Domain adaptation of transformers model for products search

Algis Dumbris

December 2020

Abstract

The project report contains a description of experiments for domain adaptation of transformers model for products search. Project code: `https://github.com/Dumbris/semantic-search-domain-adaptation`, Dataset `https://www.kaggle.com/c/home-depot-product-search-relevance/data`.

1 Introduction

Using transformer models (BERT, RoBERTa, GPT) over the last two years demonstrates significant quality improvements on many NLP tasks. For many companies selling products on the Internet, the search function is a core function. For such companies, search relevancy is a measure of how quickly they can get customers to the right products.

Traditionally for product search uses term-matching algorithms (BM25) builtin into ElasticSearch or Solr. But these systems may fail when queries and product descriptions use different terms to describe the same meaning. Semantic search based on transformer models could help in this case. In my research, I am trying to find the best methods for the domain adaptation of transformer models for improving search relevancy. I intentionally avoid any data preprocessing here and try to use text data as is for transformers models.

1.1 Team

The author of this project is **Algis Dumbris**.

2 Related Work

The paper [Reimers and Gurevych, 2019] describes how Sentence Embeddings implemented and how it can be used in search and ranking tasks.

[Padigela et al., 2019] - Comparation BERT and BM25 algorithm performance on MS MARCO dataset.

3 Dataset Description

I am using the Home Depot Product Search Relevance dataset from Kaggle. The dataset contains real customer search terms from Home Depot's website. In total dataset contains 186135 pairs of query and product with relevancy label. Home Depot has crowdsourced the search/product pairs to multiple human raters to create the ground truth labels. The relevance label is a number between 1 (not relevant) to 3 (highly relevant). For example, a search for "AA battery" would be considered highly relevant to a pack of size AA batteries (relevance = 3), mildly relevant to a cordless drill battery (relevance = 2), and not relevant to a snow shovel (relevance = 1).

Type	Train	Test
Queries	19384	4845
Docs	149143	36992
Added With Negative Samples	298270	

Table 1: Statistics of the Train / Test split

Table 2: Samples of query / docs / relevancy data

query	product_title	relevance
	GE Z-Wave 1800-Watt Resistive CFL-LED Indoor P...	3.0
	Leviton Z-Wave Controls 3-Way/Remote Scene Cap...	3.0
	Leviton Decora Z-Wave Controls 15 Amp Scene Ca...	3.0
zwave switch	GE Home Automation 120 VAC 3-Way Auxiliary Add...	3.0
	GE Z-Wave 600 Watt CFL-LED Indoor In-Wall Dimm...	2.7
	Z-Wave Wireless Lighting Control with Keypad C...	2.7
	Lutron Aurora Wireless Lighting Control System...	2.3
	Leviton Z-Wave Enabled 15 Amp Scene Capable Re...	2.0
	Milwaukee Metal Hole Saw Kit (15-Piece)	3.0
	Milwaukee 4 in. Bi-Metal Hole Saw	3.0
	Milwaukee 6-3/8 in. Recessed Light Hole Saw	3.0
$ hole saw	PRO-SERIES 19-Piece Hole Saw Set with Case	2.3
	Ryobi Hole Saw Set (6-Piece)	2.3
	Bosch Daredevil Spade Bit Set (10-Pieces)	2.0
	Milwaukee 3/8 in. Ergo Quick-Change Saw Arbor	2.0
	Milwaukee 2 in. to 7 in. Dia. Adjustable Hole ...	1.3

In my experiments, for matching, I am using only the product name text attribute. The product description text attribute potentially useful for pretraining using a masked language modeling task (MLM). It's left for future experiments.

4 Experiment Description

I wrote a python library that allows to change experiment parameters, manipulate datasets, and calculate metrics for conducting experiments. The general scheme of experiments is presented on Fig. 1. In the beginning, the whole dataset split on the train and test parts. While the model is trained, on some steps, the evaluator starts metrics calculation on test data. On the test step, the evaluator rebuilds the docs candidates list for each query using the trained model. These candidates list compared with the real docs list for metrics calculation.

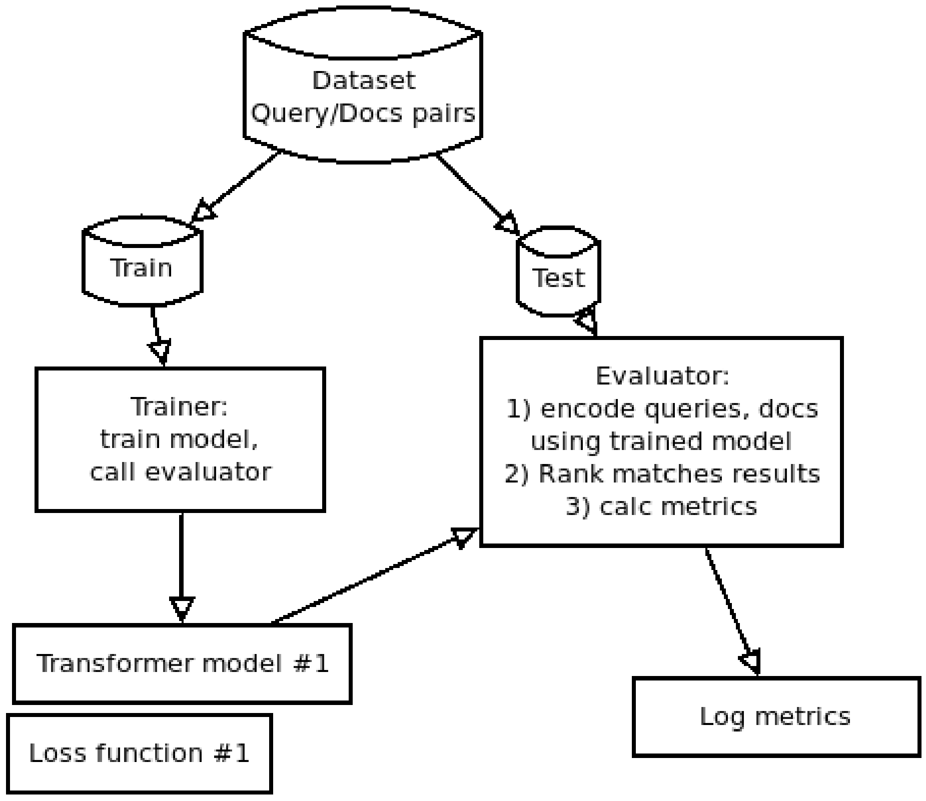

Figure 1: Experiment design.

In current setup I am using losses: *CosineSimilarityLoss* - The similarity of these embeddings is computed using cosine similarity and the result is compared to the gold similarity score. *SoftmaxLoss* - adds a softmax classifier on top of the output of two transformer networks. I am using three pretrained models from package sentence_transformers msmarco-distilroberta-base-v2, distilbert-base-nli-stsb-quora-ranking, LaBSE.

4.1 Metrics

I am using Normalized Discounted Cumulative Gain (nDCG@k) @3, @10. And Mean average precision (MAP@k) @3, @10

4.2 Results

Results for best-trained models are presented in Table. 3 for comparison, I have added BM25 and Universal Sentence Encoder models. On nDCG@10, metrics transformers give results slightly worse than the BM25 algorithm. On Map@k metrics, transformers showed poor performance - BM25 Map@3 is 0.27, but the best transformer model gives only 0.012.

Table 3: nDCG metrics on the test dataset, after training.

model	loss	**nDCG@10**	nDCG@3
BM25		0.3908	0.3450
sentence-transformers/LaBSE	**CosineSimilarityLoss**	**0.3708**	0.3408
distilbert-base-nli-stsb-quora-ranking	CosineSimilarityLoss	0.3564	0.3238
sentence-transformers/LaBSE	SoftmaxLoss	0.3478	0.3136
distilroberta-base-msmarco-v2	CosineSimilarityLoss	0.2934	0.2608
distilbert-base-nli-stsb-quora-ranking	SoftmaxLoss	0.2846	0.2487
distilroberta-base-msmarco-v2	SoftmaxLoss	0.2635	0.2301
universal-sentence-encoder/4		0.2326	0.1867

Table 4: MAP metrics on the test dataset, after training.

model	loss	**MAP@10**	MAP@3
BM25		0.2375	0.2754
universal-sentence-encoder/4		0.0091	0.0138
sentence-transformers/LaBSE	**SoftmaxLoss**	**0.0080**	0.0124
distilbert-base-nli-stsb-quora-ranking	SoftmaxLoss	0.0080	0.0120
distilroberta-base-msmarco-v2	CosineSimilarityLoss	0.0078	0.0119
sentence-transformers/LaBSE	CosineSimilarityLoss	0.0077	0.0112
distilbert-base-nli-stsb-quora-ranking	CosineSimilarityLoss	0.0076	0.0112
distilroberta-base-msmarco-v2	SoftmaxLoss	0.0076	0.0110

To understand how metrics on test data are changing during training, you can check Fig. 2. One epoch on these graphs contains 7000 steps. You can see that after the first epoch, metrics almost don't change.

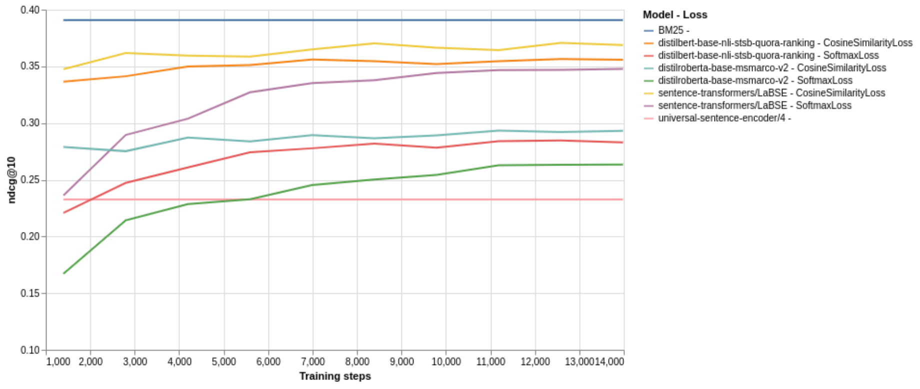

Figure 2: nDCG@10 metrics.

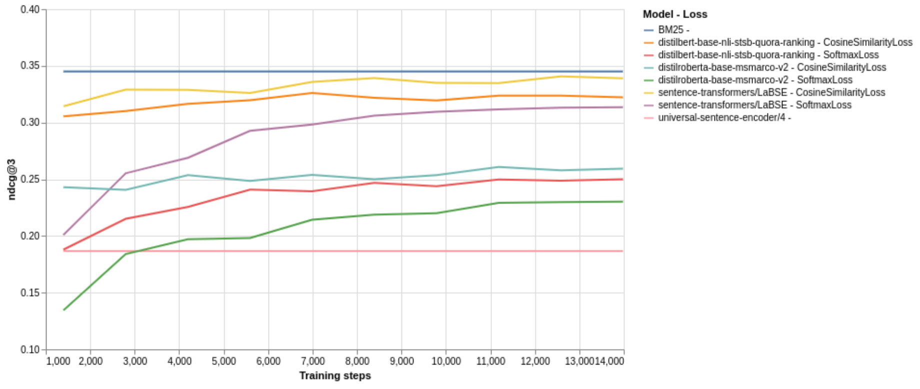

Figure 3: nDCG@3 metrics.

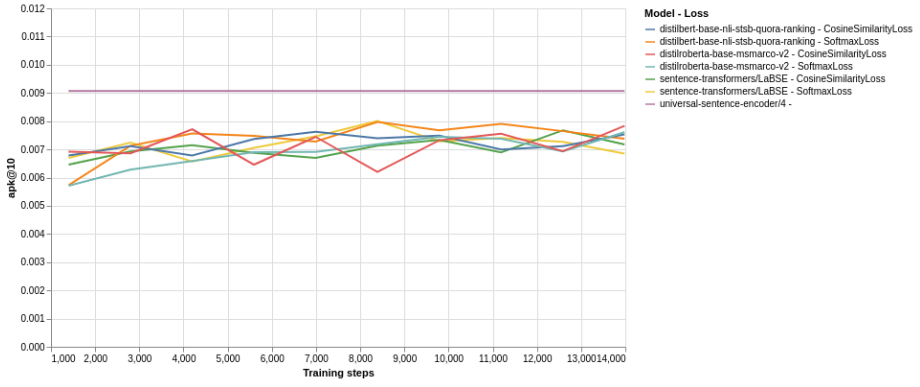

Figure 4: MAP@10 metrics.

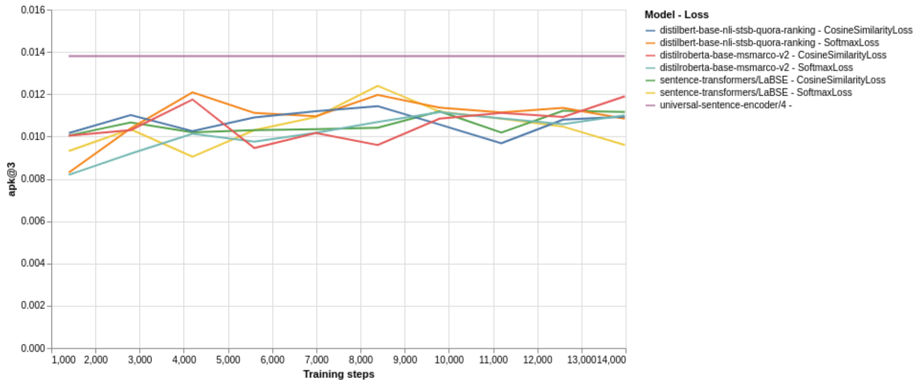

Figure 5: MAP@3 metrics.

5 Conclusion

I tried to apply BERT and RoBERTa models for product search using the product title in this project. On the current dataset, using just fine-tuning on annotated data, transformers models give results worse than the BM25 algorithm. I think it is related to the nature of data. Product names contain plenty of abbreviations, numbers, etc. Also, query and product titles are usually short and do not allow to leverage context information. Between distilled models: better performance showed by models were first trained on NLI data, then fine-tuned on the STS benchmark dataset, and finally fine-tuned on the annotated domain-specific dataset using cosine similarity loss.

References

[Padigela et al., 2019] Padigela, H., Zamani, H., and Croft, W. B. (2019). Investigating the successes and failures of bert for passage re-ranking.

[Reimers and Gurevych, 2019] Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.