PSimBERT: Query Phrase Expansion for Low Recall Vocabulary Gap Queries

Rohan Kumar*
LTI, CMU
rohankum@cs.cmu.edu

Surender Kumar
Flipkart
surender.k@flipkart.com

Samir Shah
Flipkart
samir.shah@flipkart.com

ABSTRACT

E-commerce search plays a pivotal role in online shoppers' product finding journey. Different users articulate the same purchase intent in different ways. The vocabulary of search queries can be different from the product catalog, even when the intended products are the same. Hence, query expansion is used to match the query intent with the catalog vocabulary and increase the set of matching products. We present a practical and novel BERT-based expansion system that leverages phrase expansions to improve the recall performance of such vocabulary gap queries. Our system meets strict user-path latency requirements for online deployment while giving good results. We conduct offline and online experiments and show improvements over the existing statistical and neural methods. Specifically, we see a 3.7% reduction in null searches and an improvement of 1.5% in clickthrough rate and 0.5% in units on tough tail queries.

CCS CONCEPTS

• **Computing methodologies → Neural networks**; • **Applied computing → Online shopping**; • **Information systems → Query reformulation**.

KEYWORDS

query expansion, vocabulary gap, transformers, e-commerce

ACM Reference Format:
Rohan Kumar, Surender Kumar, and Samir Shah. 2022. PSimBERT: Query Phrase Expansion for Low Recall Vocabulary Gap Queries. In *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR eCom'22).* ACM, New York, NY, USA, 5 pages.

1 INTRODUCTION

On e-commerce platforms, users with varying levels of literacy and articulation abilities articulate their queries with text different from what the sellers upload in the product catalog. Both sellers and users can use colloquial and technical terms to describe a product, although sellers are more likely to use the technical terms. This results in a low performing (low or no results) search problem arising from a phenomenon known as vocabulary or articulation gap. For example, a query like "avengers half-pant" refers to the product listed as "avengers shorts" and "pregnancy dress" listed as "maternity gown" in the catalog by the sellers. The problem of articulation gap is more pronounced in torso, tail query segments (last two tertiles of query set divided into three equal parts by query frequency) and once only queries [5]. This can be addressed using either Query Rewriting or Query Expansion. In Query Rewriting, the whole query is replaced with one or more replacements which better correspond to catalog vocabulary while avoiding user intent drift. For example, rewriting "sugar checking machine" as "blood glucose monitor" or "diabetes test strip". Multiple queries, however, lead to heavier loads on the search engine index due to multiple passes for each alternate query, and thus using one rewritten query is preferable in a user path production setup. Another way is to expand the query by adding more keywords/tokens to the original query (optionally) along with a Boolean expression like "(sugar or diabetes) checking machine". From an analysis done by human judges on a random sample of user queries, we found that many times replacing one or two phrases in the query is sufficient to generate well-performing queries. This approach is more compute-friendly as it involves a single pass on the search index.

Pseudo-relevance feedback [14] is one of the most common ways to expand a query by first fetching the documents against the given query. This is followed by extracting top keywords from these documents to append to the original query and retrieve the final list of products. This does not work for Null Searches (where no products are fetched in the original query), which forms a significant volume of user searches. It is computationally expensive since related terms from retrieved documents rely on an initial retrieval, and hence not practical in the user path. Further, pseudo-relevance feedback operates in the catalog vocabulary space and thus is not able to capture the variations of the user vocabulary. This may also lead to query drift [16], as unrelated terms from products / documents are added to the user query. To address this shortcoming, we use the user query space to find the relevant and well-performing expansion terms.

Rewriting an original query to an alternate query is typically done by first identifying a set of well-performing head [5] queries (based on individual query volume and click through rates) and then mapping an articulation gap query (tail [5] query) to the most similar query from this well-performing set [12].

As shown in [10] and also confirmed in our dataset, the coverage of replacement head query for a tail articulation gap query drops drastically in tail and once only queries while phrase substitutions cover a much larger range of queries. For example, with a certain tail query to head query mapping technique, we generate (offline) only 20K pairs from one month's data of tail-to-head queries while we receive over 15 million unique queries a day.

*Work done while at Flipkart

SIGIR eCom'22, July 15, 2022, Madrid, Spain

Hence in this work, we focus on query expansion-based approach that chunks a query into phrases, identifies the least performing phrase(s) and replaces these by better performing ones (see Fig. 1). The replacement phrase is found using an domain-adapted and fine-tuned BERT-based model that finds the contextual synonyms of the target phrase(s). This model is also optimized for online invocation in the production system. We report a significant improvement in user metrics and null searches.

- We propose a novel BERT-based query phrase expansion system.
- We successfully adapt the similarity model for the e-commerce domain with large-scale domain-adaptive pretraining and task-specific fine-tuning.
- We performed multiple offline and online experiments that show the efficacy of the proposed system on real-world data and search engine.

In the next sections, we describe related work, followed by our approach, optimizations to deploy the model at web scale, and offline and online experiments with performance metrics.

2 RELATED WORK

Ruthven et al [16] demonstrated the limitations of Pseudo-relevance feedback based query expansion method which primarily operates in the document space. Previous work by Maji et al [12] generated entirely new queries by learning a query to query similarity deep neural model which performs well but is computationally costly to run at scale. Statistical approach by Jones et al [10] overcome these limitations by switching to the user query space and used query reformulations by the users. They also found phrase based substitutions have much better coverage than query to query substitution.

Besides the co-occurring queries, other implicit user feedback like click-through rate (CTR) [3] and co-click based query graphs [2], [7] have been used to rewrite queries. However, as mentioned, articulation gap queries are typically infrequent and low performing. This results into insufficient implicit user feedback and thus limiting the efficacy of these feedback based methods.

While the work of Jones et al. [10] and the derived works rely on statistical co-occurrence-based phrase-to-phrase similarity, we develop a deep learning (BERT [4]) based semantic phrase-to-phrase similarity model. Zamani et al [20] create word embedding using pseudo-relevance feedback. Our contextual embedding approach is superior to embedding-based relevance models because of 3 reasons. First, due to pseudo relevance, their vocabulary space is still restricted to the e-commerce seller catalog and thus does not capture the user vocabulary. Second, the embeddings learned are context free due to which a term like "bank" will have the same embedding for both the queries "bank of a river" and "nearest bank to deposit money". And finally being a 2-stage retrieval model, pseudo-relevance feedback in the user path is computationally expensive and thus is rarely used in practice. Our approach overcomes these using bidirectional context for a given phrase and hence can further differentiate between queries like "milk chocolate" and "chocolate milk" due to bidirectional nature of BERT [4], which builds a deeper pretrained Transformer [18] variant using only encoders.

3 PROPOSED SYSTEM

Given a search query, our goal is to selectively expand portions of the query with synonymous terms to improve recall performance. We use a Lucene [1] based search index to retrieve the matching products for an input user query (and it's augmented variants). Our proposed system consists of the following parts:

3.1 Query Segmentation

A query chunker segments the query into "constituent phrases", some of which can then be chosen for expansion. Phrases are common word combinations with meaning at least slightly different from that of individual words. These word combinations can have semantic meaning/definition of their own, contrasting from meanings of individual words, for example, 'back cover', 'steve madden'. We explore grammar-based linguistic rules or heuristics from NLP literature but found the insufficient since search queries typically aren't complete, well-formed sentences. Instead, we extract n-gram phrases from user queries based on thresholds on co-location information. We denote the phrase set thus created by P. To segment the query, we scan the query tokens starting from the left and recursively take the longest segment that exists in P, similar to prior work [13].

3.2 Expansion Phrase Identification

Once a query is segmented into phrases, we identify the phrase(s) that can cause a vocabulary gap. Here we assume a phrase which has poor recall is the cause of the vocabulary gap and choose it for expansion. Therefore, we create a set of metrics N to store the number of products historically matched for phrases along with their historical CTRs. We conjecture that by expanding on phrases with a low or null matching set or low CTR, we are more likely to increase recall in the expanded query.

3.3 Phrase Similarity

After identifying a phrase to be replaced in the input query, we search for its replacement phrase which is synonymous in the context of the query. We formulate this problem of finding expansions as a mapping problem (as opposed to a generation problem) to simplify the task. In this formulation, we need to find the nearest neighbors from a fixed set to a given input phrase. We perform Maximum Inner Product Search (MIPS) on dense vector representations of these phrases. To model contextual synonyms, we use the BERT-based model [4, 17, 19] to identify the top-k most similar phrases to the input phrase, from a set of well-performing phrases called the mapping phrase set M.

We obtain BERT representations for the input phrase p and all the phrases in M and choose the top-k based on the dot product scores between p and each $m_i \in M$. BERT-style models typically produce subword representations which are combined to produce word representations. I our approach, the whole query is passed to the model and we get the phrase representation by concatenation of representations of the first and last words of the phrase, which is shown to work well in [9]. All other token representations are discarded, but it is still important to use the whole query in the model so we can get the representation of the phrase in the context of the query.

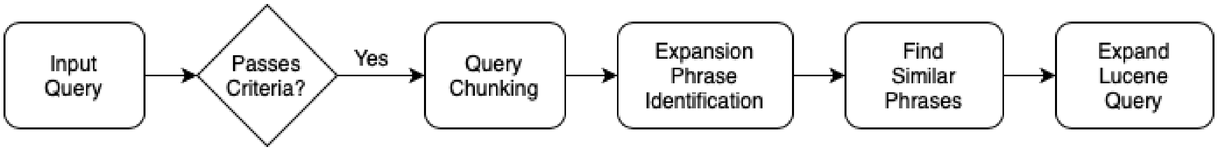

Figure 1: Simplified flow of the proposed system.

We describe the details of the creation of sets P, N, M in section 4.1.

3.4 Phrase Similarity Model Training

To obtain good representations of phrases from the BERT model, we perform domain-adaptive pretraining and task-based fine-tuning which are shown to work well in [8]. For domain-adaptive pretraining we use in-domain query chains (search queries appearing consecutively in the same user search session). We find that this leads to better representations as compared to just fine-tuning.

In the pretraining phase, the BERT-base-uncased model checkpoint is pre-trained on search queries optimizing for the masked language model (MLM) objective.

For the fine-tuning phase, the model is trained on the sentence-pair classification task using query pairs and a binary label. We train on sentence(query) pairs instead of phrases to capture contextual synonyms. We call our system PSimBERT(**Phrase****Sim**ilarity with **BERT**).

4 EXPERIMENTS

4.1 Data Sets

In this section, we describe the dataset along with the generation details of the sets Phrase Set P, Metrics Set N, and Mapping Set M. To create these 3 sets, user queries along with the size of their matching product set from one month's logs are used.

4.1.1 Phrase Set P Creation. For the purposes of our experiments, we restrict phrases to bigrams. For all adjacent word pairs in a query, we calculate co-location measures [11] such as pointwise mutual information, log-likelihood ratio, token co-occurrence counts and conditional probabilities and apply thresholds to get the set P. of 1M bigram phrases.

To evaluate the phrase set, we draw a random sample of ~1000 queries stratified on frequency volume buckets and chunk them using the strategy described in section 3.1. Human raters manually evaluated the set of ~550 identified phrases and found that ~85% of the identified phases were good/valid phrases for our phrase expansion task.

4.1.2 Metrics Set N Creation. To create the metrics set N, we segment the queries using the technique described in Section 3.1. For each phrase segment, we calculate matching set size by taking an average of matching set sizes of all queries in which the phrase occurs.

4.1.3 Mapping Set M Creation. To create this set, we chunk the queries according to the criteria described in section 3.1. For each chunk, we identify up to k queries in which it occurs most frequently. Choosing multiple queries per phrase provides different potential contexts for the phrase. This set of phrase-query pairs is called the mapping phrase set M. In our experiments, we choose the value of k as 5 and the set M consists of ~120k items.

Padding	Knowledge Distillation	Dynamic Quantization	Rutime (ms)
-	-	-	37.8
✓	-	-	33.2
✓	✓	-	16.9
✓	✓	✓	7.84

Table 1: Single-example avg. inferenece runtimes of variants of the PSimBERT model with performance optimizations

4.2 Experimental Setup

For model training, we used GPUs (NVIDIA Tesla V100-SXM2) while for production inference in the user path, we used CPUs (Intel x_86, 64 bit, 2.1GHz, VM with KVM Hypervisor). For in-domain pretraining, we used the same hyperparameter configuration as in the original BERT-base model. For fine-tuning, we set the learning rate to 0.0001 and all model parameters were updated during the training. We used the PyTorch [15] and HuggingFace Transformers [6] machine learning libraries for our model development.

4.3 Model Training

For phrase similarity PSimBERT model pre-training phase, the queries are obtained from a month's search logs after filtering for extreme tail ones to create a set of ~57M queries. We experimented with pretraining on both MLM and the next sentence prediction tasks (i.e. next query prediction in a user session from same product category) but found the performance(recall@5) to be the same as using just MLM, which we used in subsequent experiments.
For the fine-tuning phase of the sentence pair classification task, the data are labeled by human labellers. Domain experts identify vocabulary gap queries from a random sample of low-performing queries and provide a well-performing ground-truth replacement. We then add a randomly sampled negative example for every positive pair to obtain a class-balanced labeled dataset with ~69k examples.

4.3.1 Model Inference Optimizations. The basic PSimBERT model gives reasonably good performance on the task. However, owing to its large size (and corresponding latency), it is computationally infeasible to run it online and still adhere to strict user-path latency constraints. To solve this, we use three techniques: First, we remove padding during inference since the model will only receive a single query per batch in user-path. Second, we apply knowledge distillation[17] thus reducing the number of layers by a factor of 2. We distil the pretrained version of the model tuned for e-commerce

k	Recall
1	77.60
2	82.60
3	85.39
5	88.20
10	90.00

Table 2: Recall at various values of K for phrase similarity judgements.

domain and then perform fine-tuning as described in the previous section. Finally, we further simplify model inference computation by applying dynamic quantization[19]. We perform an ablation of inference runtimes on a modern CPU using 4 (physical) cores the results of which are in table 1. We see reduction in runtimes with each technique. Since the mapping set M is precomputed offline, we also precompute BERT representations for all phrases in this set, so during inference we only need to compute representation for a phrase in the input query.

4.4 Production System Integration

Once we have the rewrite phrases, we need to retrieve products using these for the input query fired by a user. Instead of firing separate queries to the index, we construct a Lucene [1] query with a disjunction(OR) of rewrite phrases m_i. For example, the query "p_1 p_2 p_3" is expanded as "p_1 (p_2 OR m_i) p_3" where m_i is the closest phrase from the mapping set. This leads to better compute efficiency on the index. The overall P95 latency of the system is 38ms, which is within acceptable limits and allows us to deploy the model online (in user path).

4.5 Evaluation

4.5.1 Phrase Similarity Evaluation. For a given phrase identified for expansion, we evaluate the quality of top 10 most similar phrases according to the model on a random sample of 500 queries on a boolean scale of good or bad. Table 2 shows recall at various values of k. For latency reasons, we restrict to top-3 phrases for expansion.

4.5.2 Offline evaluation. Our current production system uses statistical query to query replacement dictionaries, as well as a query to query MaLSTM [12] model, which is superior to multiple complete query rewrite systems (refer [12] for more details). Query expansion is a recall-focused task. Hence, for offline evaluation, our human judges measured Recall@30 to account for inefficiencies of the precision-focused relevance ranking systems that are invoked by the recall layer. We observed that Recall@30 improved by +11% over the production system.

4.5.3 Online AB Experiment. We deployed the PSimBERT model to production for an online AB experiment on 10% of user traffic sampled randomly and ran it for two weeks at 5% significance level. The model was applied with an engagement criteria of expanding only low-recall (number of matching products < 2) and short (number of words < 7) queries. The experiment was run with the current production system as the control bucket. Table 3 shows the improvements of PSimBERT with reduced null searches and increased search query CTR. The reported numbers are across all queries. While null searches can be reduced by expanding with unrelated phrases and showing unrelated products, we see that conversion metrics such as CTR and cart adds also go up, which indicates that the model is returning relevant phrases.

Metric	Improvement
Null Searches	-3.7 %
CTR	1.5%
Cart Adds	0.3%
Units/Visitor	0.5%

Table 3: Results of the AB experiment. All metrics are statistically significant.

5 CONCLUSION

Search queries can lead to poor recall due to vocabulary gap. In this paper, we propose a novel BERT-based query expansion system to improve the matching set of products beyond the input query text. Our similarity model is adapted to the e-commerce domain with large-scale domain-adaptive pretraining followed by task-specific tuning. To deploy our model online, we perform various compute optimizations. We show the efficacy of our system on a real-world search system with online and offline evaluation.

ACKNOWLEDGMENTS

We thank our Search Semantics team at Flipkart and particularly Pooja Kushalappa and Kiran Penumarthi for their support with integration of the model in the serving systems and running the AB experiments.

REFERENCES

[1] 2005. Apache Lucene. (2005). https://lucene.apache.org/core

[2] Ioannis Antonellis, Hector Garcia-Molina, and Chi-Chao Chang. 2008. Simrank++: Query Rewriting through Link Analysis of the Clickgraph (Poster) *(WWW '08)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1367497.1367714

[3] Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. 2002. Probabilistic Query Expansion Using Query Logs *(WWW '02)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/511446.511489

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[5] Doug Downey, Susan Dumais, and Eric Horvitz. 2007. Heads and Tails: Studies of Web Search with Common and Rare Queries *(SIGIR '07)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1277741.1277939

[6] Thomas Wolf et al. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://www.aclweb.org/anthology/2020.emnlp-demos.6

[7] Bruno M. Fonseca, Paulo Golgher, Bruno, Berthier Ribeiro-Neto, and Nivio Ziviani. 2005. Concept-Based Interactive Query Expansion *(CIKM '05)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1099554.1099726

[8] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 8342–8360.

[9] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language?. In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*.

[10] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating Query Substitutions *(WWW '06)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1135777.1135835

[11] Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc.

[12] Subhadeep Maji, Rohan Kumar, Manish Bansal, Kalyani Roy, Mohit Kumar, and Pawan Goyal. 2019. Addressing Vocabulary Gap in E-Commerce Search *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 1073–1076. https://doi.org/10.1145/3331184.3331323

[13] Aritra Mandal, Ishita K Khan, and Prathyusha Senthil Kumar. 2019. Query Rewriting using Automatic Synonym Extraction for E-commerce Search.. In *eCOM@ SIGIR*.

[14] Christopher Manning and Prabhakar Raghavan. 2008. *Introduction to Information Retrieval*.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[16] Ian Ruthven. 2003. Re-Examining the Potential Effectiveness of Interactive Query Expansion *(SIGIR '03)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/860435.860475

[17] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[19] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8bert: Quantized 8bit bert. *arXiv preprint arXiv:1910.06188* (2019).

[20] Hamed Zamani and W. Bruce Croft. 2017. Relevance-Based Word Embedding *(SIGIR '17)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3077136.3080831