ListBERT: Learning to Rank E-commerce products with Listwise BERT

Lakshya Kumar*
lakshya.kumar@myntra.com
Myntra Designs Pvt. Ltd.
India

Sagnik Sarkar*
sagnik.sarkar@myntra.com
Myntra Designs Pvt. Ltd.
India

ABSTRACT

Efficient search is a critical component for an e-commerce platform with an innumerable number of products. Every day millions of users search for products pertaining to their needs. Thus, showing the relevant products on the top will enhance the user experience. In this work, we propose a novel approach of fusing a transformer-based model with various listwise loss functions for ranking e-commerce products, given a user query. We pre-train a RoBERTa model over a fashion e-commerce corpus and fine-tune it using different listwise loss functions. Our experiments indicate that the RoBERTa model fine-tuned with an NDCG based surrogate loss function(approxNDCG) achieves an NDCG improvement of **13.9%** compared to other popular listwise loss functions like ListNET and ListMLE. The approxNDCG based RoBERTa model also achieves an NDCG improvement of **20.6%** compared to the pairwise RankNet based RoBERTa model. We call our methodology of directly optimizing the RoBERTa model in an end-to-end manner with a listwise surrogate loss function as **ListBERT**. Since there is a low latency requirement in a real-time search setting, we show how these models can be easily adopted by using a knowledge distillation technique to learn a representation-focused student model that can be easily deployed and leads to $\sim$ **10** times lower ranking latency.

KEYWORDS

Transformer, RoBERTa, BERT, Ranking, Retrieval, E-commerce Products, Knowledge-Distillation, NDCG, Listwise Loss, Pairwise Loss etc

ACM Reference Format:
Lakshya Kumar* and Sagnik Sarkar*. 2022. ListBERT: Learning to Rank E-commerce products with Listwise BERT. In *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR eCom'22).* ACM, New York, NY, USA, 5 pages.

1 INTRODUCTION

In an E-commerce[1] setting every day millions of users enter different queries to search for their desired products. In order to retain the users over the platform, it becomes essential to serve them relevant product results that are coherent with their query. Also, it becomes important to understand the intent of the query with respect to different products in an e-commerce catalog. There are millions of products in a catalog and serving a relevant set of products requires an understanding of both the query and the product. Recently, the transformer[28] based models have gained popularity due to their exemplary performance on various NLP tasks, owing to which they are getting adopted in different industry settings to tackle various problems. One can directly apply the transformer-based models to deeply understand the textual data of the queries and the products and learn good contextual representations. These representations can further help in learning a ranking function to finally serve the products resulting in user satisfaction. In an e-commerce platform, a huge amount of implicit user signals gets captured in the search logs that can be directly leveraged to learn a ranking function. Also in the past work[24] it has been shown that learning a ranking function from such implicit signals has more utility. In order to learn a ranking function, different learning-to-rank(LTR) methodologies can be applied but listwise is shown to be the most optimal[22]. In this work, we propose a novel approach of ranking e-commerce products given a user query with a transformer-based RoBERTa[14] model. Our model directly learns the interactions between the query and the products within the RoBERTa model and finally optimizes over a listwise surrogate loss function for an effective product ranking. We prepare a dataset from a popular fashion e-commerce platform to fine-tune the RoBERTa model directly with the listwise loss. Our main contributions are as follows:

- Propose a novel approach of end-to-end training of a transformer based model with a listwise surrogate loss function.
- Compare the efficacy of different listwise and pairwise loss functions with respect to the NDCG evaluation metric.
- Highlight on the adoption of such models in a low-latency e-commerce search setting using knowledge distillation and training a representation-focused student model.

2 RELATED WORK

Based on the loss function used, LTR approaches proposed in the literature [12] can be classified into 3 main categories namely, pointwise [26] [13], pairwise [2] and listwise [3] [30] [22]. Empirically, listwise approaches usually perform better than pointwise or pairwise approaches mainly because popular information retrieval metrics like NDCG[8] consider entire search results lists at once, unlike pointwise and pairwise approaches. Recently, pre-trained BERT models have been successfully applied to the LTR problem. One of the themes in which BERT models are applied to the ranking problem is the multi-stage ranking setup. Nogueira et al. [19] proposed a multi-stage document ranking setup where the point-wise BERT is

[1] In this work, we mean fashion e-commerce but this research work applies to general e-commerce

*Both authors contributed equally to this research.

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

applied for filtering the documents that fall below a certain threshold in the first stage. In the second stage, a pairwise BERT model is applied to rank the candidate documents to finally serve the results. This multi-stage document ranking model has shown good performance over standard datasets mainly MS MARCO [18] and TREC CAR[5]. Various other popular multi-stage ranking architectures are mentioned in [31]. Another theme for applying BERT-based models to the ranking problem is to use dense learned representations leveraging efficient approximate nearest neighbor lookup. Models like Sentence-BERT[23] and DPR[10] have been proposed for learning the dense representations for ranking. For e-commerce search use cases, there is a strict latency requirement and in order to reduce the latency several works have been proposed using knowledge distillation. Lu et al. [15] applied knowledge distillation and trained a TwinBERT model in the teacher-student framework for effective and efficient retrieval.

In this work, we apply a BERT-based[2] model and fine-tune it in an end-to-end manner on various listwise loss functions and thereby demonstrate an effective method of ranking products in a fashion e-commerce domain. We also fine-tune the BERT based model using a popular pairwise RankNet[1] loss function in order to compare it with listwise loss functions. We apply a Knowledge Distillation technique [7] to train a representation focused [6] BERT model that is more suitable in a low-latency e-commerce search setting.

3 DATA PREPARATION

3.1 Tokenization and Pre-training Dataset

The user click-stream data for the product ranking task is not sufficient to optimize the large number of parameters of BERT based models like RoBERTa. Typically these models are first pre-trained on a large corpus of domain specific text and then fine-tuned for the downstream tasks. Therefore, we first pre-train the RoBERTa model from scratch over a fashion dataset which we call as pre-training dataset. In order to better handle the out-of-vocabulory words and effectively tokenize the model inputs for pre-training as well as fine-tuning, we train the BPE[25] tokenizer over the pre-training dataset. Once trained, the BPE tokenizer is used across all the different model variants in order to give tokenized query and product inputs to the model. Our pre-training dataset consists of product descriptions (formed using product attributes) and user queries. The median and p90 of the product description lengths are **27** and **19** respectively. We collect user queries entered in a span of **1** month while filtering out single-word queries or low-frequency queries (entered less than 10 times). The median and p90 of the query lengths are **3** and **2** respectively. The overall pre-training dataset consists of over **5 M** training instances with a mixture of product descriptions and user queries.

3.2 Fine-tuning Dataset

We leverage a massive amount of click-stream data generated from user sessions on our e-commerce platform. We follow the idea presented in Santu et al. [24], to convert the empirical Click-Through-Rate (CTR) of products (appearing in the result list of a query) into graded relevance ratings. The formula for the ground truth relevance ratings using CTR, is as follows:

$$rel_{ctr}(q,d) = ceil(4.\frac{ctr(q,d)}{max_{d \in D_q} ctr(q,d)}) \quad (1)$$

where $ctr(q,d) = \frac{clicks(q,d)}{impressions(q,d)}$, D_q is the set of products to be ranked for a query q. We sample **2** weeks of click-stream data for fine-tuning our models. We consider only those products in the result set of a query that had received at least **50** impressions to ensure reliable CTR estimates. We limit the result set of a query to the top **30** products returned by the search engine[3]. The train data consists of ~ **45k** queries. The median and p90 lengths of the result set of these queries are **30** and **17** respectively. The median and p90 of the query lengths are both **3**. To evaluate the performance of our models we create the test set by sampling **1** week of click-stream data in a time period which is temporally separated from the train data period by **2** months in order to ensure there will be less overlap between the train and test queries. [4] The test data consists of ~ **20k** queries. The median and p90 lengths of the result set of the test set queries are **30** and **7** respectively. Since the train and test sets are formed using a temporal split there are queries which are common to both sets. The test set contains ~ **40** % new queries compared to the train set. Moreover, the result sets for the common queries were also different in the train and test sets. The median percentage of new products per query to be ranked in the test set for the common queries was ~ **67** %

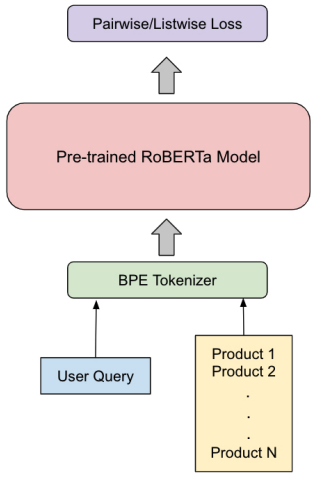

Figure 1: Training setup for ListBERT

4 APPROACH

To learn a good latent representation for both query and product along with the interactions between the query and the product tokens, we use the RoBERTa architecture as our fundamental model. We follow a two-step approach where we first pre-train the RoBERTa model from scratch for adapting to the e-commerce domain and then fine-tune it using pairwise and listwise loss functions for re-ranking of products given a user query.

[2]BERT-based, transformer-based are used interchangeably.

[3]Denotes the system within e-commerce that retrieves the products given a user query.

[4]In actual production settings, the train and test would be typically have a temporal separation of 1 week

4.1 Pre-training for Domain Knowledge

The originally proposed pre-trained RoBERTa model[14] has knowledge driven from BookCorpus(16GB)[32], CC-NEWs(76GB)[17], OpenWebText(38GB)[9], Stories(31GB)[27] etc which do not align with the e-commerce queries and products. Therefore, we create a dataset as explained in Section 3.1 for pre-training the RoBERTa model in order to incorporate an understanding of the e-commerce domain. We only use the Masked Language Modelling objective[4] which is defined as:

$$Loss(sentence) = \sum_{i \in MASK} loss(token_i) \tag{2}$$

$$loss(token_i) = -log(p(actualToken_i)) \tag{3}$$

where $Loss(sentence)$ corresponds to the overall loss with respect to a sentence in the pre-training dataset. The $loss(token_i)$ indicates the loss corresponding to each of the MASK tokens. As we randomly mask some of the tokens in a sentence, the loss is computed only with respect to the masked tokens. In equation 3, $p(actualToken_i)$ indicates the probability that the model is assigning to the actual token at position i given the left and the right context tokens. We measure the perplexity after pre-training the model and report the same in Section 5. Finally, we fine-tune this model over a ranking dataset to directly optimize for pairwise and listwise loss functions.

4.2 Fine-tuning

To re-rank the products given a user query, we fine-tune the pre-trained RoBERTa model as shown in Figure 1. We optimize a pairwise loss function, i.e., RankNet and three different listwise loss functions, i.e., ListNET[3], ListMLE[30] and approxNDCG[22]. Finally, we compare these different BERT models on a test dataset as described in 3.2. The formulation for these different loss functions is briefly described in the following subsections. We also describe a knowledge distillation technique where we train a representation focused student model using the ListBERT model as the teacher model in order to reduce the ranking latency.

4.2.1 ***RankNet.*** RankNet First introduced in Burges et al. [1] using feed forward neural network as the fundamental model. The main objective of RankNet is to perform pairwise ranking optimization by minimizing the number of inversions in the ranked list. An inversion indicates the condition where a low rank product is ranked above a high rank product. In order to solve this optimization problem as described in Pobrotyn et al. [21], we first generate the pairs of the products with respect to a given query from the ground truth dataset. Among these pairs, we calculate the true difference between their relevance and retain only those pairs where the difference is positive and finally denote them as $True_{diff}$. From the RoBERTa model, we generate scores for each of the **N** products[5] with respect to a given query and consider only those pairs which are present in ground truth pair list, i.e., $True_{diff}$ and calculate the difference in their predicted model score and denote them as $Pred_{diff}$. Finally, we apply the below loss function present in Paszke et al. [20] for all the queries in the training set to optimize for the pairwise ranking in RankNet.

$$l(True_{diff}, Pred_{diff}) = -\sum_{\substack{i \in True_{diff} \\ j \in Pred_{diff}}} [w_i * \mathbb{1}_{i>0} * log(\sigma(j)))] \tag{4}$$

$$w_i = pow(rel_m, 2) - pow(rel_n, 2) \tag{5}$$

where rel_m and rel_n indicates the true relevance of document m and n respectively in a pair i in $True_{diff}$.

4.2.2 ***ListNET.*** We can assume that we have p user queries which can be denoted as $q^{(i)}$, i = 1,2,....,p and n_i denotes the number of products corresponding to the ith query. Also, we can assume that x_j^i corresponds to a feature vector that we obtain using both the query i and the product j. Let f_w denote a scoring function that assigns a score given x_j^i. As per Cao et al. [3], the ListNET loss for a query is given by the following equations:

$$P_{z^i(f_w)}(x_j^i) = \frac{exp(f_w(x_j^i))}{\sum_{k=1}^{n_i} exp(f_w(x_k^i))} \tag{6}$$

$$L(y^i, z^i(f_w)) = -\sum_{j=1}^{n_i} P_{y^i}(x_j^i) log(P_{z^i(f_w)}(x_j^i)) \tag{7}$$

Where z^i denotes the predicted scores list for a query $q^{(i)}$. And y^i denotes the actual scores list as per the true relevance of the products for a given query.

4.2.3 ***ListMLE.*** Xia et al. [30] proposed ListMLE loss function, where the likelihood function is given as:

$$\phi(f_w(x), y) = -log(P(y|x; f_w)) \tag{8}$$

where $P(\mathbf{y}|\mathbf{x}; f_w)$ is defined as:

$$P(y|x; f_w) = \prod_{j=1}^{n_i} \frac{exp(f_w(x_{y(j)}))}{\sum_{k=j}^{n_i} exp(f_w(x_{y(k)}))} \tag{9}$$

where $x_{y(j)}$ indicates the feature vector of a product at position j in the true relevance list. For p queries in the training set, the overall likelihood loss can be given as:

$$-\sum_{i=1}^{p} logP(y^i|x^i; f_w) \tag{10}$$

where $\mathbf{y}^i$ corresponds to the actual ranked list of products and $\mathbf{x}^i$ corresponds to the feature vectors for n^i products for a given query i.

4.2.4 ***approxNDCG.*** In order to evaluate a ranking list, NDCG[8] is the popular metric that is widely used. But the NDCG cannot be directly optimized owing to its non-differentiability. In order to solve for this, as per Qin et al. [22], position in the DCG[6] calculation can be approximated using the below formulation:

$$\pi_f(i) = 1 + \sum_{j \neq i} \mathbb{1}_{f(x)|_i < f(x)|_j} \tag{11}$$

where $\mathbb{1}_{s<t}$ is an indicator function that is 1 when $s < t$ and 0 otherwise. In order to approximate NDCG, the authors proposed the below formulation using the sigmoid function,

$$\mathbb{1}_{s<t} = \mathbb{1}_{t-s>0} \approx \sigma(t-s) = \frac{1}{1 + \exp^{-\alpha(t-s)}} \tag{12}$$

[5]For every query, we have a ranked list of products which is indicated by N.

[6]DCG corresponds to discounted cumulative gain in the NDCG formula.

In our experiments, RankNet, ListNET, ListMLE, and approxNDCG loss functions are used for fine-tuning the RoBERTa model. After fine-tuning, we apply knowledge distillation to train a representation-focused student model using the best performing $RoBERTa_{NDCG}$ model as a teacher model for low latency ranking.

4.2.5 ***Knowledge Distillation.*** We train a student model consisting of a RoBERTa model with a linear layer at the top. We optimize the student model with a margin Mean Square Error(MSE) loss as proposed in [7]. The margin MSE loss is given as:

$$L(Q, P^+, P^-) = MSE(M_s(Q, P^+) - M_s(Q, P^-), \\ M_t(Q, P^+) - M_t(Q, P^-)), \quad (13)$$

where Q, P denotes a query and a product respectively. P^+ and P^- corresponds to the high and low relevant products[7] respectively. M_t denotes our teacher model, i.e., $RoBERTa_{NDCG}$ and M_s denotes student model. Unlike the teacher model, the student model takes the query and the product text without concatenation. We independently pass the query and the product to obtain their latent representation and then take the dot product to generate a score which is used to optimize the margin MSE loss. As our student model is a representation-focused model, after training it, we can pre-compute and store the representation for all the products in an e-commerce catalog. In the real-time, we generate the query representation on-the-fly from the student model to rank the relevant products by taking the dot product between the query representation and the pre-computed representations of the candidate products. In the next section, we highlight on the performance of the student model and also describe the latency improvement that can be achieved with the representation focused student model.

Model Type	Surrogate Loss	NDCG
$RoBERTa_{RNet}$	RankNet	0.625
$RoBERTa_{Net}$	ListNET	0.630
$RoBERTa_{MLE}$	ListMLE	0.662
$RoBERTa_{NDCG}$	approxNDCG	**0.754**

Table 1: Ranking Performance of ListBERT models.

5 RESULTS & EXPERIMENTAL SETUP

In order to pre-train the RoBERTa model, we first train a Byte-Pair-Encoding(BPE)[25] tokenizer over a pre-training dataset(as explained in Section 3.1). The vocabulary size of the tokenizer is **30K**. Due to the small pre-training data size(∼ 6GB) and computation constraints, we use a smaller version of the RoBERTa model having **6** hidden layers and **12** attention heads. The input dimension is **768** and the number of input tokens has a maximum size of **512**. As we optimize the Masked Language Modelling[4] objective, the pre-training data is prepared by randomly masking **15%** of the tokens from each of the sentences. For faster pre-training, the multi-GPU pre-training is done with **2 Tesla V100 GPUs** using Mixed Precision Training[16]. After pre-training for **4 epochs**, the perplexity of the RoBERTa model over the test data(∼ 1GB)[8] is **1.40**.
For fine-tuning, we take the pre-trained model and apply a linear layer on top of it. Finally, we experiment with different loss functions as explained in Section 4. We use allRank[21] framework-based implementation of pairwise and listwise losses. In order to fine-tune the RoBERTa model, we concatenate both the query and the product description with a **<SEP>** token and feed it into the RoBERTa model. We take the representation corresponding to the **<CLS>** token and feed it into the linear layer and optimize for the pairwise and listwise loss functions. For both pre-training and fine-tuning, we use **Pytorch**[20] framework and **Huggineface**[29] based implementation of the RoBERTa model. We use **Adam**[11] optimizer to update the parameters in both pre-training and finetuning. The results of different variants of the RoBERTa model after fine-tuning for **10 epochs** are shown in Table 1. The RoBERTa model fine-tuned with approxNDCG outperforms RankNet, ListNET and ListMLE loss functions based RoBERTa models with an NDCG of **0.754**.
For knowledge distillation, we train a student RoBERTa model by obtaining the scores from the teacher model[9] for every query and product pair in the training data[10]. Student RoBERTa model takes the query and product independently unlike the teacher which takes the concatenation of query and product. The student model takes the dot product between the query and product representation and then tries to optimize using margin MSE loss as explained in 4.2.5. The student RoBERTa model achieves an NDCG of **0.73** over the test data after training for **4 epochs**. For the teacher model, the average ranking latency is ∼ **154 ms**, whereas for the student model the ranking latency is ∼ **15 ms**.

6 CONCLUSION & FUTURE WORK

We proposed a novel way of end-to-end training of the transformer-based RoBERTa model in a listwise setting which we call as **ListBERT**, for ranking e-commerce products. Fine-tuning the RoBERTa model with listwise loss function represents a novel way of learning query and product interactions along with their contextual representations that helps in more optimal ranking of e-commerce products. The RoBERTa model with approxNDCG as the listwise loss outperforms both ListNET and ListMLE with an NDCG of **0.754**. The ApproxNDCG based RoBERTa model also outperforms the pairwise loss based RoBERTa which we calls as $RoBERTa_{RNet}$. In order to reduce the ranking latency, we trained a representation-focused student model using knowledge distillation that achieves an NDCG of **0.73** with a 10x latency improvement as compared to the best teacher model, i.e., $RoBERTa_{NDCG}$. In future this work can be extended to incorporate user features for personalized product ranking. It will be interesting to explore different ways to reduce the model size and see the impact of the same on the model performance. Experimenting with other knowledge distillation approaches is another interesting future work.

REFERENCES

[1] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to Rank Using Gradient Descent. In

[7] P^+ has higher relevance as compared to P^-

[8] This test data is with respect to the pre-training data and consists of user queries and product descriptions.

[9] we take the best teacher model, i.e., $RoBERTa_{NDCG}$

[10] 12M instances.

Proceedings of the 22nd International Conference on Machine Learning (Bonn, Germany) *(ICML '05)*. Association for Computing Machinery, New York, NY, USA, 89–96. https://doi.org/10.1145/1102351.1102363

[2] Chris J.C. Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview*. Technical Report MSR-TR-2010-82. https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/

[3] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to Rank: From Pairwise Approach to Listwise Approach. In *Proceedings of the 24th International Conference on Machine Learning* (Corvalis, Oregon, USA) *(ICML '07)*. Association for Computing Machinery, New York, NY, USA, 129–136. https://doi.org/10.1145/1273496.1273513

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[5] Laura Dietz. 2019. TREC CAR Y3: Complex Answer Retrieval Overview.

[6] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. 55–64. https://doi.org/10.1145/2983323.2983769

[7] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation. *CoRR* abs/2010.02666 (2020). arXiv:2010.02666 https://arxiv.org/abs/2010.02666

[8] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (Oct. 2002), 422–446. https://doi.org/10.1145/582415.582418

[9] & David Bourginl Joshua Peterson, Stephan Meylan. 2019. openwebtext. (2019). https://github.com/jcpeterson/openwebtext

[10] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906* (2020).

[11] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. http://arxiv.org/abs/1412.6980 cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[12] Hang Li. 2011. *Learning to Rank for Information Retrieval and Natural Language Processing*. Morgan and Claypool Publishers.

[13] Ping Li, Christopher J. C. Burges, and Qiang Wu. 2007. McRank: Learning to Rank Using Multiple Classification and Gradient Boosting. In *Proceedings of the 20th International Conference on Neural Information Processing Systems* (Vancouver, British Columbia, Canada) *(NIPS'07)*. Curran Associates Inc., Red Hook, NY, USA, 897–904.

[14] 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. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692 http://arxiv.org/abs/1907.11692

[15] Wenhao Lu, Jian Jiao, and Ruofei Zhang. 2020. TwinBERT: Distilling Knowledge to Twin-Structured BERT Models for Efficient Retrieval. *CoRR* abs/2002.06275 (2020). arXiv:2002.06275 https://arxiv.org/abs/2002.06275

[16] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2017. Mixed Precision Training. *CoRR* abs/1710.03740 (2017). arXiv:1710.03740 http://arxiv.org/abs/1710.03740

[17] Sebastian Nagel. 2016. News Dataset. (2016). https://commoncrawl.org/2016/10/news-dataset-available/

[18] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *CoRR* abs/1611.09268 (2016). arXiv:1611.09268 http://arxiv.org/abs/1611.09268

[19] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-Stage Document Ranking with BERT. *CoRR* abs/1910.14424 (2019). arXiv:1910.14424 http://arxiv.org/abs/1910.14424

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[21] Przemyslaw Pobrotyn, Tomasz Bartczak, Mikolaj Synowiec, Radoslaw Bialobrzeski, and Jaroslaw Bojar. 2020. Context-Aware Learning to Rank with Self-Attention. *ArXiv* abs/2005.10084 (2020).

[22] Tao Qin, Tie-Yan Liu, and Hang Li. 2010. A general approximation framework for direct optimization of information retrieval measures. *Inf. Retr.* 13 (08 2010), 375–397. https://doi.org/10.1007/s10791-009-9124-x

[23] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *CoRR* abs/1908.10084 (2019). arXiv:1908.10084 http://arxiv.org/abs/1908.10084

[24] Shubhra Kanti Karmaker Santu, Parikshit Sondhi, and ChengXiang Zhai. 2019. On Application of Learning to Rank for E-Commerce Search. *CoRR* abs/1903.04263 (2019). arXiv:1903.04263 http://arxiv.org/abs/1903.04263

[25] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural Machine Translation of Rare Words with Subword Units. *CoRR* abs/1508.07909 (2015). arXiv:1508.07909 http://arxiv.org/abs/1508.07909

[26] Amnon Shashua and Anat Levin. 2002. Ranking with Large Margin Principle: Two Approaches. In *Proceedings of the 15th International Conference on Neural Information Processing Systems (NIPS'02)*. MIT Press, Cambridge, MA, USA, 961–968.

[27] Trieu H. Trinh and Quoc V. Le. 2018. A Simple Method for Commonsense Reasoning. *CoRR* abs/1806.02847 (2018). arXiv:1806.02847 http://arxiv.org/abs/1806.02847

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017). arXiv:1706.03762 http://arxiv.org/abs/1706.03762

[29] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. arXiv:1910.03771 [cs.CL]

[30] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise Approach to Learning to Rank: Theory and Algorithm. In *Proceedings of the 25th International Conference on Machine Learning* (Helsinki, Finland) *(ICML '08)*. Association for Computing Machinery, New York, NY, USA, 1192–1199. https://doi.org/10.1145/1390156.1390306

[31] Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. 2021. Pretrained Transformers for Text Ranking: BERT and Beyond. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining* (Virtual Event, Israel) *(WSDM '21)*. Association for Computing Machinery, New York, NY, USA, 1154–1156. https://doi.org/10.1145/3437963.3441667

[32] Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books. In *arXiv preprint arXiv:1506.06724*.