web search ranking by incorporating user behavior information, in: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2006, pp. 19–26.

[2] P. Covington, J. Adams, E. Sargin, Deep neural networks for youtube recommendations, in: Proceedings of the 10th ACM conference on recommender systems, 2016, pp. 191–198.

[3] S. Maji, R. Kumar, M. Bansal, K. Roy, M. Kumar, P. Goyal, Addressing vocabulary gap in e-commerce search, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 1073–1076.

[4] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in Neural Information Processing Systems (NIPS), 2013, pp. 3111–3119.

[5] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[6] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, 2019. URL: http://arxiv.org/abs/1908.10084.

[7] Z. Chi, L. Dong, F. Wei, N. Yang, S. Singhal, W. Wang, X. Song, X.-L. Mao, H. Huang, M. Zhou, InfoXLM: An information-theoretic framework for cross-lingual language model pre-training, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Online, 2021, pp. 3576–3588. URL: https://aclanthology.org/2021.naacl-main.280. doi:10.18653/v1/2021.naacl-main.280.

[8] M. Zhang, Y. Wu, R. Rustamov, H. Zhu, H. Shi, Y. Wu, L. Tang, Z. Zhang, C. Wang, Advancing query rewriting in e-commerce via shopping intent learning, in: Proceedings of the 2022 ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA, 2022, p. 9.

[9] L. Kumar, S. Sarkar, Neural search: Learning query and product representations in fashion e-commerce, 2021. arXiv:2107.08291.

[10] T. Bayes, An essay towards solving a problem in the doctrine of chances, Philosophical Transactions of the Royal Society of London 53 (1763) 370–418.

[11] C. D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.

[12] Y. Papakonstantinou, V. Vassalos, Query rewriting for semistructured data, ACM SIGMOD Record 28 (1999) 455–466.

[13] S. Yu, J. Liu, J. Yang, C. Xiong, P. Bennett, J. Gao, Z. Liu, Few-shot generative conversational query rewriting, in: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, 2020, pp. 1933–1936.

[14] S.-C. Lin, J.-H. Yang, R. Nogueira, M.-F. Tsai, C.-J. Wang, J. Lin, Multi-stage conversational passage retrieval: An approach to fusing term importance estimation and neural query rewriting, ACM Transactions on Information Systems (TOIS) 39 (2021) 1–29.

[15] A. Mandal, I. K. Khan, P. S. Kumar, Query rewriting using automatic synonym extraction for e-commerce search, in: eCOM@ SIGIR, 2019.

[16] H. Lu, Y. Xu, Q. Yin, T. Cao, B. Aleksandrovsky, Y. Song, X. Fan, B. Yin, Unsupervised synonym extraction for document enhancement in e-commerce search (2021).

[17] Z. Tan, C. Xu, M. Jiang, H. Yang, X. Wu, Query rewrite for null and low search results in ecommerce, in: eCOM@SIGIR, 2017. URL: https://api.semanticscholar.org/CorpusID:59528277.

[18] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, arXiv preprint arXiv:1409.3215 (2014).

[19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners (2019).

[20] S. Hofstätter, J. Chen, K. Raman, H. Zamani, Fid-light: Efficient and effective retrieval-augmented text generation, 2022. arXiv:2209.14290.

[21] M. Dehghani, H. Zamani, A. Severyn, J. Kamps, W. B. Croft, Neural ranking models with weak supervision, 2017. arXiv:1704.08803.

[22] H. Cui, J.-R. Wen, J.-Y. Nie, W.-Y. Ma, Probabilistic query expansion using query logs, in: WWW, ACM, 2002, pp. 325–332.

[23] U. Ozertem, O. Chapelle, P. Donmez, Learning to suggest: A machine learning framework for ranking query suggestions, in: Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, 2012, pp. 25–34.

[24] S. Fujita, G. Dupret, R. Baeza-Yates, Semantics of query rewriting patterns in search logs, in: Proceedings of the fifth workshop on Exploiting semantic annotations in information retrieval (ESAIR '12), Association for Computing Machinery, New York, NY, USA, 2012, pp. 7–8. doi:10.1145/2390148.2390153.
[25] R. Baeza-Yates, C. Hurtado, M. Mendoza, Query recommendation using query logs in search engines, in: EDBT, Springer, 2004, pp. 588–596.
[26] H. Jiang, T. Cao, Z. Li, C. Luo, X. Tang, Q. Yin, D. Zhang, R. Goutam, B. Yin, Short text pre-training with extended token classification for e-commerce query understanding, 2022. arXiv:2210.03915.
[27] C. E. Shannon, A mathematical theory of communication, The Bell System Technical Journal 27 (1948) 379–423.
[28] L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. Bennett, J. Ahmed, A. Overwijk, Approximate nearest neighbor negative contrastive learning for dense text retrieval, 2020. arXiv:2007.00808.
[29] A. van den Oord, Y. Li, O. Vinyals, Representation learning with contrastive predictive coding, 2019. arXiv:1807.03748.
[30] H.-F. Yu, K. Zhong, I. Dhillon, Pecos: Prediction for enormous and correlated output spaces, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2020.
[31] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of ir techniques 20 (2002) 422–446. doi:10.1145/582415.582418.

A. Bi-encoder Q2Q model

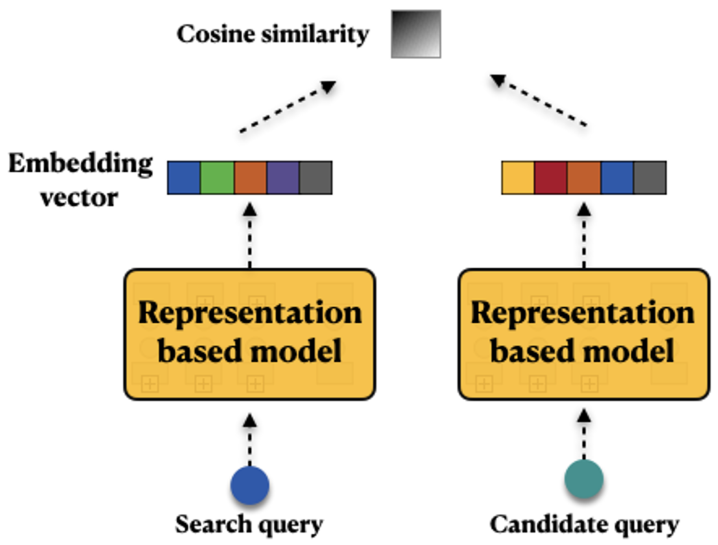

Figure 5: Bi-encoder Q2Q model

B. Cross-encoder Q2Q model

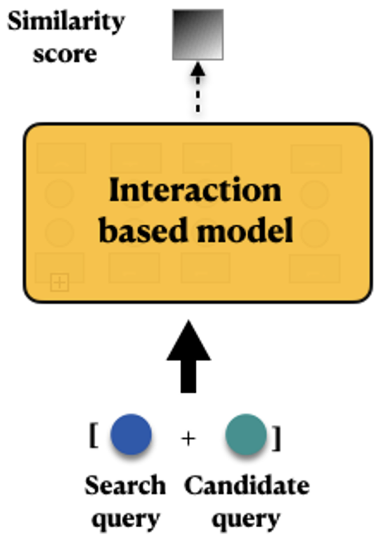

Figure 6: Cross-encoder Q2Q model

C. Similarity layer

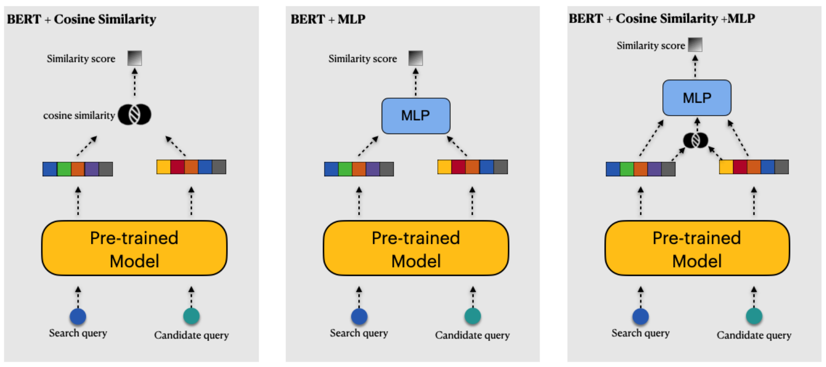

Figure 7: 3 approaches to calculate similarity layer