Solution of Team GraphMIRAcles in the KDD Cup 2022 Query-Product Ranking Task

Hanzhu Chen
University of Science and Technology of China
chenhz@mail.ustc.edu.cn

Zhihao Shi
University of Science and Technology of China
zhihaoshi@mail.ustc.edu.cn

Zhanqiu Zhang
University of Science and Technology of China
zqzhang@mail.ustc.edu.cn

Jie Wang*
University of Science and Technology of China
jiewangx@ustc.edu.cn

ABSTRACT

To improve the users' shopping online experience, a search engine aims to show a ranked list of items that best match a user's query intent. The Query-Product Ranking task is formulated as search relevance to rank a given query-item pair by relevant labels: exact, substitute, complement, or irrelevant (ESCI) in the Shopping Queries Dataset, a large dataset of difficult Amazon search queries and results. However, many existing pre-trained models suffer from several challenges: noise in the data, inadaptation to the product data, slow convergence, and overfitting during fine-tuning. To address these challenges, we use the following methods in three components—data preprocessing, pre-training, and fine-tuning, respectively. We use regular expressions to clean the data and preprocess the data through data splicing, keyword extraction, and key sentence extraction. Then, we adapt our model to the domain corpus by Masked Language Model (MLM) pre-training. Finally, we use ranking loss in fine-tuning to accelerate convergence. To reduce model overfitting and improve model robustness, we use Fast Gradient Method (FGM) adversarial training. Experiments demonstrate that our solution achieves an nDCG of 0.9002 on the private test dataset with a single model and can rank among the top 10 teams. By using ensemble methods, our models achieve an nDCG of 0.9028 on private test data and came fourth on the leaderboard.

KEYWORDS

product search, Roberta, pre-training, fine-tuning

ACM Reference Format:
Hanzhu Chen, Zhihao Shi, Zhanqiu Zhang, and Jie Wang. 2022. Solution of Team GraphMIRAcles in the KDD Cup 2022 Query-Product Ranking Task. In *KDD Cup 2022 Workshop: ESCI Challenge for Improving Product Search, August 17, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 4 pages.

1 INTRODUCTION

Improving the quality of search results can significantly enhance users' experience and engagement with search engines [2]. When the front row of the product list returned by the query contains irrelevant products, even a small number of irrelevant products will seriously affect the users' shopping experience and even affect the users' desire to purchase. In addition, common search tasks that only divide products into related and unrelated ones are not conducive to optimizing product sequencing results. For example, when a user searches for "iPhone 11", the iPhone charger and mobile phone case will be marked as irrelevant according to the common search tasks, but in fact, users often search for "iPhone 11" to buy chargers and phone cases. For this reason, the Query-Product Ranking task divides relevance into the following four classes to measure the relevance of items in search results:

• **Exact (E):** the item is relevant for the query, and satisfies all the query specifications [12] (e.g., a phone matching all attributes of a query "iPhone 11")

• **Substitute (S):** the item is somewhat relevant i.e., it fails to fulfill some aspects of the query but the item can be used as a functional Substitute [12] (e.g., iPhone 10 for a "iPhone 11" query)

• **Complement (C):** the item does not fulfill the query, but could be used in combination with an exact item [12] (e.g., iPhone charger for "iPhone 11" query)

• **Irrelevant (I):** the item is irrelevant, or it fails to fulfill a central aspect of the query [12] (e.g., television for a "iPhone 11" query)

In the Query-Product Ranking task, what we need to do is to rank the product list of each query, with 'exact' at the top, then 'substitute' and 'completion', and finally 'irrelevant', to give users the best search experience. However, many existing pre-trained models suffer from several challenges. The performance of the pre-trained model may be limited by the noise in the data and the inadaptation of pre-trained models to the product data. For example, the data contains a large number of HTML tags and special meaningless characters, which will directly affect the tokenizer of the pre-trained model because it does not match the vocabulary. At the same time, for product data, there is very little in the original training corpus of the pre-trained model, so the model will not adapt to product data. In addition, the fine-tuning speed of pre-trained models is slow, and because the fine-tuning data is much less than the pre-training data, it is easy to overfit.

To address these challenges, our solution consists of three components—data preprocessing, pre-training, and fine-tuning. Our solution is based on the basic model xlm-Roberta [3]. In the data preprocessing part, we use regular expressions to clean the data. In addition, we use YAKE [1] algorithm for keyword extraction and Textrank [11] algorithm for key sentence extraction. In the

*Corresponding author.

KDDCup '22, August 17, 2022, Washington, DC, USA

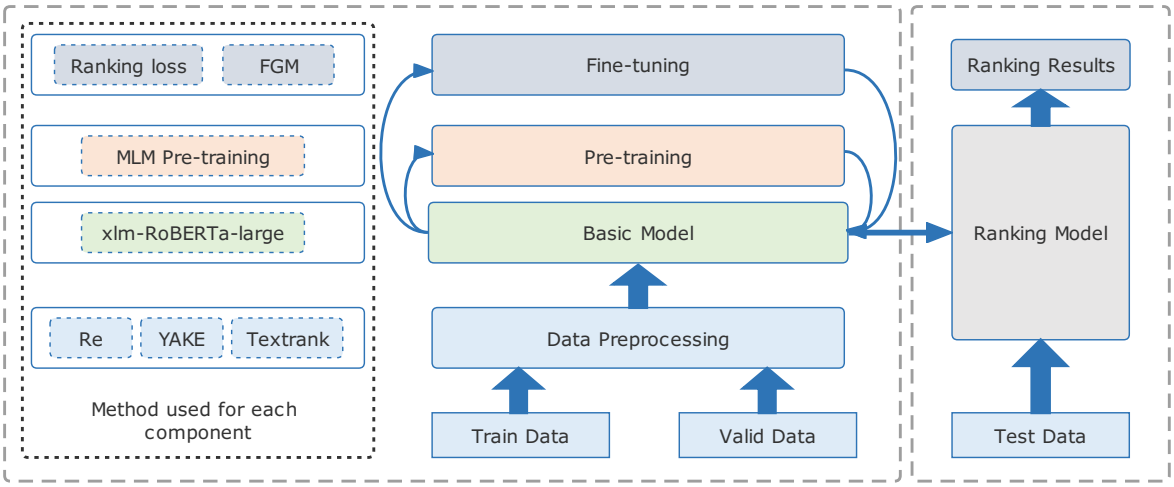

Figure 1: The overall architecture of our method. In the middle of the figure is a schematic diagram of each component. The left is the method used in each component, and their positions and colors correspond to the components in the middle.

pre-training part, we allow the model to continue MLM [7] pre-training to adapt to the domain corpus. In the fine-tuning part, we use ranking loss to accelerate convergence and use FGM [6] adversarial training to reduce model overfitting and improve the robustness of the model.

To show the effectiveness of our model, we conduct experiments on the Shopping Queries Dataset in the KDD cup 2022 - Query-Product Ranking task of ESCI challenge for improving product search. Experiments demonstrate that our solution achieves an nDCG of 0.9028 on the private test dataset.

2 RELATED WORK

Pre-trained models In recent years, pre-trained models have brought natural language understanding into a new era. BERT [4] is a classic model among pre-trained models in recent years, which stands for Bidirectional Encoder Representations from Transformers. Roberta [8] shows that training BERT for longer on more data significantly enhances performance. xlm-Roberta shows that large-scale multilingual models can significantly enhance performance on a wide range of cross-lingual transfer tasks.

Data Preprocessing The set of techniques used prior to the application of a data mining method is named as data preprocessing for data mining [5]. Data preprocessing usually includes processing missing values and keyword extraction. Many approaches are available to tackle the problematic imposed by the missing values in data preprocessing [9]. Keyword extraction includes YAKE based on statistical features and Textrank based on word graph model.

Pre-training Pre-training is the key process of pre-trained model. Recent work has shown that continued pre-training of pre-trained models on domain corpus allows models to adapt to domain tasks and enhance performance [7].

Fine-tuning Generally, in order to enhance the performance of the model in downstream tasks, the model will be fine-tuned. The core of fine-tuning model is to construct loss function. The loss function is used to evaluate the degree to which the predicted value of the model is different from the actual value. The more appropriate the loss function, the better the performance of the model. The commonly used losses on pre-trained models for fine-tuning include regression loss and ranking loss, and the choice of loss is still mainly based on the characteristics of downstream tasks.

3 MODEL ARCHITECTURE

The overall architecture of our method is shown in Figure 1. Our method contains three components—data preprocessing, pre-training, and fine-tuning. First, for the original the Shopping Queries Dataset, we first perform data preprocessing. The basic model is pre-trained on the preprocessed data to adapt the model to the domain corpus. It is worth mentioning that our basic model is xlm-Roberta-large. Then, we fine-tune the basic model on the preprocessed data to enhance the performance of the model in downstream tasks. Finally, we build a ranking model that can rank the product list corresponding to the query in the test set.

3.1 Data Preprocessing

Noise in the data can greatly affect the performance of the model on downstream tasks. Our data preprocessing consists of three steps. The first is data cleaning, using regular expressions to remove a large number of HTML tags and special meaningless characters in the data to make the data purer. Then to prevent the impact of characters on the model tokenizer, we insert spaces before and after general characters and convert the data to lowercase. Finally, due to the limitation of the length and form of the model input, we splice the columns of the given table data to extract keywords and key sentences. We use YAKE algorithm for keyword extraction and Textrank algorithm for key sentence extraction.

3.2 Pre-training

To better adapt the model to the domain corpus, we follow the paper [7]. We splice the training set and the product catalogue to

form a pre-training corpus and let the basic model continue MLM pre-training in the pre-training corpus.

We construct a total of 2 kinds of corpus, the first one is the splicing of query and product information, and the second is splicing only with product information. Finally, we find that the model pre-training on the first kind of corpora will perform better in downstream tasks.

3.3 Fine-tuning

Fine-tuning a pre-trained model for downstream tasks is generally slow and prone to overfitting to the training set. In order to fine-tune the model better, we use ranking loss to accelerate the convergence of the model, and use FGM (Fast Gradient Method) adversarial training to reduce model overfitting and improve the robustness of the model.

3.3.1 Ranking Loss. There are a total of four classes in the product list corresponding to a query. For the commonly used regression loss, the scores of the four classes of products corresponding to a query can only be fitted to four fixed constants. For example:

$$\begin{aligned} q &= \text{“iPhone 11”} \\ p_1 &= \text{“iPhone 11”},\ y(q,p_1) = 1 \\ p_2 &= \text{“iPhone 10”},\ y(q,p_2) = 2/3 \\ p_3 &= \text{“charger”},\ y(q,p_3) = 1/3 \\ p_4 &= \text{“television”},\ y(q,p_3) = 0 \end{aligned} \tag{1}$$

Where q means query, then p is the product corresponding to q, and $y(q,p_i)$ is the label of p_i corresponding to q, 1, 2/3, 1/3, and 0 correspond to Exact, Substitute, Complement, and Irrelevant respectively. Common regression loss is defined as:

$$L_{Re} = \sum_{i=1}^{N} |(f(q,p_i) - y(q,p_i))| \tag{2}$$

Where $f(q,p_i)$ is the output score of the model for the input of q and p_i. N is the total number of query-product pairs selected from the training set.

However, this linear division may not be reasonable, so we can use ranking loss. For the above example, the output score requirement of ranking loss is:

$$f(q,p_1) > f(q,p_2) > f(q,p_3) > f(q,p_4) \tag{3}$$

Ranking loss does not require the output scores of different classes of products corresponding to a query to be fitted to different constants. Compared with regression loss, ranking loss has less stringent requirements for model output scores, and this can accelerate the convergence of the model. The ranking loss is defined as

$$L_{Ra} = \max\left(\{0\} \cup \left\{f(q,p_j) - f(q,p_i) \middle| y(q,p_i) > y(q,p_j)\right\}\right) \tag{4}$$

But the max function is not easy to optimize, we use LogSumExp to approximate the max operator. The LogSumExp is shown in the following Equation.

$$\max(x,y,z,\ldots) = \lim_{k\to+\infty} \frac{1}{k} \ln\left(e^{kx} + e^{ky} + e^{kz} + \ldots\right) \tag{5}$$

Finally, our loss is defined as:

$$L = \log\left(1 + \sum_{y(q,p_i)>y(q,p_j)} e^{k(f(q,p_j)-f(q,p_i))}\right)/k \tag{6}$$

3.3.2 FGM adversarial training. Following paper [10], adversarial training can be written in the following format:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\Delta x\in\Omega} L(x+\Delta x, y;\theta)\right] \tag{7}$$

Where $\mathcal{D}$ represents the training set, x represents the input, y represents the label, θ is the model parameter, $L(x+\Delta x, y;\theta)$ is the loss of a single sample, Δx is the adversarial perturbation, and Ω is the perturbation space.

The core of adversarial training is to calculate Δx. Increasing Δx can increase the difficulty of model training and improve the robustness of the model, but Δx cannot be too large, which will reduce the performance of the model in downstream tasks. The FGM method defines Δx as:

$$\Delta x = \epsilon \nabla_x L(x,y;\theta) \tag{8}$$

Where ϵ is a hyperparameter, we find during the experiment that $\epsilon = 1.0$ works best. Then, in order to prevent Δx from being too large, it is normalized:

$$\Delta x = \epsilon\ \text{sign}\left(\nabla_x L(x,y;\theta)\right) \tag{9}$$

4 RESULTS

In this section, we demonstrate the effectiveness of three components of our solution. The evaluation indicators in the experiment all adopt the nDCG calculation method provided by the competition organizing committee. In our case, we have 4 degrees of relevance (rel) for each query and product pair: Exact, Substitute, Complement, and Irrelevant; where we set a gain of 1.0, 0.1, 0.01, and 0.0, respectively.

We randomly sample 10000 queries from the officially announced train data as the valid data. The hyperparameters in our method are as follows: Max length: 256, Lbda: 1, Gradient accumulation steps: 16, Num epochs: 4, Batch size: 1, Sample size: 8, and Learning rate: 1e-5.

The results for different model settings are shown in Table 2. Briefly speaking, the xlm-Roberta-large model containing all components achieves the best nDCG results. Furthermore, Table 1 illustrates the effectiveness of our components. Specifically, when the basic model does not contain the configuration of the components and is only training with the basic model and regression loss, the nDCG of the model only reaches 0.8958. When data preprocessing is then add to the basic model, the model's nDCG increases to 0.8963. It is worth emphasizing that the addition of pre-training makes the performance of the model more obvious, and the nDCG of the model reaches 0.8993. Finally, the addition of FGM and ranking loss also achieves a good improvement, and the final model's nDCG on the valid data set reaches 0.9018.

As shown in Table 3, the single model achieves an nDCG of 0.9002 on the private test set, and this performance can also rank in the top 10 on the leaderboard. Then we perform multi-fold training on the model, and select models with the same training configuration

Table 1: Experiment results of different valid-part

Model	Valid-part nDCG										nDCG-fix
	0	1	2	3	4	5	6	7	8	9	
$model_1$	0.8974	**0.8992**	**0.8948**	0.9014	**0.8981**	**0.9025**	0.8961	**0.8988**	0.8967	**0.8997**	0.8962
$model_2$	**0.8982**	0.8980	0.8945	**0.9024**	0.8980	0.9012	**0.8977**	0.8984	**0.8973**	0.8981	**0.8964**

Table 2: Experiment results of different model settings. Re: Regression loss; Dp: Data preprocessing; Pt: MLM pre-training; Ra: Ranking loss; FGM: FGM adversarial training.

Basic model	Model settings	nDCG (valid)
xlm-Roberta-large	Re	0.8958
xlm-Roberta-large	Dp + Re	0.8963
xlm-Roberta-large	Dp + Pt + Re	0.8993
xlm-Roberta-large	Dp + Pt + Ra	0.9012
xlm-Roberta-large	Dp + Pt + Ra + FGM	**0.9018**

Table 3: Experiment results of the single model and the ensemble model

Model	nDCG (private test)
Single Model	0.9002
Ensemble Model	**0.9028**

at different checkpoints. Finally, by using ensemble methods, the ensemble model achieves an nDCG of 0.9028 on private test data and came fourth on the leaderboard.

5 DISCUSSION

It is worth mentioning that during our experiments, we find that the final nDCG performance of the model mainly depends on the test data, especially when the test data is small relative to our valid data. We randomly select part of our valid data set, and a total of 10 valid-part data with the same number of queries as the private test set is collected. The experimental results are shown in Table 1. We find that the nDCG values of the models on these 10 datasets are quite different, and the nDCG fluctuates greatly between different models, making it difficult to evaluate the performance of the two models. However, an excellent model should be insensitive to the distribution of test data. Its mean value on all test data should be relatively large and its standard deviation should be relatively small. Therefore, we propose a more reasonable calculation method nDCG-fix: take the mean avg and standard deviation std of the nDCG of the 10 valid-part datasets, and the final nDCG-fix = $avg - std$.

6 CONCLUSIONS

In this paper, we introduce our solution for the KDD cup 2022 - Query-Product Ranking task of ESCI challenge for improving product search. In our solution, we integrate three components—data preprocessing, pre-training, and fine-tuning. Experiments demonstrate that all three components of our method are effective, and our solution finally ranks fourth on the leaderboard. It is worth emphasizing that the addition of pre-training makes the performance of the model more obvious than other components. At the same time, we experimentally find that the calculation of nDCG on the current private test set may be unreasonable. We propose a modified nDCG calculation method nDCG-fix to better evaluate model performance.

REFERENCES

[1] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. YAKE! Keyword extraction from single documents using multiple local features. *Information Sciences* 509 (2020), 257–289.

[2] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K Reddy. 2022. ANTHEM: Attentive Hyperbolic Entity Model for Product Search. *WSDM* 22 (2022), 21–25.

[3] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116* (2019).

[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] Salvador García, Julián Luengo, and Francisco Herrera. 2015. *Data preprocessing in data mining*. Vol. 72. Springer.

[6] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[7] 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. *arXiv preprint arXiv:2004.10964* (2020).

[8] 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* (2019).

[9] Julián Luengo, Salvador García, and Francisco Herrera. 2012. On the choice of the best imputation methods for missing values considering three groups of classification methods. *Knowledge and information systems* 32, 1 (2012), 77–108.

[10] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[11] Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*. 404–411.

[12] Chandan K Reddy, Lluís Màrquez, Fran Valero, Nikhil Rao, Hugo Zaragoza, Sambaran Bandyopadhyay, Arnab Biswas, Anlu Xing, and Karthik Subbian. 2022. Shopping Queries Dataset: A Large-Scale ESCI Benchmark for Improving Product Search. *arXiv preprint arXiv:2206.06588* (2022).