Graph-based Multilingual Product Retrieval in E-Commerce Search

Hanqing Lu
Amazon Search
luhanqin@amazon.com

Youna Hu
Amazon Search
ynhu@amazon.com

Tong Zhao
Amazon Personalization
zhaoton@amazon.com

Tony Wu
Amazon Search
tonywu@amazon.com

Yiwei Song
Amazon Search
ywsong@amazon.com

Bing Yin
Amazon Search
alexbyin@amazon.com

Abstract

Nowadays, with many e-commerce platforms conducting global business, e-commerce search systems are required to handle product retrieval under multilingual scenarios. Moreover, comparing with maintaining per-country specific e-commerce search systems, having a universal system across countries can further reduce the operational and computational costs, and facilitate business expansion to new countries. In this paper, we introduce a universal end-to-end multilingual retrieval system, and discuss our learnings and technical details when training and deploying the system to serve billion-scale product retrieval for e-commerce search. In particular, we propose a multilingual graph attention based retrieval network by leveraging recent advances in transformer-based multilingual language models and graph neural network architectures to capture the interactions between search queries and items in e-commerce search. Offline experiments on five countries data show that our algorithm outperforms the state-of-the-art baselines by 35% recall and 25% mAP on average. Moreover, the proposed model shows significant increase of conversion/revenue in online A/B experiments and has been deployed in production for multiple countries.

1 Introduction

Modern e-commerce search engines (Huang et al., 2020; Nigam et al., 2019) typically consist of a retrieval stage and a ranking stage. The retrieval stage is responsible for collecting a set of relevant products with minimum computational resources. The ranking stage then applies sophisticated machine learning (ML) algorithms to determine their impression positions. Traditional retrieval models rely on keyword matching (Manning et al., 2008), which may lead to poor results when the exact term match is unavailable. Recently, semantic matching models (Huang et al., 2013; Pang et al., 2016) have been adopted to improve retrieval performance (Mitra et al., 2018). These models are trained using click/purchase logs and typically separated by countries (Ahuja et al., 2020). However, such per-country specific training schema exposes three major drawbacks. First, maintaining country-specific models increase both operational burden and model iteration risks among countries. Second, the small amount of training data in low traffic countries may limit the ML model performance and this can also block the business expansion to new countries. Third, such models can not handle second language searches well. For example, the training data in US are dominated by English, which produces a model that cannot handle Spanish searches well. To solve above issues, ideally, a multilingual semantic retrieval model should be considered over monolingual retrieval models. However, how to design an effective and scalable multilingual semantic retrieval model for industry grade e-commerce search engine remains unsolved.

Built upon the success of pre-trained transformer-based models (Devlin et al., 2018; Yang et al., 2019b; Liu et al., 2019) such as Bidirectional Encoder Representations from Transformers (BERT) for natural language processing, multilingual BERT (M-BERT) has also demonstrated success for multilingual tasks (Pires et al., 2019). Though the techniques are promising, it is not straightforward to directly apply them to our problem due to the *vocabulary gap* issue (Mandal et al., 2019), i.e., customer searched queries are often short and from spoken input (e.g. 'fancy clothes') but product descriptions are usually in formal written style (e.g. 'formal attire'). There lacks a well established practice for fine-tuning multilingual BERT models on product search retrieval tasks.

In this work, we address the vocabulary gap by sharing information between queries and products in the model via graph convolution networks

(GCN). The query-to-product purchase/click logs naturally form a bipartite graph where each clicked product links with searched queries as neighbors. We expect to improve the product representation for retrieval tasks by incorporating information from its neighbor queries. For example, it is difficult for neural networks to directly match the query 'great gifts for child' to the product 'Disney puzzles for kids' given the vocabulary gap. But using the information that the product is connected with query 'children gifts', we can incorporate this information in its final representation, and the product will have a higher chance to be matched with the given query.

This paper presents an end-to-end multilingual retrieval system for e-commerce search engine. Our contributions are three-fold. 1**. Model:** We present a general framework that is compatible with any transformer-based models and any GCN architectures to capture interactions between products and search queries; 2**. Practice:** We provide a principled and practical guide of how to train the proposed model for large-scale product retrieval problem, e.g., how to define effective loss functions, how to feed online model-based hard negative samples to train the model and how to train the multilingual model with a novel one-language-at-a-batch (Sec 2.2) approach; 3**. System:** We discuss how to deploy the model to support product retrievals in multiple countries for e-commerce search.

To validate the effectiveness of our proposed method, we take offline experiments on billion-scale data across five languages and conduct online A/B testing experiments to measure the real traffic impacts. Through experimental results, our model outperforms state-of-the-art baselines by more than 25% and increases revenue and conversion over the current production system.

2 Methodology

We formulate the search retrieval task as follows. Supposed that we have a set of products $P = \{p_1, ..., p_n\}$. Each product p_i has a number of neighbor queries $Q_i = \{q_{i,1}, ..., q_{i,t}\}$ where $(q_{i,j}, p_i)$ appears in the search logs (customers search for $q_{i,j}$ and purchase p_i). For an arbitrary query q, we want to find the top-K relevant products from P. Note that q is not in $Q = \{Q_1, ..., Q_n\}$ when our retrieval system handles unseen queries.

2.1 Model Architecture

The model has two main components: (1) a query encoder that encodes search queries; (2) a product encoder that encodes both the product description and its neighbor queries. The product encoder has a GCN component that encapsulates the neighbor queries and product information.

Query Encoder: The query encoder could be any transformer-based encoder, such as BERT (Devlin et al., 2018), XLNet (Yang et al., 2019b), DistilBERT (Sanh et al., 2019), and RoBERTa (Liu et al., 2019). Choosing these transformer-based encoders over other encoders (e.g. LSTM (Hochreiter and Schmidhuber, 1997)) has several benefits. First, these models employ the word-piece tokenization which is robust to spelling errors and allows us to share vocabulary between different languages. In addition, transformer-based models can be easily parallelized and deployed online. We use the last hidden states of the encoders' [CLS] token as the embeddings for the query. For the other computationally more costly option of using the average pooling of the last hidden states of all tokens, we did not observe significant performance difference.

Product Encoder: The product encoder consists of 1) a transformer based encoder layer to extract the features of a product and its neighbor queries, and 2) a graph convolution layer that aggregates the extracted features to compute the final embeddings for a given product. The transformer-based encoder layer shares its parameters with the query encoder, and we also use the last hidden states of the encoder's [CLS] token as the features. The operations in the graph convolution layer are described in Algorithm 1.

Algorithm 1: Graph Convolution Layer

```
Input: extracted feature $\mathbf{h}_{p_i}$ of a product $p_i$, extracted features $\{\mathbf{h}_{q_{i,1}}, ..., \mathbf{h}_{q_{i,t}}\}$ of $p_i$'s neighbor queries $\{q_{i,1}, ..., q_{i,t}\}$
Output: the final product embedding $\mathbf{x}_{p_i}$
Step 1: $\mathbf{h}_{q_i} = \frac{1}{t}\sum_j \mathrm{ReLU}(\mathbf{W_q} \cdot \mathbf{h}_{q_{i,j}} + \mathbf{b_q})$
Step 2: $\mathbf{x}_{p_i} = \mathrm{ReLU}(\mathbf{W_p} \cdot \mathrm{CONCAT}(\mathbf{h}_{p_i}, \mathbf{h}_{q_i}) + \mathbf{b_p})$
```

The intuition of adding the graph convolution layer to the product encoder is that it can fill the vocabulary gap between queries and product descriptions. With the vocabulary gap and length distribution discrepancy between queries and product descriptions, directly using the transformer-extracted embeddings of queries and products for matching is sub-optimal. By incorporating neighbor query

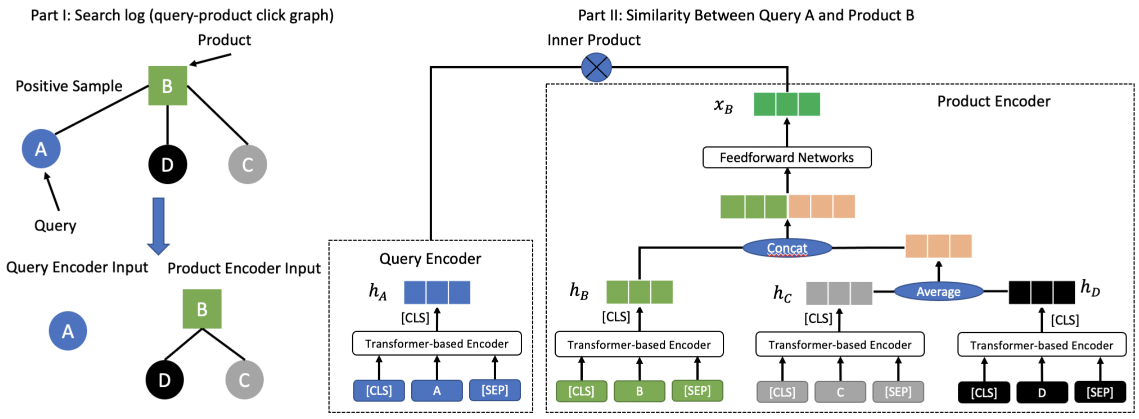

Figure 1: The base architecture of multilingual GCN. Each circle indicates a query and each rectangle represents a product. Query A and product B is a positive pair in the training data, while C and D are the neighbor queries of B. The query encoder takes query A as the input and outputs A's embedding, h_A. The product encoder takes B, C, D as the input and output B's embedding, x_B.

information into the product representation, the matching model not only learns information from query-to-product similarity, but also learns from query-to-query similarity. This is especially helpful to tail queries that have limited behavior signals.

Note that our framework is compatible with any GCN architectures in theory, such as GCN (Kipf and Welling, 2017) and GAT (Veličković et al., 2018), and we will leave those explorations for future work.

Loss Function: In the training stage, we use a pairwise ranking loss to train the model. Specifically, for each query q_i in the training set, we sample a positive product p_{i+} and a negative product p_{i-}. The triplet loss is defined as

$$L = \sum_i \text{Log}(1 + \exp(\mathbf{x}_{q_i} \cdot \mathbf{x}_{p_{i-}} - \mathbf{x}_{q_i} \cdot \mathbf{x}_{p_{i+}})) \tag{1}$$

The intuition of is that we want the inner product of the positive pairs <$\mathbf{x}_{q_i}$, $\mathbf{x}_{p_{i+}}$> to be larger than the inner product of the negative pairs <$\mathbf{x}_{q_i}$, $\mathbf{x}_{p_{i-}}$>and the margin to be as large as possible.

2.2 Training Details

There are two key factors in successfully training the aforementioned framework: 1) how to select proper negative samples for training the model; 2) how to properly feed training data from different languages to the model.

Negative Sampling: Defining negative samples for semantic retrieval tasks is a tricky problem. A widely-adopted method is *random sampling*, where one can randomly sample a product from the product catalog as the negative samples for a given query. However, this simple setting would generate sub-optimal results, since the randomly selected negative samples can be too easily distinguished from the positive samples. Thus, it rarely brings knowledge for model learning and produces a model with low discriminative power for the retrieval task.

Recent research work has indicated that using hard negatives could improve the model performance for retrieval tasks (Ying et al., 2018; Nigam et al., 2019; Huang et al., 2020). The hard negative sample should be the product that is somewhat related to the query but not a exact match. In the search retrieval scenario, we can define the following three kinds of hard negatives.

Behavior-based hard negative: the negative samples are defined by users' behavior and are extracted from the search logs. For a given query, we take those products that were shown to the customer but not clicked as the hard negatives.

Offline model-based hard negative: the negative samples are calculated by the current model in an offline fashion. Specifically, we first use the current model to generate the embeddings for all queries and products in the training set, and then calculate the inner product between all queries and all products. For each query, we sample negatives from its top-200 to top-1000 relevant products.

Online model-based hard negative: the negative samples are generated on-the-fly with model learning. Specifically, we first randomly sample a batch of products. Then we use the current model to calculate the inner product between embeddings of these products and a batch of queries. For each query in the batch, we select the product with highest inner product value as the hard negative sample.

We argue that the online model-based hard negative is the most suitable sampling method to our application. The behavior-based hard negative samples requires additional data collection processes and often yields worse results in the search retrieval task (Huang et al., 2020). In fact, it is more suitable to the ranking task where the candidate pool is more refined. Besides, the offline model-based hard negative is too time-consuming, as we have to compute K-NN for each query in training set when we select/update the hard negatives.

Multilingual Data Fusion: How to properly feed the multilingual data to the model is another crucial factor to the training process. The amount of training data from different languages/countries varies greatly, and therefore low-resource languages would be underrepresented in the neural network model. Inspired by (Devlin et al., 2018), we perform exponentially smoothed weighting of the data. We would take the exponent of the percentage of a language by factor S and then re-normalize. Suppose there are two languages, English and Spanish, which accounts for 90% and 10% of training data respectively. The re-normalized distribution is $\frac{0.9^{0.7}}{0.9^{0.7}+0.1^{0.7}} = 0.82$ for English. Therefore, high-resource languages will be under-sampled, and low-resource languages will be over-sampled.

We also find that mixing training samples from multiple languages in one training batch makes it harder to train the model. Firstly, the negative sampling space is more complicated: we could sample a Spanish product as the negative sample of a English query. These easy negatives provide little knowledge to the model. In addition, different languages of training data appear in the same batch, which makes the batch gradient less stable. We propose to train the model with *one-language-at-a-batch*, and make the negative sampling process language-dependent. In the experiment, we observe that doing so dramatically increases the performance on all languages by 5-6% recall.

3 Deployment

The deployment of the proposed model has two parts: a query encoder and pre-computed product embeddings. As the query encoder is a standard transformer-based model and many papers have talked about the serving of it, this part can be easily deployed online. For the pre-computed product embeddings, we first compute the embeddings for products in our catalog, and incrementally update the product embedding periodically. To avoid repeated computations during the inference time (multiple products might have the same neighbor query), we first use the transformer-based encoder to compute the embeddings for all queries and products in the graph, and then join the products' embeddings with their neighbor queries' embeddings. Lastly, we pass these intermediate embeddings through the GCN layer to generate the final product embeddings. During the search retrieval step, we simply use the query encoder to extract the embedding for an input query. Then, we find the K-nearest neighbors (K-NN) products by calculating the cosine-distance between the query embedding and the pre-computed product embeddings. The K-NN products are used to augment the matchset of the given query.

4 Experiments

We collect the data from a large e-commerce platform that has business in multiple countries. To provide a comprehensive understanding on the role of multilingual queries in a real-world product search system, we select five countries: United States (US), Spain (ES), France (FR), Italy (IT) and Germany (DE). We subsample our data from one year of search log in each country. We organize the collected search log into query-product pairs with different customer behavior signals, e.g. click/purchase. For model offline testing, we first randomly sample 20K queries from each country. We then use our algorithm to rank a sub-corpus of 100K products (in each country) for those queries. The 100K product corpus consists of purchased products for those 20K queries and additional random negatives. Since our work focuses on the retrieval part of a product search engine, we adopt two matching metrics to summarize our results: Recall@10 (recall) and mean Average Precision (mAP). We employ the multilingual DistilBERT (Sanh et al., 2019) with 6 layers and 768 hidden units as our encoder. We set the batch size to 640 and use Adam optimizer (Kingma and Ba, 2014) with $\alpha = 0.0001, \beta_1 = 0.9$, and $\beta_2 = 0.999$. We run all the experiments on an AWS p3dn.24xlarge instance with 768GB memory and 8 NVIDIA V100 GPUs. We train the model on 8 GPUs in a distributed fashion. The model is trained for 140K batches, where the 28K 'warm up' batches are trained with random negatives and remaining batches are trained with online model-based hard negatives.

Table 1: Matching performance for our model and baselines.

	US		ES		FR		IT		DE	
method	recall	mAP	recall	mAP	recall	mAP	recall	mAP	recall	mAP
DSSM	49.53%	34.09%	38.82%	22.26%	38.16%	21.98%	42.51%	24.68%	46.87%	30.16%
Multilingual BERT	38.82%	25.14%	23.06%	12.07%	25.41%	13.67%	24.36%	13.06%	25.31%	15.18%
Our model w/o BERT	79.79%	60.06%	66.53%	39.01%	68.01%	41.43%	70.32%	42.66%	74.69%	51.79%
Our model w/o GCN	80.83%	60.68%	68.98%	41.73%	70.03%	42.28%	72.88%	44.98%	76.69%	53.75%
Our model	**85.86%**	**66.69%**	**73.60%**	**44.40%**	**74.97%**	**47.07%**	**77.16%**	**48.03%**	**81.44%**	**58.33%**

4.1 Comparison Results

We compare against the following baselines:
DSSM (Huang et al., 2013) is an earlier work to extract the semantic representations of queries and documents from large-scale click-through data by leveraging deep neural networks. We train 5 language-specific DSSM models with monolingual training data.
Multilingual BERT (Devlin et al., 2018) is the vanilla BERT without any fine-tuning. We use the *bert-base-multilingual-cased* model from huggingface implementation. We directly use the Multilingual BERT to encode the queries/products, and take the output [CLS] embeddings as the representations of queries/products.
Our model w/o BERT is a variant of our model, where we replace the DistilBERT encoder with a one-layer feed forward neural network and use the same word embedding matrix as the DistilBert. We train this model with exactly the same settings as we train the main model.
Our model w/o GCN is a variant of our model, where we remove the GCN module. It means that we only use product descriptions to get the product embeddings, and there is no graph convolution layer in the product encoder.

Table 1 shows (1) Multilingual BERT without any fine-tuning does not work for multilingual search retrieval tasks. It has the lowest recall and mAP, which proves the necessity of designing proper fine-tuning tasks for BERT-based model; (2) replacing the feed forward neural networks with DistilBERT leads to 6% - 7% recall and 5.5% - 6% mAP improvement on all languages; (3) adding GCN module to the product encoder further achieves significant boosts (5-6% recall improvement and 4.5%-6% mAP improvement), suggesting that GCN help the *vocabulary gap* issue.

4.2 Ablation Study

Negative Sampling: We try three kinds of hard negatives as illustrated in Section 2.2. Table 2 shows the results of training our model with different hard negatives. We observe that the performance of offline model-based hard negatives is similar to that of online model-based hard negatives ($<$ 0.4% recall difference). However, computing the offline model-based hard negatives takes a total of 4x training time. Besides, training with behavior-based hard negatives has worst results (-10% recall, -7% mAP), because behavior-based negatives are not appropriate for retrieval tasks (Huang et al., 2020), since most impressed products are often relevant to the query. Including them as negatives confuses the model from focusing on retrieval tasks.

Multilingual Training Data Fusion: We test three data fusion strategies: 1) sample by unweighted data size + train with one language at a batch (*unweight+separate*); 2) sample by exponentially weighted data size + train with mixed languages in a batch (*weight+mix*); 3) sample by exponentially weighted data size + train with one language at a batch (*weight+separate*). Table 3 shows the results from different multilingual data fusion strategies. By exponentially weighting the training data, we can improve the matching performance in low-resource languages (ES, FR, IT) without hurting the performance in high resource languages (DE and US). Besides, *weighted+separated* beats *weighted+mixed* by 2-3% recall and 1-2% mAP margin on all languages, suggesting that training with one-language-at-a-batch is superior to mixed training.

4.3 Online Experiments

We report our findings from online A/B experiments on a large-scale e-commerce website with our multilingual GCN model. We run online match set augmentation experiments in three countries and two languages. The proposed algorithm significantly improves business metrics in all countries, leading to +1.8% increase in average clicks, +0.3% in revenue, and +0.4% in conversion. We also observe reformulated searches decreased by 1%. This reduction results in customers finding their desired products with less effort, likely from that our model bridges the vocabulary gap between queries and products. All results provide evidence that our

Table 2: Matching performance with different kinds of hard negatives.

	US		ES		FR		IT		DE	
hard negative	recall	mAP	recall	mAP	recall	mAP	recall	mAP	recall	mAP
behavior	76.62%	59.36%	63.28%	38.80%	63.90%	40.31%	66.17%	42.18%	68.44%	49.67%
offline model	85.44%	66.43%	**73.95%**	**44.33%**	74.86%	46.76%	**77.43%**	**48.20%**	**81.52%**	**58.68%**
online model	**85.86%**	**66.69%**	73.60%	44.40%	**74.97%**	**47.07%**	77.16%	48.03%	81.44%	58.33%

Table 3: Matching performance with different multilingual data fusion strategies.

	US		ES		FR		IT		DE	
fusion strategy	recall	mAP	recall	mAP	recall	mAP	recall	mAP	recall	mAP
weight+mix	84.61%	65.24%	70.45%	42.01%	71.41%	44.34%	73.39%	45.46%	78.85%	55.91%
unweight+separate	85.82%	66.60%	71.69%	42.93%	73.29%	45.63%	74.53%	46.66%	80.61%	57.73%
weight+separate	**85.86%**	**66.69%**	**73.60%**	**44.40%**	**74.97%**	**47.07%**	**77.16%**	**48.03%**	**81.44%**	**58.33%**

algorithm leads to better retrieval performance and can help customers fulfill their shopping missions.

5 Related Works

Search engine retrieval has been based on lexical match to identify relevant documents for queries. Recently, major industry search engines (Nigam et al., 2019; Huang et al., 2020; Fan et al., 2019) have incorporated *semantic matching* for improvements. Such algorithms can be classified into *embedding-based models* and *interaction models*. Embedding based models such as DSSM (Huang et al., 2013) and its subsequent works (Shen et al., 2014; Palangi et al., 2016; Hu et al., 2014) convert queries and documents into embeddings for retrieval. Interaction models, like MatchPyramid (Pang et al., 2016) and DRMM (Guo et al., 2016) leverage interaction matrices to capture local term matching. However, they are computationally costly for industry data.

With BERT (Devlin et al., 2018) becoming the state-of-the-art embedding method, it is adopted for various applications (Yang et al., 2019a; Yu et al., 2020; Khattab and Zaharia, 2020; Humeau et al., 2020; Chang et al., 2020). However, how to properly fine-tune BERT for retrieval tasks in product search remains unstudied. Our work fills this gap and provides a practical guide to fine-tune BERT-based models using production-scale search data. Furthermore, M-BERT provides representations for 104 languages and has proven ability to handle multilingual tasks (Pires et al., 2019). Other multilingual embedding models have also been proposed and validated (Schwenk and Douze, 2017; Conneau and Lample, 2019; Conneau et al., 2020). Our method is flexible and so that all these models can serve as a component.

Notably, our multilingual problem is different from the cross-lingual information retrieval (CLIR) problem (Nie, 2010; Jiang et al., 2020) , which refers to the scenario where the query is in one language but document is in other languages. In our problem, product descriptions and search queries are always in the same primary language and except a small fraction in different languages

Graph neural network is gaining prominence in ML applications (Ying et al., 2018; Zhang et al., 2019). The notion of "graph convolutions" is first proposed in (Bruna et al., 2014) with spectral graph theory. Later, GraphSAGE (Hamilton et al., 2017) redefines it to avoid operating on the entire graph. Recent efforts (Wang et al., 2019; Berg et al., 2018) adopt GCN to the user-item interaction graph and leverage the neighbors for recommendation. LightGCN (He et al., 2020) reported that neighborhood aggregation is the only important component of GCN, and weighted-sum of neighbor embeddings yield the best results. Our method leverages GCN to incorporate neighbor queries' information into product embedding, which bridges the vocabulary gap between query and product. Moreover, our framework is compatible with any GCN architectures, so can leverage the advances there.

6 Conclusion

Our paper present a multilingual graph convolution networks model for language-agonistic semantic retrieval in product search engine. Our method not only can handle multilingual text data, but also addresses the *vocabulary gap* issues between queries and product descriptions. We also provide a practical guide of fine-tuning the proposed model on retrieval tasks. We conduct various experiments including offline evaluation on 5 languages and online A/B test in three countries. In all experiments, our model consistently beats the baselines and demonstrates improved product discoverability.

References

Aman Ahuja, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. 2020. Language-agnostic representation learning for product search on e-commerce platforms. In *Proceedings of the 13th ACM International Conference on Web Search and Data Mining*, pages 7–15.

Rianne van den Berg, Thomas N Kipf, and Max Welling. 2018. Graph convolutional matrix completion. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and locally connected networks on graphs. In *2nd International Conference on Learning Representations*.

Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval. In *8th International Conference on Learning Representations*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, pages 7057–7067.

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*.

Miao Fan, Jiacheng Guo, Shuai Zhu, Shuo Miao, Mingming Sun, and Ping Li. 2019. MOBIUS: towards the next generation of query-ad matching in baidu's sponsored search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2509–2517. ACM.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30*, pages 1024–1034.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, pages 1735–1780.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050.

Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2553–2561.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, pages 2333–2338.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2020. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *8th International Conference on Learning Representations*.

Zhuolin Jiang, Amro El-Jaroudi, William Hartmann, Damianos G. Karakos, and Lingjun Zhao. 2020. Cross-lingual information retrieval with BERT. In *Proceedings of the workshop on Cross-Language Search and Summarization of Text and Speech@LREC*, pages 26–31.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations*. OpenReview.net.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Aritra Mandal, Ishita K Khan, and Prathyusha Senthil Kumar. 2019. Query rewriting using automatic synonym extraction for e-commerce search. In *Proceedings of eCOM Workshop@the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Christopher D Manning, Hinrich Schütze, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Cambridge university press.

Bhaskar Mitra, Nick Craswell, et al. 2018. *An introduction to neural information retrieval*. Now Foundations and Trends.

Jian-Yun Nie. 2010. *Cross-Language Information Retrieval*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian Allen Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. 2019. Semantic product search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2876–2885.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pages 694–707.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert? In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 4996–5001.

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*.

Holger Schwenk and Matthijs Douze. 2017. Learning joint multilingual sentence representations with neural machine translation. In *Proceedings of Rep4NLP@the 55th Annual Meeting of the Association for Computational Linguistics*, pages 157–167.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pages 101–110.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *6th International Conference on Learning Representations*.

Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pages 165–174.

An Yang, Quan Wang, Jing Liu, Kai Liu, Yajuan Lyu, Hua Wu, Qiaoqiao She, and Sujian" Li. 2019a. Enhancing pre-trained language representations with rich knowledge for machine reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2346–2357.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019b. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online.

Yuan Zhang, Dong Wang, and Yan Zhang. 2019. Neural ir meets graph embedding: A ranking model for product search. In *Proceedings of The Web Conference (WWW)*, pages 2390–2400.