Cross-Encoder Data Annotation for Bi-Encoder Based Product Matching

Justin Chiu **Keiji Shinzato**
Rakuten Institute of Technology
Rakuten Group Inc.
`{justin.chiu, keiji.shinzato}@rakuten.com`

Abstract

Matching a seller listed item to an appropriate product is an important step for an e-commerce platform. With the recent advancement in deep learning, there are different encoder based approaches being proposed as solution. When textual data for two products are available, cross-encoder approaches encode them jointly while bi-encoder approaches encode them separately. Since cross-encoders are computationally heavy, approaches based on bi-encoders are a common practice for this challenge. In this paper, we propose cross-encoder data annotation; a technique to annotate or refine human annotated training data for bi-encoder models using a cross-encoder model. This technique enables us to build a robust model without annotation on newly collected training data or further improve model performance on annotated training data. We evaluate the cross-encoder data annotation on the product matching task using a real-world e-commerce dataset containing 104 million products. Experimental results show that the cross-encoder data annotation improves 4% absolute accuracy when no annotation for training data is available, and 2% absolute accuracy when annotation for training data is available.

1 Introduction

Product matching refers to the task of determining whether two different entries in the product catalog refer to the same real-world product. It is a core task for an e-commerce company where product catalog entries come from different sources and duplicates need to be identified and managed. There are two popular types of deep learning models that had been applied to the recent product matching work; cross-encoder (Li et al., 2020; Peeters et al., 2020) and bi-encoder (Shah et al., 2018; Tracz et al., 2020). Figure 1 depicts the architectures to demonstrate the difference between two models. When you have textual data for two products such as title pairs, the cross-encoder encodes them jointly, and the interaction between the two occurs through all encoder layers. The bi-encoder encodes both of them separately and there is no interaction between them until computing similarity.

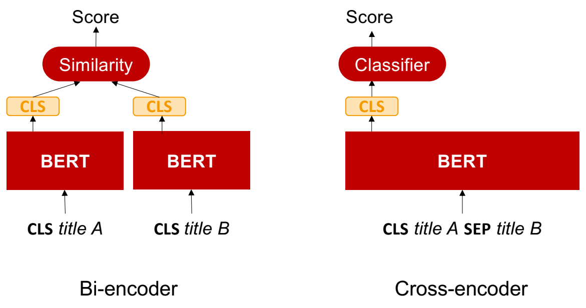

Figure 1: Bi- and cross-encoders using BERT.

The cross-encoder can capture more context due to its more complex interaction between the two inputs of data. However, because of its complexity, the model requires much more computational resources and time. A previous work (Reimers and Gurevych, 2019) reported that it takes 65 hours to find the most similar pair from a collection of 10,000 sentences using the cross-encoder (which requires us building 100,000,000 pairs for cross-encoder to process), while it only takes 5 seconds to encode all 10,000 sentences and compute cosine similarity for every possible pairs using the bi-encoder; that is 46,800 times difference in processing time. Since an e-commerce platform could easily have millions of products in its product catalog, such a volume will be a great challenge to use an approach that requires lots of computing resources, such as the cross-encoder. As a result, the bi-encoder is more viable for product matching in actual production systems.

In this paper, we propose cross-encoder data annotation, which is a technique that benefits from both model architectures. We first apply a cross-encoder on training data for a bi-encoder, and then train a bi-encoder using the data with high prediction scores from the cross-encoder. This process is a way to utilize the knowledge learned in the cross-encoder to support the bi-encoder training. This approach is inspired by knowledge distillation (Hofstätter et al., 2020), where information learned by the cross-encoder is passed to the bi-encoder through the loss function. This can also be considered as a semi-supervised approach for product matching, where the new data is unlabeled.

In this paper, we make three major contributions. First, we demonstrate that our cross-encoder data annotation is effective for the product matching task on a real-world e-commerce dataset. Selecting the subset that both cross-encoder prediction and human annotation consider positive can train a better performance model in comparison to baseline. This can happen in an e-commerce company where different existing trained models are available for research purposes and new data are coming in. When building a model with the human annotated new data, our approach is applicable.

Second, we show that even when the human annotation of the bi-encoder training data is not available, we can use the prediction result from the cross-encoder as the data annotation, and build a model that performs better than a model based on product search results. This scenario can happen when reliable annotation for the new data is not available, and we can use the prediction of the cross-encoder to replace human annotation.

Finally, from the first contribution, we get to train a better model with less training data. This is not a common practice in deep learning since more training data is usually preferred. Our analysis shows that through the cross-encoder data annotation, we focus our training data on less queries. Since the bi-encoder tries to learn the difference between the two, the cross-encoder data annotation removes extreme samples in training data, which acts similarly to noise removal.

2 Product Matching

Given a product entry, a system is required to find entries in a product corpus that represent the same product, despite that the content in the entries are different. A product entry contains a set of information for a specific product, such as title, description, image, or categories. The multiple entries for the same product can be created by different vendors using the same e-commerce system. Assuming that our product corpus C contains M product entries for N different products $C = \{p_1, p_2, p_3 ..., p_M\}$, when given a query product p_q, we want to find a target product p_t that matches the query product in the corpus. The product corpus can be the product database for a deployed e-commerce system, which means the corpus size can easily contain over tens of millions of product entries.

We consider product matching as an information retrieval problem. We learn the similarity between the product entries, and use the similarity between query product p_q and target product p_t to decide whether two product entries are for the same product. This approach does not have to retrain the model every time the number of products N changes. By increasing the N in the product corpus C, we can increase the coverage of our product matching system.

3 Real-World E-Commerce Scenario

As a solution to the real-world product matching, there are two different types of approaches. The first is an approach based on a bi-encoder and another is based on a cross-encoder. We refer to the bi-encoder as *known bi-encoder*, and the cross-encoder as *known cross-encoder*. These known models require annotated data, which can be considered as the system that already exists in the e-commerce platform or systems that researchers develop as a proof-of-concept.

Each time a new item is received, sellers on the e-commerce platform upload the product data to the product database. Since the uploaded data sometimes contain incorrect information, a business unit in the e-commerce company periodically inspects if product data is correct. These inspection results can be regarded as human annotated data. Therefore, regardless of the annotation, new data is accumulated in the e-commerce platform. To incorporate the latest product information into a production system, we want to build a *new bi-encoder* using the new data. We exploit the known bi-encoder and known cross-encoder to train the new bi-encoder effectively, which is our key contribution for this work.

In the rest of this section, we describe the known bi-encoder, the cross-encoder, and how to train the new bi-encoder using these existing models.

3.1 Known Bi-Encoder

The known bi-encoder serves as a simple search engine to create training data for the known cross-encoder and new bi-encoder. We train the known bi-encoder using triplet loss and in-batch negative, as reported in previous work (Karpukhin et al., 2020) for the open-domain Question Answering (QA) task. One key difference from the previous work is that we use one encoder to encode product title pairs although the bi-encoder for the QA task requires two different encoders, one to encode the question, and another to encode potential answer passages. We use BERT (Devlin et al., 2019) as an encoder, and regard the embeddings of `[CLS]` token as a representation of the given product title.

After training the bi-encoder, we encode our entire product corpus with the trained model, and then index them using FAISS (Johnson et al., 2019) offline. FAISS is an open-source library for similarity search that can be easily applied onto billions of vectors. After we have our entire corpus indexed, whenever we received a new query product, we can encode it with the same model and retrieve top k product titles from the FAISS index that are closest to the encoded query in the product embedding space.

3.2 Known Cross-Encoder

The known cross-encoder serves as a complex model you can built from the annotated training data to capture the most contextual information between product pairs. We employ BERT as an encoder, and put a classifier layer on the top of BERT. We convert product title pairs in the form of `[CLS] Title 1 [SEP] Title 2`. We regard the embeddings of `[CLS]` token obtained from BERT as a representation of the title pairs, and feed it to the classifier layer to judge if both titles refer the same product.

3.3 New Bi-Encoder

The new bi-encoder is the model we want to build when new data becomes available. The new data might come with reliable annotations or not, yet we still want to train model from it so our model can capture the most up to date information.

To train the new bi-encoder, we first retrieve relevant products for each query product using the known bi-encoder, and then apply the known cross-encoder to product pairs constructed from query and relevant products. For the pairs that the known cross-encoder predicts as matching pairs, we annotate positive pairs as training data. We call this approach *cross-encoder data annotation*. When using the cross-encoder data annotation, no human annotations for this training set are required, which we believe will be a convenient scenario in a real-world, large-scale setup. Similarly to the known bi-encoder, after training the new bi-encoder, we index the entire product corpus using the model and FAISS.

When building the new bi-encoder, in the ideal scenario, annotation of the training data will be accurate and available so we can simply use the human annotations to decide what are the matching pairs. We can further improve the quality of human annotation by using the known cross-encoder to annotate the same data, and then use the intersection of both sets as positive training data. This will lead to a smaller training set than human annotation. However, there could be situations where such high quality human annotations are not available. As such, we directly use the set annotated by the known cross-encoder as positive training data.

4 Experiments

Our experiments are focused on the new bi-encoder. There are two main scenarios for our experiments, depending on whether the human annotations of the new products are available. For each scenario, we compare the experimental result with and without the cross-encoder data annotation. When the human annotations of the new products are available, we intersect human annotations with cross-encoder data annotations to improve the quality of training data. When the human annotations of the new products are not available, we use the cross-encoder data annotation for model training.

In future work, we will compare the new bi-encoder with a bi-encoder model trained with all query products used for training the known bi-encoder, known-cross encoder, and the new bi-encoder. The reason why we skip this comparison is that the goal of our experiments is to see how changing the annotation of the same data such as doing intersection with other annotations can improve models.

4.1 Dataset

Our experiments are based on our in-house dataset. The dataset contains product entries that are created by sellers on our e-commerce platform, and each entry consists of product ID,[1] title and description written in Japanese. The entries that refer the same product have the same product ID. The total number of products is 104 million. We regard this

[1]More precisely, it is a global trade item number (GTIN).

Parameter	Cross-encoder	Bi-encoder
Batch size	128	128
Max seq. length	256	64
Learning rate	1e-05	1e-05
Temperature	n/a	1.0
Warmup rate	0.1	n/a
Vocabulary size	32,000	32,000
Max epoch	10	20

Table 1: Hyper-parameters for each model.

dataset as the product corpus. We only use the title in the product entry for model training. This avoids the mismatch where some sellers provide rich product descriptions while others provide limited or no descriptions.

4.2 Model

As encoders for cross- and bi-encoder models, we adopt BERT base (Devlin et al., 2019) in Japanese from Huggingface[2] and use its tokenizer to segment product titles into sub-words. We convert all characters in the titles into full-width before the segmentation. The average length of a product title is 27 sub-words.

The hyper parameters we used for training are reported in Table 1.

4.3 Training

To train the known bi-encoder, known cross-encoder, and new bi-encoder, we use three unique sets of 110K products as query products. We selected these query products from the results of business operation provided from a business unit in the company. Each query product has a product ID that the business unit assigned through the operation. There is no overlap of the product IDs within the three sets of the query products. We use 100K of the products as a training set and 10K products as a development set.

4.3.1 Known Bi-Encoder

For each entry in the first set of 110K query products, we randomly select a product title in the corpus having the same product ID, and we use these selected pairs as positive pairs for training. For negative pairs, we adopt the in-batch negative strategy proposed by Karpukhin et al. (2020).

[2]https://huggingface.co/cl-tohoku/bert-base-japanese

4.3.2 Known Cross-Encoder

The known cross-encoder is trained on the second set of 110K query product titles. We search these 110K product titles on the product corpus with the known bi-encoder to collect top 50 products for each query. This creates 5.5 million pairs of product titles, including both positive and negative pairs. Five million pairs are used as training data and the rest as development data. We randomly select 2.56 million pairs from the five million pairs and build the known cross-encoder. The benefit for constructing training pairs following this approach is to incorporate hard negative pairs into the training. Since hard negative pairs are similar in text, but do not refer to the same product, we can expect that the model learns the difference between products such as differences in product color.

4.3.3 New Bi-Encoder

The new bi-encoder is built on the third set of 110K query product titles and is the main focus of this paper. The set is similar to the situations where new products come into the product corpus. We want to build the bi-encoder using these new data. The new bi-encoder is built differently depending on whether the annotation for the new data is available.

When Available For each query product, we search the product corpus with its title using the known bi-encoder and collect the top 50 products. After collecting the top 50 products, we first use our human annotation to create training pairs based on all matching according to the annotation between the query product title and the retrieved product titles. This setup creates 113,374 training pairs that we called *human* annotated training pairs. We then use the known cross-encoder to predict all the product pairs in the top 50 retrieved results. For all the pairs that the known cross-encoder predicted as a match, we called them *cross-encoder* annotated training pairs, which have 107,059 pairs. Lastly, we perform an intersection on the human annotated training pairs and the cross-encoder annotated training pairs to utilize the knowledge in both pairs.

As a result, we obtain 84,688 training pairs which we called *intersection* pairs. The bi-encoder trained with the intersection pairs is our proposed approach for this scenario.

When Not Available The bi-encoder trained with the cross-encoder annotated training pairs is our proposed approach.

Training data	Accuracy
Human (baseline)	0.7356
Intersection (ours)	0.7575

Table 2: Results when new data annotation is available.

Training data	Accuracy
Top 1 product pairs (baseline)	0.6985
Cross-encoder (ours)	0.7423

Table 3: Results when new data annotation is not available.

Training data	# of query products
Human	46,160
Cross-encoder	42,186
Intersection	36,872

Table 4: Number of query products in different training data.

4.4 Baselines

We prepare different baselines depending on whether the annotation for the new data is available. When the annotation is available, the model trained with the human annotated pairs is a baseline. On the other hand, when the annotation is not available, we first search the query product titles on product corpus with the known bi-encoder. We still collect the top 50 retrieved results. However, since the annotation is not available, we only use all the top 1 retrieved results for every query product to create product pairs for training the baseline. Since we have one pair per query, we will have 100,000 training pairs in this setup. We call this dataset *top 1 product pairs*.

4.5 Evaluation

We select 9,991 products from the operation results, and use them as evaluation data. The data has less than 2% overlap between each of the 110K product sets described above. This set can help us understand whether our models can be effective to the product that is not in our training data.

For the evaluation, we chose a model with the lowest loss value on the development set. The evaluation measure is the top 1 accuracy of the search result. We check if the top 1 retrieved product and the query product are the same product.

4.6 Results

Tables 2 and 3 show the results for using our cross-encoder data annotation. When the annotation of training data is available, we can further refine the quality by conducting an intersection between the human annotation and the cross-encoder data annotation. Since the human annotated training pairs is larger than the intersection pairs, we can also observe that more training data does not guarantee better performances. When the annotation is not available, we can see that the model with the cross-encoder data annotation outperforms the baseline, which relies on the retrieval approach, to form positive training data. Our cross-encoder annotation result is also slightly better than the human annotation result. This could be caused by the randomness on the training data, or our cross-encoder annotation is focused on lesser queries, which will be discussed in the section 5.1.

5 Analysis

Since our experiments use the identical modeling approach, we focus on understanding the composition of training data and how such differences in training data affects the prediction performance.

5.1 Analysis of Training Data

We studied how many matching pairs are created for each query. In the retrieval phase, since we use every matching pair available, it is possible to have multiple matching pairs for a single query product. Table 4 shows the number of query products in each set. Note that although we have 100K query products to search the corpus to form query pairs, we only have positive product pairs constructed from 46K product queries in the human annotation. This is because there are query products that the known bi-encoder does not return matching products in the top 50 results. Those query products cannot form any training pairs. As such, even if the human annotation training data have 113K pairs, they are from 46K distinct products. The number of query products dropped to 42K for the cross-encoder training pairs, and further lowered to 36K for the intersection of both sets. This reduced number of queries makes training process focus on the product pairs that are relatively easy to judge as the same product. In other words, removing some extreme training instances is effective to train better models, and it might work similar to removing data noise. A set

Title 1	**Title 2**
【１ケース】ファンタグレープ　１６０ｍｌ缶	【送料無料】　コカ・コーラ　ファンタグレープ　１６０ｍｌ缶　３０入　果汁ブレンドのフルーティーなおいしさ　果汁１％配合　【コカコーラからお客様へ直接お届けします】【代引不可】
[1 case] Fanta Grape 160 ml Can	[Free Shipping] Coca-Cola Fanta Grape 160 ml Can 30 Packs Fruity Taste of Fruit Juice Blend Contain 1% Fruit Juice [From Coca-Cola to you directly] [Cash on delivery is not available]

Table 5: Example of a positive title pair removed by the intersection process (top) and its translation (bottom).

Type	**# of pairs**
Both incorrect	2,227
Only human correct	196
Only intersection correct	414
Both correct	7,154

Table 6: Number of title pairs in the test set for each correct/incorrect type.

of less diverse training data could model the most common difference between the two product titles and contribute to better performance.

Table 5 shows an example of a positive product title pair removed by the intersection process. Even though they refer to the same product, and the identical product name (*i.e., Fanta Grape*) also shows up in both titles, we can observe the differences in both titles. In addition to the product name, the product title 2 contains extra descriptions such as a manufacturer name, a quantity, and shipping information. Learning from pairs with unbalanced product information might impact the effectiveness of features such as manufacturer names and quantities, which affects the overall performance of the trained model. We can expect that the intersection process removes such unbalanced pairs from the training data.

5.2 Analysis of Prediction Results

Table 6 shows the numbers of title pairs in the test set when we categorize the pairs according to the judgment results of the human and intersection models. From the table we can see that the number of the pairs in “Only intersection correct” is two times larger than that in “Only human correct.” To see what kind of product pairs in the test set the

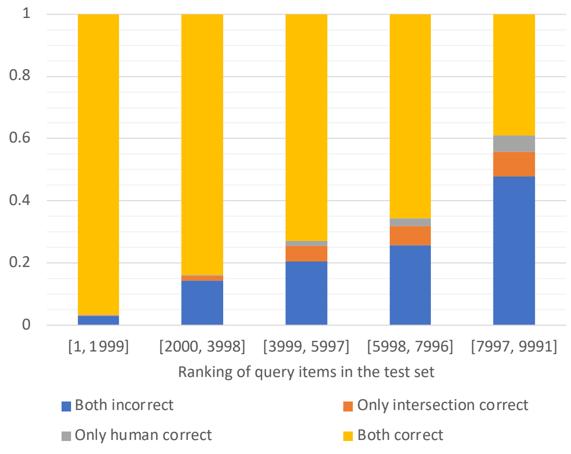

Figure 2: Ratio of correct and incorrect predictions of each model at each range. The x-axis shows the ranges of ranking when we sort title pairs in ascending order according to the distance returned from FAISS for the human annotation model. The title pair with shorter distance is at higher ranking.

cross-encoder data annotation effectively works, we investigate squared Euclidean (L2) distance between two product titles returned from FAISS for our human annotation model. We first sort the title pairs in ascending order according to the distances returned from the human annotation model, and then categorize the title pairs into five ranges equally. After that, for each pair in each range, we check if the prediction of the intersection model is correct.

Figure 2 shows the ratio of correct and incorrect predictions of each model at each range. From the figure, we can observe that the portion labeled “Only intersection correct” gets larger as the ranking gets lower, and is always larger than the portion of “Only human correct.” This means that our intersection model has improvement on the subset of pairs that are far in distance in the embedding space created by the human annotation model.

5.3 Analysis for Continuous Improvement

As new data keep coming into the system in the real-world e-commerce scenario, we can continuously improve our models by integrating the training data of new model into the training data of known models. However, given the training data of updated cross-encoder model has more negative pairs comparing with positive pairs, this updated might have class imbalance issue where it will be mostly negative pairs. This can be addressed by setting a threshold during training data updates to ensure certain percentage of pairs that need to be positive pairs, and ensure the proper class balance in the updated training data.

6 Related Work

Product matching is a fundamental task for an e-commerce platform. Given the text in product entries, we want to match it to a specific product so duplicates can be identified. Earlier works tried to solve it by extracting defined product attributes and perform matching based on the extraction results (Mauge et al., 2012; Ghani et al., 2006).

Recently, more efforts have shifted toward focusing on text (Shah et al., 2018; Tracz et al., 2020). This avoids the need of doing attribute extraction and can directly be used on product titles and descriptions. There are two main directions for these efforts. One is considering the product matching task as an extreme classification problem, and another considering it as a zero-shot learning problem. As an extreme classification problem (Shah et al., 2018), the paper built a multi-class classifier that categorizes each input product information into a class, whereas each class represents a different product. The challenge is how to manage a multi-class classifier with several millions of classes, and the need to retrain the classifier every time a new product has entered the database. On the other hand, when considering a zero-shot learning problem, it focuses on learning the difference between the product texts (Tracz et al., 2020; Xiong et al., 2020). This makes it easier to apply the model on new products and more ideal in a production scaling environment. Both cross-encoder and bi-encoder had been used for solving product matching as a zero-shot learning problem. However, the computational complexity of the cross-encoder makes it hard to scale millions of items. For the bi-encoder models, different loss functions (Reimers and Gurevych, 2019; Tracz et al., 2020) had been applied to the task but the approaches are fundamentally similar.

In addition, there are several works in different domains that inspired our paper. Knowledge distillation (Hofstätter et al., 2020) is proposed to let a teacher model instruct a student model through learning. While their focus is on training, we applied similar ideas for data annotation. The bi-encoder approach for retrieval was also used in the open-domain QA task (Karpukhin et al., 2020; Yamada et al., 2021). Though we do not need two separate encoders for the question and answer separately, the retrieval task is still similar in implementation. There are also works (Luan et al., 2021) focused on analysis representation for text retrieval. Our cross-encoder and bi-encoder models also use the difference in the representation created by different encoder for the product matching task.

7 Conclusion

We demonstrated that we can use a cross-encoder to provide data annotation and to improve product matching performance on a bi-encoder. While such an approach can be useful when human annotation for new data is not available, it can also improve the quality of human-annotated data by conducting intersection. Our empirical analysis suggests that the intersection of our cross-encoder annotation and human annotation creates more focused training data that improves the quality of the product embedding space. As a result of this annotation technique, we obtained a new way to improve human annotation quality or building bi-encoder model without human annotation for product matching.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages

4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. 2006. Text mining for product attribute extraction. *ACM SIGKDD Explorations Newsletter*, 8(1):41–48.

Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving efficient neural ranking models with cross-architecture knowledge distillation. *arXiv preprint arXiv:2010.02666*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *arXiv preprint arXiv:2004.00584*.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345.

Karin Mauge, Khash Rohanimanesh, and Jean-David Ruvini. 2012. Structuring E-commerce inventory. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 805–814, Jeju Island, Korea. Association for Computational Linguistics.

Ralph Peeters, Christian Bizer, and Goran Glavaš. 2020. Intermediate training of bert for product matching. *small*, 745(722):2–112.

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 and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Kashif Shah, Selcuk Kopru, and Jean-David Ruvini. 2018. Neural network based extreme classification and similarity models for product matching. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 8–15, New Orleans - Louisiana. Association for Computational Linguistics.

Janusz Tracz, Piotr Iwo Wójcik, Kalina Jasinska-Kobus, Riccardo Belluzzo, Robert Mroczkowski, and Ireneusz Gawlik. 2020. BERT-based similarity learning for product matching. In *Proceedings of Workshop on Natural Language Processing in E-Commerce*, pages 66–75, Barcelona, Spain. Association for Computational Linguistics.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*.

Ikuya Yamada, Akari Asai, and Hannaneh Hajishirzi. 2021. Efficient passage retrieval with hashing for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 979–986, Online. Association for Computational Linguistics.