Amazon Query Product Ranking

Li Yuan*
Jay Kim*
Daron Greenblatt*
li.yuan@vanderbilt.edu
jay.kim@vanderbilt.edu
daron.e.greenblatt@vanderbilt.edu
Data Science Institute at Vanderbilt University
Nashville, Tennessee, USA

Figure 1. Amazon KDD Cup, 2022.

Abstract

Improving the relevance of search results can significantly enhance the customer experience and their engagement with search. Amazon and other e-commerce want to design a new recommendation system which can give more accurate and relevant search results given the query. In this paper, the task is that given one query the system needs to re-rank a list of search products matched with this query to maximize the relevance. We proposed three types of models to tackle this task. The first one is Multi-Genre Natural Language Inference. The second is Cross-Encoders Regression. The third is Cross-Encoder Classification. Finally, we evaluate these three models in the test set. All our codes can be found at GitHub: https://github.com/vanderbilt-data-science/sna. Official website of the Amazon KDD Cup 2022 competition: https://www.aicrowd.com/challenges/esci-challenge-for-improving-product-search.

Keywords: Recommendation System, Rank products, Transformers, Multi-Genre Natural Language Inference, Cross-Encoders Regression, Cross-Encoder Classification

ACM Reference Format:
Li Yuan, Jay Kim, and Daron Greenblatt. 2022. Amazon Query Product Ranking. In *Proceedings of Social Network Analysis at Data Science Institute (Social Network Analysis).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

*Both authors contributed equally to this research.

Social Network Analysis, March 29–May 8, 2022, Nashville, TN

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Nearly everyone in our Social Network Class has utilized a recommender system one way or another, whether it be browsing through recommended items on a online clothing retail store, viewing advertisements that pop up on the side of our screen, or even when grocery shopping online. Our group wanted to explore what goes behind building these

systems and how a computer identifies between relevant and irrelevant items. Thus, our project was based on Amazon KDD Cup 2022 where the overall goal of the competition was to ultimately improve the relevance of search results. This is an important task, as such improvements can significantly enhance the customer experience and increase your engagement with the Amazon online platform. While there have been incredible advances in the field of machine learning and data science as a whole, accurately classifying items in a recommender system remains a challenging task. Some of these difficulties range from noisy datasets, ambiguous query searches, and the potential lack of diversity in the items available. Further, the goals of recommender systems is not only to accurately return relevant products, but also to rank them in order of relevance. The competition consisted of three parts, but we only worked on the first, the 'Query Product Ranking', where the primary objective was to be able to rank relevant products over nonrelevant ones given a real user query, a typical information retrieval task according to the world of e-Commerce. The input would be a list of queries, and the algorithm outputs a csv file and identifies a list of queries that would be ranked in relevance. Some of the immediate challenges we faced were in the initial model selection and the heavy computation resources that these models may require. We quickly found out that our initial attempt of implementing a graph neural network-based model was not feasible and decided it was not the best method as suggested by other people. We decided to first create BERT word embedding for the products and queries to be able to represent text as numerical representations and used Multi-Genre Natural Language Inference, Cross-Encoder Classification, and Cross-Encoder Regression as our models. It is important to note that a high accuracy rate is vital in user-experience driven data projects, as a few irrelevant recommendations can greatly negatively affect user-experience.

2 Data Description

For this project, we were given three datasets to work with, two train and one test. For the train datasets, we had a table of 'products' and a table of 'queries'. The product table was roughly 800,000 rows and consisted of all the products' ID number as well as its corresponding product title, description, bullet point, brand, color, and locale. The queries table was nearly 700,000 rows and contained all the queries' IDs, the query itself, query locale, the product ID of the product that was outputted, along with a tag indicating its level of relevance. The tags were one of the following: Relevant, Complement, Substitute and Irrelevant. The test data set was much smaller than our train, just under 170,000 rows, and was not a labeled data set. The table consisted only of the query ID, query, locale, and product ID. The competition goal is to train a model and re-rank the product list given the each query in the test set. Once finish, submit the re-rank the test result to Amazon cup. One example of query set is below table 1. An example of a set of products is in the table 2. Two examples of the test set are given below table 3. Our goal is to place the most relevant product produced by our model at the first position matched with the query, while the least relevant product at the bottom position.

3 Data Pre-processing

The product data set contains all product attributes such as product id, title, description, location, bullet points, and so on. The first step we did was to concatenate all product attributes (string types) into one longer column (string) excluding the location attribute, as us, jp because we thought that the two-letter location would be a noise to the transformers or other model.

The second step is to join product data set with query data set on the product id so that we have the pairwise query string, product string, and label column (r, s, c, i) in one table and build models on it.

4 Modeling

After discussing that simple linear neural network is not capability of modeling this task and a graph neural network is also not a good solution to the recommendation system because we didn't find explicit links between every query and product and coming up with new links and building graphs need very strong and reasonable assumptions which need further experiments and tests. Due to the time limit, we decided to try other models, which heavily rely on the natural language processing techniques because the main data types in the joined data set are string and there are tons of state-of-the-art off-the-shelf NLP techniques which are easy to train and employ. We did a lot of search online and found that three types of NLP models may be suitable to resolve this rerank task. After modeling, fortunately, the Cross-Encoder model we selected was also chosen to be the baseline model by the Amazon competition officials.

4.1 Multi-Genre Natural Language Inference

We were inspired by the work of three professors, Adina Williams, Nikita Nangia and Samuel R. Bowman at New York University. They published a paper in 2018 that introduces the Multi-Genre Natural Language Inference (MultiNLI) corpus, a data set designed to test and evaluate new machine learning methods of natural language processing [Williams et al. 2017]. In this paper, the main task in natural language inference is to recognize textual entailment. The table below 4 illustrates this task by giving three typical examples. The model is given the pair of sentences. One is the premise, which is the true fact; the other is the hypothesis, which is the description of this premise. The meaning relationship between the premise and hypothesis has three kinds of labels. The first one is a contradiction, which shows that the

query_id	query	query_locale	product_id	esci_label
0	!awnmower tires without rims	us	B00004RA3F	Exact

Table 1. One Example of query data

product_id	product_title	product_description	product_bullet_point	product_brand	product_color_name
B0188A3QRM	Amazon Basics Wood-cased #2 Pencils, Un-sharpene...	NaN	144 woodcase #2 HB pencils made from high-qual...	Amazon Basics	Yellow

Table 2. One Example of product data

query_id	query	query_locale	product_id
33777	!qscreen fence without holes	us	B000HMCA7W
33777	!qscreen fence without holes	us	B0019Y6TTC

Table 3. Two Examples of test data

hypothesis is incorrect in its premise. The second one is entailment which shows that the hypothesis is correct about the premise. The third one is neural, which shows that the premise cannot tell true or false about the hypothesis.

premise	hypothesis	label
I like apples.	I don't like apples.	contradiction
I met my girl-friends.	At 8am, my girl-friends came to see me.	entailment
It's raining.	I'm studying.	neutral

Table 4. Examples of recognizing textual entailment.

Inspired by their work and paper, we want to extend and apply it to our Amazon re-ranking task. In our task, we have four categories: exact, substitute, complement, irrelevant. Our query should be the premise and our concatenated product description should be its matched hypothesis, and we extend the three categories into four. The below 5 illustrates examples in our re-ranking task.

Next, we would like to fine tune a DistilBERT base model [Devlin et al. 2018] because this model is uncased: it does not make a difference between apple and Apple. DistilBERT base is a general-purpose pre-trained model with 40% smaller, 60% faster and maintains the 97% ability of natural language inference on the same corpus in a self-supervised way by using the BERT base model as a teacher. The two main advantages of this Distil-BERT model is it is pre-trained to be more general purpose instead of task-specific like some prior BERT models; the second is that this model as a student model is to learn and mimic the larger teacher model's behavior as close as possible. In such a way, it is easier for us to fine-tune and transfer this model for other tasks, like text classification, we did, and the model is lighter, so the fine-tuning process is faster.

query	product	label
!awnmower tires without rims	American Lawn Mower Company 1204-14 14-Inch 4-...	Irrelevance
landmowers	American Lawn Mower Company 1204-14 14-Inch 4-...	Exact
reel mower with grass catcher	American Lawn Mower Company 1204-14 14-Inch 4-...	Substitute
classroom friendly supplies pencil sharpener	Wood-Cased #2 HB Pencils, Yellow, Pre-sharpene...	Complement

Table 5. Examples of Amazon query product ranking.

Then, we stratified the split of the merged data into **571,223** row training, **142,806** row validation, and **79,337** row test set. To feed our string data to the transformers model, we need to tokenism them by using Transformers Tokenizer, including converting the tokens to their corresponding IDs in the pre-trained vocabulary. We also define validation accuracy as the training metric. We fully fine-tuned the training data set, and it took us 6 hours to run on the colab GPU. To facilitate this training process, we purchased the colab pro+ subscriptions to obtain a more powerful GPU and virtual memory for handling this huge data set and speeding

up the training. Finally, the GPU that we used to train was **Tesla V100-SXM2-16GB** and the size of the virtual machine that we used was **54.8 GB**. Because training one epoch took 3 hours to finish, we found that training for more than 3 epochs did not significantly improve validation accuracy. As a result, we set up two epochs to train in two epochs.

The following hyper-parameters were used during training:

- learning_rate: 2×10^{-5}
- train_batch_size: 16
- eval_batch_size: 16
- seed: 42
- optimizer: Adam with $\beta = (0.9, 0.999)$ and $\epsilon = 10^{-8}$
- lr_scheduler_type: linear
- num_epochs: 2

4.2 Training Results

The below figure 2 is what the training eventually displayed. We obtained 66% validation accuracy after training. This was not bad when we had four categories to classify, the probability of random guessing was only 25%.

[71404/71404 5:07:34, Epoch 2/2]

Epoch	Training Loss	Validation Loss	Accuracy
1	0.898100	0.866245	0.637050
2	0.783700	0.824447	0.661700

Figure 2. Training results on the full training set

In addition to evaluating our model on the validation set, we would also like to evaluate it on the held-out test. On the **79,337** rows of the test, the model had **65%** accuracy and **0.65** F1 score. This test result was very similar to the validation accuracy, which showed that our model had a good generalization result and a good accuracy on this text classification task.

The below figure 3 visualized the details of the classification results of each category. As we can observe, this model had a strong ability to predict the exact class and did not perform well on the complement class.

The reason why we got this was that the label distributions of training set was unbalance as shown below table 6. The training set and original data-set contain much more exact and substitute classes than others.

Label	Percentage
complement	5%
exact	43%
irrelevant	16%
substitute	34%

Table 6. Percentages of each class in training set

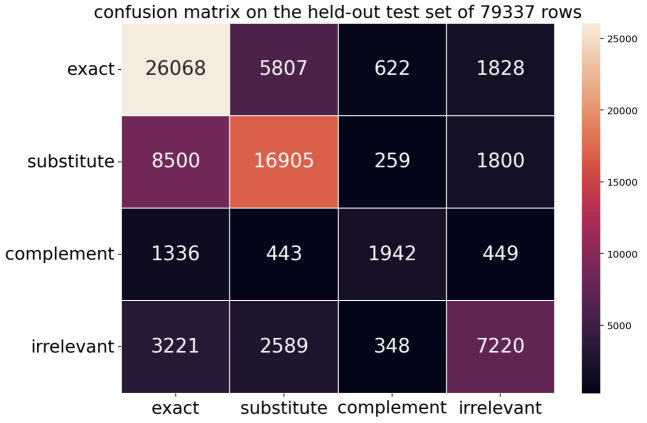

Figure 3. Confusion Matrix on the test set

4.3 Codes and Hugging Face Model Card

All the codes and implementation can be viewed at GitHub: https://github.com/vanderbilt-data-science/sna/blob/main/06-fine-tune-our-dataset.ipynb.

After 6 hours of complete training, we also published our model card for the hugging face: https://huggingface.co/LiYuan/amazon-query-product-ranking.

You can go to this website to play around the model by typing different queries and product descriptions into the right hosted inference API. At the time of writing this paper, there are 23 community members who downloaded our model to test, and as time goes by, more downloads are available.

You can download our fine-tuned model by copying and pasting the following codes:

```
from transformers import AutoTokenizer,
    AutoModelForSequenceClassification

tokenizer = AutoTokenizer.from_pretrained("LiYuan/
    amazon-query-product-ranking")

model = AutoModelForSequenceClassification.
    from_pretrained("LiYuan/amazon-query-product-
    ranking")
```

4.4 Cross-Encoders Regression

This model is actually very accurate for this task, intuitively inspired by information retrieval techniques. In 2019, Nils Reimers and Iryna Gurevych introduced a new transformers model called Sentence-BERT, Sentence Embeddings using Siamese BERT-Networks. [Reimers and Gurevych 2019]

This new Sentence-BERT model is modified on the BERT model by adding a pooling operation to the output of BERT model. In such a way, it can output a fixed size of the sentence embedding to calculate cosine similarity, and so on. To obtain a meaningful sentence embedding in a sentence vector space where similar or pairwise sentence embedding are close, they

created a triplet network to modify the BERT model as the architecture below figure 4

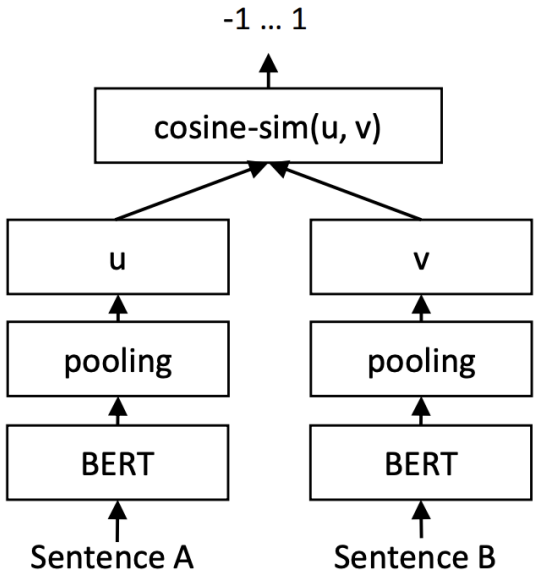

Figure 4. SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

As we can observe from above figure 4, a pooling layer is added on the top of each BERT Model to obtain the sentence embedding u and v. Finally, the cosine similarity between u and v can be computed to compare with the true score or make them semantically meaningful, then the mean square error loss, which is the objective function, can be back-propagated through this BERT network model to update the weights.

In our amazon case, the query is sentence A and concatenated product attributes are sentence B. We also stratified split the merged set into **571,223** rows for training, **500** rows for validation, **3,000** rows for test. We limited the output score between 0 and 1. The following scores represent the degree of relevance between the query and the product attributes in light of Amazon KDD Cup website; however, this can be adjusted to improve the model performance.

- **1**: exact
- **0.1**: substitute
- **0.01**: complement
- **0**: irrelevance

4.5 Training Results

After **8** hours of training on the colab GPU, figure 5 shows the training results.

For this regression model, we used **Pearson correlation coefficient** and **Spearman's rank correlation coefficient**

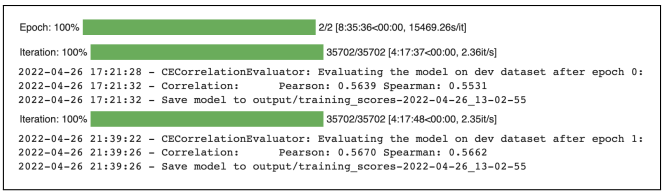

Figure 5. Cross-Encoder Regression Training Results for 2 epochs

to measure the model performance. If the correlation coefficient is high, the model performs well. The validation Pearson is **0.5670** and validation Spearman is **0.5662**. This is not bad result.

We also evaluated the model on the test set. We got **0.5321** for Pearson and **0.5276** for Spearman. These results from the test evaluation have results similar to those of the validation set, suggesting that the model has a good generalization.

Finally, once we have this fine-tuned Cross-Encoder Regression model, given a new query and its matched product list, we can feed them into this model to get the output score to rerank them so that this can improve the customer online shopping experience.

4.6 Codes and Hugging Face Model Card

All the codes and implementation can be viewed at GitHub: https://github.com/vanderbilt-data-science/sna/blob/main/05-Train-CrossEncoder-scores.ipynb.

After 8 hours of full training, we also published our Model Card to the Hugging Face: https://huggingface.co/LiYuan/Amazon-Cup-Cross-Encoder-Regression.

You can go to this website to play around with the model by typing different queries and product descriptions into the right hosted inference API.

You can download our fine-tuned model by copying and pasting the following codes:

```
from transformers import AutoTokenizer,
    AutoModelForSequenceClassification

tokenizer = AutoTokenizer.from_pretrained("LiYuan/
    Amazon-Cup-Cross-Encoder-Regression")

model = AutoModelForSequenceClassification.
    from_pretrained("LiYuan/Amazon-Cup-Cross-
    Encoder-Regression")
```

4.7 Cross-Encoder Classification

There are two types of Cross-Encoder models. One is the Cross-Encoder Regression model that we fine-tuned and mentioned in the previous section. Next, we have the Cross-Encoder Classification model. These two models are introduced in the same paper [Reimers and Gurevych 2019]. Both models resolve the issue that the BERT model is too time-consuming and resource-consuming to train in pairwised

sentences. These two model weights are initialized as the BERT and RoBERTa networks. We only need to fine-tune them, spending much less time to yield a comparable or even better sentence embedding. The below figure 6 shows the architecture of Cross-Encoder Classification.

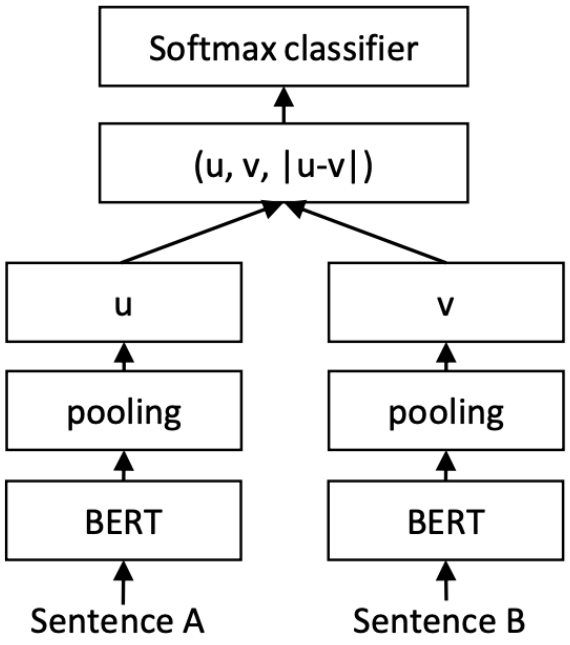

Figure 6. SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

We made the elementwise difference between two embeddings u and v, then the classification objective function is

$$o = \text{softmax}\left(W_t(u, v, |u - v|)\right)$$

.

In our case, we had four labels to train: exact, substitute, complement, and irrelevance. We chose the bert-base-uncased [Devlin et al. 2018] as our initialization BERT model and weights. We stratified the merged data into 200,000 rows of training set, 2,000 rows of validation set, and 2,000 rows of test set. We tried to fully fine-tune on the 571,223 rows of training set, but the training process was too slow with one epoch taking 12 hours and low improvement. Therefore, we considerably reduced the training set to 200,000 rows and obtained similar and comparable results.

4.8 Training Results

The below figure 7 was our training results for the first epoch. The best validation accuracy we got from this training process is **46%**.

Then we evaluated the model performance on the 2,000 held-out test set. We also got a test accuracy **46.05%** that is

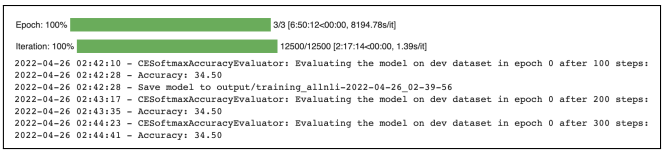

Figure 7. Training Results for Cross-Encoder Classification

almost identical to the best validation accuracy, suggesting a good generalization model. The below figure 8 visualized the confusion matrix we got.

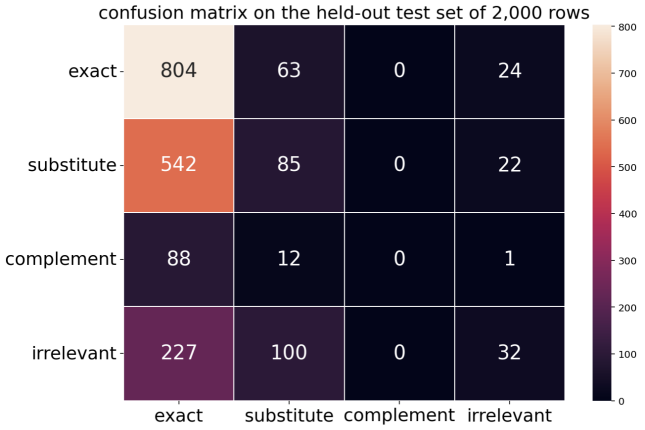

Figure 8. Confusion Matrix for Cross-Encoder Classification Test

However, we found this Cross-Encoder model didn't perform well on this four category task, because it classified model instances into exact class. This is because of overwhelming exact class instances in our original and training data-set. If we would like to get an improved the model classification, the balanced the training data-set is needed. So, this Cross-Encoder classification model didn't give us a satisfactory result. Also detailed in their paper [Reimers and Gurevych 2019], usually the Cross-Encoder Regression model performs better than the Cross-Encoder Classification model. This is why we would get poor performance on this classification task.

4.9 Codes and Hugging Face Model Card

All the codes and implementation can be viewed at GitHub: https://github.com/vanderbilt-data-science/sna/blob/main/04-Train-CrossEncoder-classification.ipynb.

After 8 hours of full training, we also published our Model Card to the Hugging Face: https://huggingface.co/LiYuan/Amazon-Cross-Encoder-Classification.

You can go to this website to play around the model by typing different queries and product descriptions in the right hosted inference API.

You can download our fine-tuned model by copying and pasting the following codes:

```
from transformers import AutoTokenizer,
    AutoModelForSequenceClassification

tokenizer = AutoTokenizer.from_pretrained("LiYuan/
    Amazon-Cross-Encoder-Classification")

model = AutoModelForSequenceClassification.
    from_pretrained("LiYuan/Amazon-Cross-Encoder-
    Classification")
```

5 Challenges

Initially, we would like to select one of the graph neural networks to model it, but we found that graph modeling did not fit to this reranking task. It's reported by others who did try graph neural network on this re-ranking task that performance isn't good and even worse than some tradition recommendation system, like Alternating Least Square (ALS) Matrix Factorization in Collaborative Filtering [Bell et al. 2009].

Fine-tuning the full training set took a lot of time and GPU resources. We even purchased the co-lab **Pro+** subscription to increase our virtual memory quota, obtain more powerful GPU and longer back-end running time. Only if we have these enough hardware resources, fine-tuning on fully training data set can be done. Then we can reach a more consolidating conclusion and compete for this Amazon KDD cup.

Compared with their held-out test, we found that Multi-Genre Natural Language Inference performs better than both Cross-Encoder models. However, due to the difference measurement of model performance, Pearson and Spearman score used for regression model, while accuracy and F1 score used for classification model, it is hard to directly compare these two Cross-Encoder models' performance.

As suggested by some other researchers, the Cross-Encoder regression model may be more suitable to our re-ranking task because there is some implicit order of four categories. **exact** > **substitute** > **complement** > **irrelevant**. However, this order relationship is not strong enough to discard the Cross-Encoder classification model. It was better to experiment with both of them and compare them.

6 Future Work

1. In our previous work due to the time limit, we only tried the default hyperparamaters, however, there are some ways and packages combined with transformers which can operate **Grid Search** on hyperparameter for fine-tune models to get the best combinations
2. Actually, the Cross-Encoder class is a wrapper around Hugging Face `AutoModelForSequenceClassification`, most Hugging Face pre-trained models which are compatible with `AutoModel` may be used in the Cross-Encoder models. Thus, We can compare more different transformer pre-trained models.
3. For Cross-Encoder Regression model, we previously only tried the assigned scores of four categories provided by Amazon, but we think we can adjust the assigned scores try to improve the test performance
4. Try to combine Cross-Encoder Regression and Classification models and improve the results (this is very creative and require a lot of time to think)
5. So far, we only plainly concatenated all product attributes together. However, trying different ways of concatenating product attributions and manipulating it a bit may improve the test performance.
6. Look for new ranking methods which are specialized in the context of e-commerce. So far, our methods or models are general to any information retrieval. There may be some methods that are specialized designed for e-Commerce and have better results.

7 Rights Information

Authors of any work published by ACM will need to complete a rights form. Depending on the kind of work, and the rights management choice made by the author, this may be copyright transfer, permission, license, or an OA (open access) agreement.

The rights information is unique to the work; if you are preparing several works for an event, make sure to use the correct set of commands with each of the works.

The ACM Reference Format text is required for all articles over one page in length, and is optional for one-page articles (abstracts).

Acknowledgments

To Tyler Derr for giving the amazing social network analysis course in Spring 2022

The thanks are given to Jesse Spencer-Smith for giving the impressive Transformers introductory course in Spring 2022.

References

R. Bell, Y. Koren, and C. Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 08 (aug 2009), 30–37. https://doi.org/10.1109/MC.2009.263

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://doi.org/10.48550/ARXIV.1810.04805

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. https://doi.org/10.48550/ARXIV.1908.10084

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2017. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. https://doi.org/10.48550/ARXIV.1704.05426