TFKD Solution for KDD Cup 2022 Amazon Product Search

Cheng Hsu
hsuchengmath@gmail.com
National Cheng Kung University

ABSTRACT

In this competition, participants are provided with large scale Amazon product search data, and the data contains three languages, English, Spanish and Japanese. In order to make the ranking evaluations more precisely between the user query intent and products. The competition break down relevance into the following four classes Exact, Substitute, Complement, Irrelevant (ESCI) which are used to measure the relevance of the items in the search results.

In this paper, we propose a robust multilingual model to improve the quality of search results, its name is Text-Feature-extraction-Knowledge-Distillation (TFKD) Our model can be mainly divided into three parts, including text features extraction, self-knowledge distillation and model ensemble. In text features extraction, We propose to use BM25 and PageRank to extract products text features under the N-gram. Because we cut product document into small fragments, it can avoid biased extraction of text features. To precisely extract features we not only consider the relevance scores between user query and product documents, but also the relevance between product documents can be considered. In self-knowledge distillation, we add student models to inference can effectively improve the performance in model. In model ensemble, we use regression and classification model as component of model ensemble. We release the source code associated with this work. [1]

CCS CONCEPTS

• **Information systems** → *Retrieval models and ranking*; *Learning to rank*.

KEYWORDS

search relevance, e-commerce, semantic matching, multilingual

ACM Reference Format:
Cheng Hsu. 2021. TFKD Solution for KDD Cup 2022 Amazon Product Search. In *Proceedings of Amazon @ KDD CUP'22 Workshop (Amazon KDD CUP'22).* ACM, New York, NY, USA, 5 pages.

1 INTRODUCTION

The goal of an product search engine is to show a ranked list of items that best match a shopper's query intent. Unlike standard search engines [5, 6], product search engines cannot solely rely on the textual and semantic content of a query. This is because shoppers' queries are often ambiguous, broad, lack specific intent or have an implicit intent.

For this reason Amazon break down target label into the following four classes ESCI (Exact, Substitute, Complement, Irrelevant) which are used to measure the relevance of the items in the search results.

In this competition, the players are provided with a real-world large scale shopping query ,product data and query-product four classes ESCI labels. Additionally, these data is multilingual, as it contains queries in English, Spanish, and Japanese.[7] The players are asked to to build new ranking strategies and, simultaneously, identify interesting categories of results (i.e., substitutes) that can be used to improve the customer experience when searching for products.

In this paper, we propose extract texts features extraction strategy and modeling strategies for product search task. In texts features extraction strategy, user query and product documentation contain a lot of information. If we extract text features only use word frequency features (ex. TF-IDF), it con probably that only extract partial. It causes loss partial user intent information.

Therefore, we extract text fragments of user query and product documents under N-gram. It means that we cut user query and product documents into small fragments. User intent is distributed in these fragments and we will extract product text features by each user query fragments.

Specifically, we use BM25 [10] to calculate the relevance score between fragments of user query and fragments of product document. Of course, we can directly use this relevance score to filter which fragments of product documents as product features and experiment also tells us that it is indeed effective.

However, after some data mining exploration, it was found that the product features extracted by the above method, if user queries contains too much noise, it may cause the extracted features to be full of noise. Therefore, we expect that the fragments of product documents can consider the relevance scores of each other, that is, When there is noise in fragments of user query, the extracted information will not completely depend on the query fragments, but will also depend on the other fragments of the product documentation. To combine query-product, product-product information, we use PageRank [1] to update score of each product documents fragments, and extract product features based on relevance scores.

In modeling strategies part, we use cross encoder [8, 9] as base model. We have two improvement directions. The first direction is to strengthen the model inference ability by different labels. The second direction is to improve the diversity of model inference, that is, expect that model inference is not too single. For the first direction, we will let the tasks be set to regression tasks and classification tasks When training regression tasks, we design two stage training by trick label. it is easier to strengthen the inference ability of different labels simultaneously. When the classification task is trained, it is easier to infer Exact label that can obtain higher scores. For the second direction, we use self-knowledge distillation [12] model training with threshold selection to make it easier for student models to learn diverse inference. Although these two-direction strategies cannot be significantly show in the performance of a single model, it achieve performance improvement after model ensemble.

[1]https://github.com/ChengHSUHSU/KDD_Cup2022_AmazonProductSearchESCI

Table 1: Summary of notations

Notation	Meaning
q	user query.
p	product document.
w_i^q	i-th word of q, where q is user query.
w_j^p	j-th word of p, where p is product document.
F^q	User query fragments, where q is user query.
F^p	Product document fragments, where p is product document.
f_i^q	i-th fragment of F^q, where q is user query.
f_j^p	j-th fragment of F^p, where p is product document.
$r_{i,j}^{q,p}$	query-product relevance score of fragment f_i^q and f_j^p, it is calculated by BM25.
$r_{j,k}^{p,p}$	product-product relevance score of fragment f_j^p and f_k^p, it is calculated by BM25.
$s_{i,j}$	query-product relevance score of fragment f_i^q and f_j^p, it is updated by $r_{i,*}^{q,p}$, $r_{*,*}^{p,p}$ and PageRank.
t	Self knowledge distillation threshold.

2 METHODOLOGY

In this section, we first discuss the problem statement and then explain the different components of our architecture and detail its training strategies. The summary of notations show on Table 1.

2.1 Problem Statement

Given a user specified query Q and a list of matched product List P, the goal of this task is to rank the list of products in decreasing order of relevance, i.e., first the Exact matches, then Substitutes, followed by Complements, and Irrelevants at the end. The maximum number of products per query is 40, and at least one is guaranteed to be non irrelevant (either Exact, Substitute, or Complement).

2.2 Text Features Extraction

The user query and the product documents contain many different pieces of information. In order to avoid biased extraction of this information. We use N-grams to extract user query fragments and product document fragments by window-cutting, where N will be discussed in Experiment.

$$q = (w_1^q, w_2^q, ..., w_m^q) \tag{1}$$

$$F^q = ([w_1^q, ..., w_N^q], [w_{N+1}^q, ..., w_{2N}^q], ...) = (f_1^q, f_2^q, ...) \tag{2}$$

$$p = (w_1^p, w_2^p, ..., w_k^p) \tag{3}$$

$$F^p = ([w_1^p, ..., w_N^p], [w_{N+1}^p, ..., w_{2N}^p], ...) = (f_1^p, f_2^p, ...) \tag{4}$$

Next, we use BM25 [10] to calculate the relevance score of each user query fragments to matched product document fragments. We utilize BM25 [10] to compute the relevance scores between product document fragments.

$$r_{i,j}^{q,p} = BM25(F^q, F^p)|_{i,j} \tag{5}$$

$$r_{j,k}^{p,p} = BM25(F^p, F^p)|_{j,k} \tag{6}$$

For the relevance score between product document fragments, we can think of it as a fully connected graph with weights, where each node is an product document fragments and the weight of the edge is the relevance score. Moreover, we can regard the relevance score of the given user query fragments and product documents fragments as the node weight of this graph. Next, we use PageRank to iteratively update the score for each node (fragments of the product document). We filter product document fragments as product features by higher scores.

$$weights = (r_{j,k}^{p,p}, r_{i,j}^{q,p}) \tag{7}$$

$$s_{i,j} = PageRank(P_{\mathcal{G}}, weights) \tag{8}$$

where given i-th user query fragment f_i^q and $P_{\mathcal{G}}$ is fully connected graph of matched product documents, $r_{j,k}^{p,p}$ are edge weights of the graph and $r_{i,j}^{q,p}$ are node weights of j-th product document fragment f_j^p. $s_{i,j}$ are updated query-product relevance score of j-th product document fragment, for all j.

All in all, for each fragments of the user query, we can generate a fully connected graph with weights, and their difference is mainly in the weight of the nodes, and then update the score of each product document segment by PageRank [1], and decide which product fragments as product features by higher scores.

2.3 Cross Encoder

Using a cross-encoder sentecne-bert1, sentecne-bert2 as a re-ranker usually achieves superior performance. cross-encoder computes the relevance score score(q, p), where the input is the concatenation of q and p with a special token [SEP]. Subsequently, the [CLS] representation of the output is fed into a linear function to compute the relevance score. Cross-encoder enables cross interaction of q and p in each transformer layer and thus is effective yet inefficient.

In addition, we divide the methods of prediction into two types, namely regression model and classification model. The regression model mainly uses output value of the cross encoder to rank matched products. The classification model mainly uses the label probability of the label in the form of weighted sum obtained output values and rank matched products.

2.4 Self Knowledge Distillation

In order to apply knowledge distillation [12] on a current training model, we need to obtain soft target probabilities in cross encoder classification for all four classes.

However, the knowledge learned by the teacher model does not perform well on all user queries. In order to avoid Student model learns failure knowledge. We set the threshold value t, when metric score of use query (nDCG) is lower than the threshold t, we do not use the soft probability given by the teacher model, and then use soft probability generated by label smoothing. Finally, we use the collected soft probabilities as target labels to train the student model and the student model architecture is same with teacher model.

2.5 Training Strategies

To improve robustness and performance of model, we adopt some training strategies.

Exponential Moving Average :

Our model uses EMA [4] to smooth the trained parameters. Evaluations that use averaged parameters sometimes produce significantly better results than the final trained values. Formally, we define the smoothed variables and trained variables as θ_s and θ_t, EMA decay weight as: η. for each training step, we update θ_s by:

$$\theta_s \leftarrow \eta\theta_s + (1-\eta)\theta_t \tag{9}$$

Regression Model label Setting :

In order to make the model more effective in distinguishing the four labels E, S, C, I, we divide the regression training into two stages. The first stage is to treat all original E label as 1.0 and S, C, I label as 0.0. The second stage is to treat all original E label as 1.0, and S label as 0.4, treat the C, I label as 0.0 and retrain it with the regression model. The loss function used for regression model training is MSE.

In first stage training, we found that the model was able to effectively distinguishes E label and other.

In second stage training, we found that the model was able to distinguishes S label and C, I label. the performance of E and other label slightly trade-off reduce.

Classification Model :

In order to make the model more effectively distinguish the four types of labels E, S, C, I, we divide the labels into three types, E, S, CI The loss function used for classification model training is Cross Entropy.

Ensemble Modell :

Since predicted value range by the regression and classification model is different. Before the model ensemble, we use Min-Max-Normalization [11] to ensure that the range of the value range is between 0 and 1. Next, average the values of each query and product in equal proportions as the updated predicted value.

3 EXPERIMENTS

3.1 Metric Setting

The task performance will be evaluated using Normalized Discounted Cumulative Gain (nDCG). This is a commonly used relevance metric in the literature. Highly-relevant documents appearing lower in a search results list should be penalized as the graded relevance is reduced logarithmically proportional to the position of the result. In our case, we have 4 classes labels for each query and product pair: Exact, Substitute, Complement, and Irrelevant, and the competition set gain values of 1.0, 0.1, 0.01, and 0.0, respectively.

3.2 Model Performance

In this experiment, we compare our model performance against the baselines on the task of search model. Our language representation model is XLM-Roberta-Large [2, 3]. The performance is evaluated on the metrics of nDCG and label gains follow official setting. The results of our experiments show on Table 2.

Baseline :

(1) CLF: Four label classification in cross encoder, The model rank product list by L_E*P_E + L_S*P_S + L_C*P_C + L_I*P_I, where P_E, P_S, P_C, P_I is label probabilities of model. The class weight is the gain of four labels, where the weight of E label L_E is 1.0, S label L_S is 0.1, C label L_C is 0.01 and I label L_I is 0.0.
(2) REG: Regression in cross encoder, The model rank product list by model output.
(3) CLF+REG: It is CLF, REG as component of ensemble model and their weights are same.

3.3 Texts Feature Extraction

This experiment mainly shows the performance of the model can be significantly improved after using our proposed Texts Feature Extraction method. From the experiments, we can find that the overall performance has been significantly improved. In addition, we also tuning N_q, N_p, we found that when N_q=3, N_p=8 has better performance, where N_q, N_p are N-gram of user query and product documents. The results of our experiments show on Table 3.

3.4 Self Knowledge Distillation

The experiment only conduct on classification case. From the experiment, we can find that the performance of student model is generally slightly worse than the teacher model. In addition, when the knowledge distillation threshold t set 0.8, it can perform better performance than other settings. Then, we build ensemble model by teacher model and student model and their weights are same. We find that the ensemble model achieves better performance than the teacher model, we think student model can provide different inference directions, so the ensemble model can get better performance. The results of our experiments show on Table 4.

3.5 Ensemble Model

In the experiment, we build ensemble model by regression and classification model. From the experimental results, we can find that the regression model, although overall score is not higher than classification model, but ranking performance between S and CI is significantly better than the classification model. From here

Table 2: Performance comparison of TFKD against the baselines on the task of search relevance ranking in three locale on four classes E-S-C-I. The evaluation metric is mainly to use the nDCG introduced earlier, in order to more clearly analyze the ranking status of each classes and we additionally evaluate these three pair: E-S, E-CI, and S-CI. It is mentioning that, for the convenience of observation, for these three additional evaluations, we set gain as 1 and 0.

	English(US)				Spanish(ES)				Japanese(JP)			
Label	**TFKD**	**CLF**	**REG**	**CLF+REG**	**TFKD**	**CLF**	**REG**	**CLF+REG**	**TFKD**	**CLF**	**REG**	**CLF+REG**
E-S	0.835	0.829	0.824	0.829	0.831	0.819	0.817	0.823	0.839	0.831	0.829	0.831
E-CI	0.891	0.884	0.882	0.886	0.893	0.887	0.884	0.889	0.891	0.887	0.886	0.888
S-CI	0.823	0.798	0.816	0.819	0.821	0.784	0.812	0.815	0.821	0.768	0.819	0.817
ESCI	0.894	0.885	0.884	0.888	0.896	0.888	0.889	0.890	0.897	0.891	0.892	0.892

Table 3: Text features extraction experiment

Component	E-S	E-CI	S-CI	ESCI
None	0.825	0.880	0.820	0.884
N_q=2,N_p=8	0.834	0.891	0.823	0.891
N_q=3,N_p=8	0.836	0.891	0.821	0.896
N_q=4,N_p=8	0.836	0.890	0.823	0.895
N_q=2,N_p=9	0.833	0.889	0.822	0.890
N_q=3,N_p=9	0.835	0.892	0.820	0.896
N_q=4,N_p=9	0.833	0.891	0.822	0.893

Table 4: Self-knowledge distillation experiment. we set the range of knowledge distillation threshold t is in (0.0, 0.7, 0.8). We don't use teacher model output as soft label when nDCG score of given query is lower than t and we use label smoothing as soft label. When t is 0.0, it means we only use teacher model output as self label.

Component	E-S	E-CI	S-CI	ESCI
teacher	0.831	0.889	0.823	0.892
student(t=0.0)	0.826	0.884	0.817	0.886
student(t=0.7)	0.827	0.885	0.817	0.889
student(t=0.8)	0.828	0.885	0.816	0.890
ensemble(t=0.0)	0.833	0.889	0.819	0.893
ensemble(t=0.7)	0.835	0.891	0.821	0.895
ensemble(t=0.8)	0.836	0.891	0.821	0.896

Table 5: Ensemble model experiment

Component	E-S	E-CI	S-CI	ESCI
TFKD(Reg)	0.834	0.889	0.823	0.893
TFKD(Clf)	0.836	0.890	0.812	0.894
TFKD(Reg+Clf)	0.836	0.891	0.821	0.896

we can infer that the classification model is easier to distinguish E with S, C, I labels, but the ranking performance between S, C, I is not well. However, the regression model can have a certain performance in the ranking of each labels. Therefore, in the part of the ensemble model, we use regression and classification model as the components of ensemble model, which can improve the performance of the overall model. The results of our experiments show on Table 5, where Reg is regression model, Clf is classification model and Reg+Clf is ensemble model of Reg, Clf.

4 CONCLUSION

In this paper, we propose methods for text features extraction, self-knowledge distillation, model training strategies and model ensemble and make the model robustness have significant performance. The proposed text features extraction method not only can consider the relevance between user query the product documents text, but also consider the relevance between product document text to avoid obtaining biased text features. The proposed self-knowledge distillation method does not improve the performance of a single model, but when model ensemble, the overall model performance is significantly improved. This is because the inference methods learned by the student model are different from the teacher model. Finally, we use regression and classification model as component of model ensemble, and overall performance was improved. In future work, we will try to extract text features using GNN methods, and we will also try to use contractive learning.

REFERENCES

[1] Brin and L. Page. 1998. The anatomy of a large-scale hypertextual web search engine.. In *Comput. Netw. ISDN Syst., 30 (1-7).* 107–117.

[2] 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. In *arXiv preprint arXiv:1911.02116.*

[3] Naman Goyal, Jingfei Du, Myle Ott, Giri Anantharaman, and Alexis Conneau. 2021. Larger-Scale Transformers for Multilingual Masked Language Modeling.

[4] Sangchul Hahn. [n.d.]. A new approach of moving average method in time series analysis. In *In 2013 Conference on New Media Studies (CoNMedia), pages=1–4, year=2013.*

[5] Harendra Guturu Alex Ksikes Preslav Nakov Michael A Wooldridge Marti A Hearst, Anna Divoli and Jerry Ye. 2007. BioText Search Engine: beyond abstract search. Bioinformatics. 2196––2197.

[6] Qais Saif Qassim Mo-hammed Ahmed Subhi Mohammad Najah Mahdi, Abdul Rahim Ahmad and Taofiq Adeola Bakare. 2022. A Survey on the Use of Personalized Model-Based Search Engine. In Proceedings of International Conference on Emerging Technologies and Intelligent Systems. In *Mostafa Al-Emran, Mohammed A. Al-Sharafi, Mohammed N. Al-Kabi, and Khaled Shaalan (Eds.). Springer International Publishing, Cham.* 88–98.

[7] 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.

[8] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing.*

[9] Nils Reimers and Iryna Gurevych. 2020. Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

[10] Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. In *Foundations and Trends® in Information Retrieval 3(4):333-389*. 333–389.

[11] Kishore Kumar Sahu S. Gopal Krishna Patro. 2015. Normalization: A Preprocessing Stage.

[12] Heeyoul Choi Sangchul Hahn. 2019. Self-Knowledge Distillation in Natural Language Processing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing, RANLP 2019*.