BERT-based similarity learning for product matching

Janusz Tracz[1], **Piotr Wójcik**[1], **Kalina Jasinska-Kobus**[1, 2],
Riccardo Belluzzo[1], **Robert Mroczkowski**[1], **Ireneusz Gawlik**[1, 3]
[1] ML Research at Allegro.pl
[2] Poznan University of Technology
[3] AGH University of Science and Technology
{janusz.tracz,piotr.wojcik,kalina.kobus,riccardo.belluzzo,
robert.mroczkowski,ireneusz.gawlik}@allegro.pl

Abstract

Product matching, i.e., being able to infer the product being sold for a merchant-created offer, is crucial for any e-commerce marketplace, enabling product-based navigation, price comparisons, product reviews, etc. This problem proves a challenging task, mostly due to the extent of product catalog, data heterogeneity, missing product representants, and varying levels of data quality. Moreover, new products are being introduced every day, making it difficult to cast the problem as a classification task.

In this work, we apply BERT-based models in a similarity learning setup to solve the product matching problem. We provide a thorough ablation study, showing the impact of architecture and training objective choices. Application of transformer-based architectures and proper sampling techniques significantly boosts performance for a range of e-commerce domains, allowing for production deployment.

1 Introduction

With more and more retailers moving their business online the number of items available on e-commerce marketplaces, such as Amazon, Alibaba or Allegro.pl, grows exponentially. In an environment with hundreds of millions of items listed for sale every day, providing a satisfactory search and purchase experience brings many challenges.

One such huge challenge for e-commerce portals is introducing product-based experience, both for the buyers and for the merchants. From the buyer's perspective, this means facilitating the search process by grouping offers that refer to the same real-world product while being sold by different merchants. Merchants, on the other hand, benefit by having access to a high-quality product catalog, which allows them to speed up the listing process and provide the buyers with more complete product descriptions. Achieving product-based experience in any e-commerce portal is only made possible by being able to automatically find offers of the same product. This process is often called product matching. Product matching in e-commerce is a non-trivial task mostly because of the large number of products, their high heterogeneity, missing product representants, and varying levels of data quality. For a more detailed overview of challenges that make product matching hard, we refer the reader to (Shah et al., 2018).

Classical approaches to product matching and its generalization (entity matching) often rely on rule-based methods and hand-crafted features such as string similarity measures (Thor, 2010; Köpcke et al., 2010; Konda et al., 2016; Brunner and Stockinger, 2019). Recent advances of deep learning in natural language processing (NLP) sparked increasing interest in end-to-end deep learning approaches both for entity matching (Brunner and Stockinger, 2020; Li et al., 2020b) and product matching (Shah et al., 2018; Ristoski et al., 2018; Li et al., 2020a). Here, we follow this trend and present our experiences with leveraging transformer-based neural language models (Vaswani et al., 2017) combined with the similarity learning setup for product matching. While transformer architectures have been very recently applied to entity matching (Brunner and Stockinger, 2020), to the best of our knowledge, we are the first

Proceedings of the Workshop on Natural Language Processing in E-Commerce (EComNLP), pages 66–75
Barcelona, Spain (Online), Dec 12, 2020.

to apply them specifically for the product matching problem in e-commerce. Aside from architectural choices, we show that sampling techniques tailored to the product matching domain prove to be crucial in the progress of similarity learning.

The contributions of our paper are as follows:

- we apply state-of-the-art BERT-based models (Devlin et al., 2019) in the similarity learning setup to solve the product matching task in the e-commerce domain,
- we compare the usefulness of modern BERT-based architectures such as BERT and DistilBERT (Sanh et al., 2019) for the product matching task,
- we propose *category hard* batch construction strategy, which proves to increase the fraction of active training triplets and the performance of the final model,
- we adopt and evaluate different batch construction strategies in the similarity learning setup for solving product matching.

This work is organized as follows. Section 2 introduces the product matching problem. Section 3 describes our approach. In Section 4 we present the experiments and results. Section 5 briefly reviews the related work on product matching. Finally, we discuss the challenges and conclude our work in Section 6.

2 Product matching

Product matching aims at identifying offers of the same product across many merchants selling it in an e-commerce portal and integrating the information into a single entry in a product catalog. An offer is an instance of a specific good described by vendor-provided information. This information may include but is not limited to, title, its text description, attributes, category, and photos. A product represents a manufacturer's description of a good and is described similarly. At a typical e-commerce marketplace, one may find many offers of the same product. Also, not all products need to have offer representatives (e.g. legacy or yet unmatched products).

Recent papers mostly focus on using only the information contained in the titles or using both titles and attributes (Li et al., 2020a). In this work, in addition to using the title and attributes information, we also make use of the category, i.e., an identifier of a set of goods of the same type.

2.1 Generalized zero-shot multi-class classification

Production use of a product matching system requires that the system is able to operate correctly for yet unrepresented products. As products are being introduced, the product matching system should be capable to handle new instances.

Motivated by this requirement, we formalize the product-matching problem as generalized zero-shot multi-class classification following the definition in (Li et al., 2019). Let $\mathcal{O}$ denote the set of offers, and $\mathcal{P} = \mathcal{P}_s \cup \mathcal{P}_u$ the set of products consisting of two non-overlapping sets of seen products $\mathcal{P}_s$ and unseen products $\mathcal{P}_u$. Suppose we have three sets of data $\mathcal{D} = \{\mathcal{D}_s, \mathcal{D}_u, \mathcal{D}_a\}$, where $\mathcal{D}_s$, $\mathcal{D}_u$ and $\mathcal{D}_a$ are training, test and semantic description sets, respectively. The training and test sets consist of matching offer-product pairs, i.e., $\mathcal{D}_s = \{(o, p) : o \in \mathcal{O}, p \in \mathcal{P}_s\}$ and $\mathcal{D}_u = \{(o, p) : o \in \mathcal{O}, p \in \mathcal{P}\}$. The aim is to learn transferable knowledge from $\mathcal{D}_s$ to be able to predict matching products for offers present in pairs from $\mathcal{D}_u$ additionally making use of the semantic descriptions of offers and products $\mathcal{D}_a = \{\mathcal{O}, \mathcal{P}\}$.

Importantly, as this is a generalized zero-shot classification problem, not all products are observed in available training data. To be able to classify offers to products not observed during training, not only offers but also product classes need some representation.

Since the above definition of product matching is general and representation-agnostic, here we also formalize the specific settings of the product matching problem we solve in this work. We define each offer $o \in \mathcal{O}$ and each product $p \in \mathcal{P}$ as triples (t, a, c), where t is the title, a is the set of attributes and $c \in C$ is the category. The set of attributes a contains triples $(a_{name}, a_{values}, a_{unit})$, where a_{name} is the

attribute name, a_{values} is a set of one or more values and a_{unit} is an optional unit. Titles of products or offers t, attribute names a_{name}, attribute values a_{values} and attribute units a_{unit} are strings possibly containing multiple words.

Given an offer $o = (t_o, a_o, c_o)$ the problem we tackle in this work is to find its matching product $p \in P$. In the classic product matching problem $P = \mathcal{P}$. Here, we consider a variant of this problem, where $P = \{p = (t_p, a_p, c_p) : p \in \mathcal{P}, c_o = c_p\}$, i.e., we only look for the matching product in a set of products that belong to the same category as the offer.

3 Product matching with similarity learning

In this work, we interpret the product matching problem as a similarity learning task. This approach allows us to cast product matching problem as a zero-shot learning problem, in result avoiding re-training the model on the introduction of new products or product categories.

3.1 Triplet loss objective

To solve the product matching problem with triplet loss (Hoffer and Ailon, 2015), we introduce a notion of similarity between offers and products, defined as proximity of their representations in some embedding space. Each training example is defined as a triplet (o, p^+, p^-), denoting an offer (anchor), a matching product (positive) and a non-matching product (negative). Given encoders $\mathcal{E}_\theta : \mathcal{O} \to \mathbf{R}^N$ and $\mathcal{E}_\phi : \mathcal{P} \to \mathbf{R}^N$, both transforming respective entities to vector representations in some embedding space (Section 3.3), and distance measure d (in our case the cosine distance), we adjust network parameters θ and ϕ to minimise the triplet loss objective:

$$\mathcal{L}(o, p^+, p^-) = max(0, m + d(\mathcal{E}_\theta(o), \mathcal{E}_\phi(p^+)) - d(\mathcal{E}_\theta(o), \mathcal{E}_\phi(p^-))),$$

where margin m is a hyperparameter. In this work we tie parameters of $\mathcal{E}_\theta$ and $\mathcal{E}_\phi$, employing a single instance of the encoder for both of them.

One shortcoming of the standard triplet loss objective is that majority of possible triplets prove trivial. To ensure that loss values do not vanish after the initial phase of training, a proper batch construction strategy is essential. Properly selected *active* (non-zero loss) triplets let one effectively evaluate and optimise the specified loss function (Section 3.4). The batch construction strategy will be an object of the ablation study.

3.2 Textual representation

The representation of each offer and product is a concatenated title, attributes values, and units, which is then lower-cased and pass into a fast byte-pair encoder (Sennrich et al., 2016). We also tested including descriptions and attributes names in the representation, but in all of our experiments, they deteriorate the model performance. We fit the tokenizer on our domain offer and product corpora with a vocabulary size of 30k tokens.

3.3 Encoder architectures

It is important in the similarity learning setting to properly choose the encoder architecture. Our main focus is the transformer (Vaswani et al., 2017) based encoders, which recently gained a lot of attention due to achieving state-of-the-art results on Natural Language Understanding benchmarks (Wang et al., 2018; Rybak et al., 2020). Our BERT usage as an encoder is inspired by (Reimers and Gurevych, 2019). We use a bag of words (BOW) embeddings model based on StarSpace (Wu et al., 2017b) as a simple baselines. The attention mechanism used in the transformer allows the model to learn the context and importance of words in an entity, whereas, in BOW embeddings, all of the words are treated equally and independently. Transformer architectures also allow the model to track the position of the words in a sequence, which we find beneficial in product matching task. To leverage our non-annotated data, we use the masked language model (MLM) pretraining objective described originally in the BERT paper (Devlin et al., 2019), but follow the improved training procedure from (Liu et al., 2019). We also explore scaling down the number of parameters with both the number of layers reduction and the knowledge distillation

approach (Sanh et al., 2019). For our product matching downstream task, we finetune the standard BERT model with an additional layer of 768 linear units on top of the mean pooled last layer BERT activations which we call eComBERT in the rest of the paper.

3.4 Batch construction strategy

The metric learning training aims at embedding similar items closer to each other than to the dissimilar ones. This objective is not optimized directly, but via a surrogate like margin-based triplet loss on a batch of triplets. The strategy for choosing which triplets to include in a batch heavily impacts the learning curve and the final performance of the model. If the negatives are too distant compared to the positives, the triplet loss is zero, and such triplet does not contribute to the progress of training. If the negatives are too similar to the anchors or positives, the model may be trained on noise. In our setup, the anchors and positives are well defined, as they are offer and positive product pairs from a batch of matches. However, the choice of negatives in non-trivial. One option is to select the negatives uniformly at random (Wu et al., 2017b). In metric learning, plenty of other strategies for selecting triplets exist, for example, batch hard (Hermans et al., 2017), semi-hard (Schroff et al., 2015), or distance weighted sampling (Wu et al., 2017a). Those strategies are designed for anchors and positive and negative items coming from the same domain, which is not the case in our problem. For a well-structured overview of possible strategies, we refer the reader to (Musgrave et al., 2019).

In our setup the triplets batch construction begins with sampling a batch of matches used as the anchor-positive pairs in triplets. For negative item selection we consider three strategies. The simplest strategy is a modification of the random negative selection. It randomly selects a negative from the non-matching products in the category of the anchor. For the purpose of this work, we name it *category random* (CR) strategy.

The second explored approach is a modification of the batch hard strategy adjusted for our problem setup. The standard batch hard strategy, as proposed in (Hermans et al., 2017), starts with a batch of items from K classes, with N items per class. Then, to construct the batch of triplets, it uses each item as an anchor, and for each anchor selects the least similar matching (from the same class) item as the *hard positive* and the most similar mismatching item as the *hard negative*. As in our setup anchors and positive/negative items belong to different domains, we need to modify this strategy. As before, it starts with a batch of anchor-positive pairs and then selects the negative as the most similar product from all the non-matching products in the sampled batch of matches. We refer to this strategy as *batch hard* (BH) in the experiments. The performance of this strategy depends on the batch size, as the larger is batch, the harder negatives are likely to be found.

For the third batch construction method, we propose the *category hard* (CH) strategy. It is similar to the category random strategy, but instead of selecting the negative at random from the products in a given category, it selects the one that is most similar to the anchor offer. It requires embeddings of all products, which are time and resource consuming to obtain. However, as the model and the similarity of products does not change too fast, we recompute the embeddings every several, e.g. 100 or 500, updates. This value should be set to properly trade-off the time spent in embedding the products, and the mismatch of the current state of the model and used product embeddings.

4 Experiments and results

In this section we present our experimental results, focusing on different aspects of the training pipeline. After briefly introducing the datasets and the baselines, we show different trade-offs that cover both the architectural choices for the model, and the batch construction strategies.

4.1 Datasets

We perform all the experiments using proprietary datasets composed by offer-product matches originating from a real-world e-commerce application. We conduct all the experiments using three datasets: ELECTRONICS, BEAUTY, and CULTURE. The three datasets differ mainly in the number of products and their main statistics are shown in Table 1.

	Available matches	Products
CULTURE	300K	800K
ELECTRONICS	200K	400K
BEAUTY	300K	200K

Table 1: Datasets used for our experiments. For each dataset, we report the number of training matches and the number of products.

4.2 Training setup

We split the available matches into train (80%) and test (20%) sets. Unless specified differently, in all experiments the test set was generated in such a way that half of the products originated from $\mathcal{D}_u$, while the other half did not. This test set better emulates the generalized zero-shot multi-class classification scenario we are dealing with, i.e., users may create an offer whose matching product already exists in the product catalog, or not. A deeper analysis of the zero-shot performances of our model is provided in Section 4.7.

After separately pretraining the specific language model, we trained one encoder for each dataset. Unless differently stated, we run all the experiments training each model for 5000 steps, setting the batch size to 32 and Adam optimiser (Kingma and Ba, 2014), with initial learning rate set to $2 \cdot 10^{-5}$.

The main evaluation metric we report is accuracy (often referred to as ACC@1 in similarity learning literature), i.e., the fraction of correctly matched offers. In particular, we consider that an offer matches a product when, given an offer representation in the embedding space, the closest product representation in terms of cosine distance is indeed the matching product.

4.3 Baselines

We compare eComBERT against the following baselines:

- a modified implementation of the StarSpace (Wu et al., 2017c) BOW encoder, a commonly used neural embedding baseline for similarity learning problems,
- non-finetuned[1] HerBERT (Rybak et al., 2020) a BERT-based encoder pretrained in RoBERTa's fashion on a big Polish language corpus,
- finetuned HerBERT, with an additional 768 dimension linear layer on the top of mean pooled last layer BERT activations (see Section 3.3),
- non-finetuned eComBERT.

Since language-specific BERT models perform better than general-purpose English models (Rybak et al., 2020), we do not include the latter among the baselines. To make a fair comparison, we apply the CR strategy and objective as described in Section 3.1 for all the experiments.

	CULTURE	ELECTRONICS	BEAUTY
BOW	0.8863	0.8032	0.7687
HerBERT-NFT	0.8206	0.6716	0.5542
HerBERT	0.9550	0.8580	0.9064
eComBERT-NFT	0.8208	0.6755	0.6127
eComBERT	**0.9777**	**0.8840**	**0.9219**

Table 2: Test accuracy per each dataset. NFT stands for *non-finetuned*.

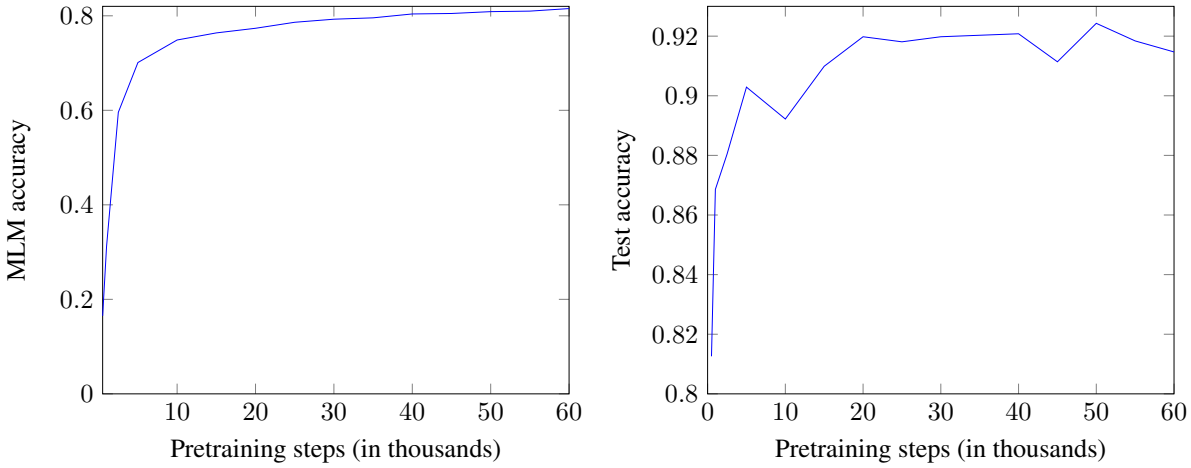
Results shown in Table 2 justify the application of more complex models such as BERT-based architecture over a simple BOW-based one for a product matching task. Moreover, we see how much we

[1]By *non-finetuned* we mean that we did not further train the weights of the encoder, but we used the mean pooled BERT last layer activations as embeddings for our entities.

improve pretrained BERT embeddings with our finetuning procedure. Finally, since the only difference between HerBERT and eComBERT is the corpus they were trained on, we conclude that pretraining the language model on a task-specific corpus (such as textual descriptions of offer and products in an e-commerce platform) leads to more accurate results.

4.4 Pretraining steps vs performance

Standard BERT model needs hundreds of thousands of update steps to converge and longer training generally translates into better downstream task performance. Since MLM objective is not strictly related to our product matching task, we check how the number of pretraining iterations affects the downstream task performance. Figure 1 shows text accuracy for the BEAUTY dataset for BERT models pretrained for a different number of steps. We see significant gains early on into pretraining, but after 20k steps into pretraining further optimization of the MLM objective does not translate into better downstream task performance, as model performance on product matching tasks fluctuates, even though MLM optimization does not converge. We have observed similar phenomena for different datasets and different BERT architectures and it suggests, that longer pretraining does not improve product matching performance.



(a) Masked language model prediction accuracy.

(b) Performance of BERT-based fine-tuned model (eComBERT) trained with CR batch construction strategy.

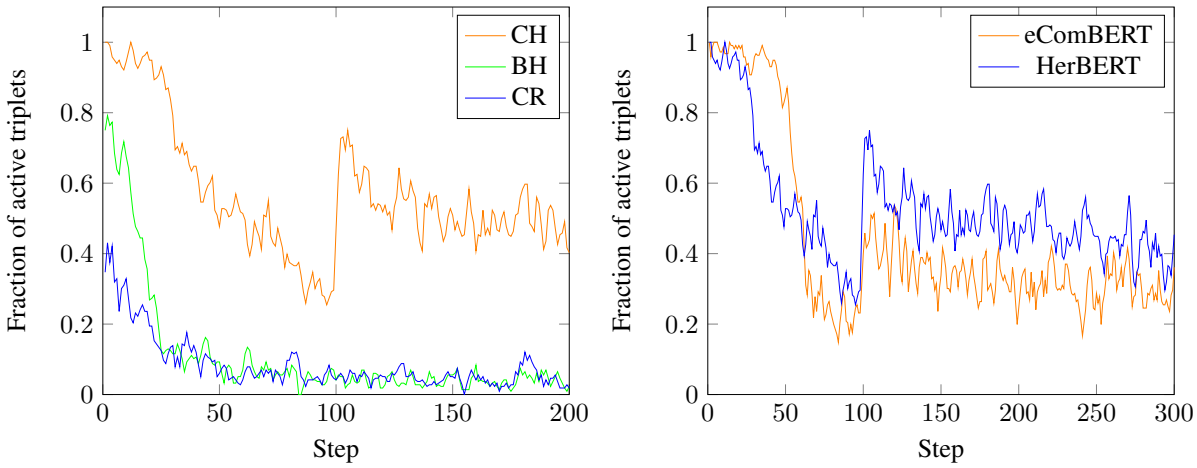
Figure 1: Pre-training and downstream task model performance.

4.5 Batch construction strategy

We explore the impact of the sampling strategy on the fraction of active triplets and the performance of the finetuned models. Particularly, we experiment with the three batch construction strategies proposed in Section 3.4: category random, batch hard and category hard.

First, we investigate how the fraction of active triplets changes depending on how the batch is constructed. In Figure 2, we show the smoothed fraction of active triplets for BERT-based models in the initial phase of finetuning. In Figure 2a, we compare the batch construction strategies for the HerBERT model. In the very first steps of finetuning, BH leads to more active triplets than CR, but after around 50 steps those two strategies perform on par. The proposed CH strategy results in much more active triplets, with nearly all triplets being active in the beginning. Later, the fraction drops, and after the product embeddings are recomputed after the 100th step, it raises sharply. In the following stage of finetuning, we do not observe such sudden changes, since the model has already started to converge and there is no dramatic difference between subsequent recomputations of the embeddings.

Next, in Figure 2b we compare the active triplets fraction for two encoder architectures: eComBERT and HerBERT, both employing the CH strategy. Interestingly, in the initial 60 training steps training, the fraction of active triplets is higher for eComBERT than for HerBERT, while later the situation changes, and the HerBERT encoder consistently yields more active triplets. This may be caused by the fact that



(a) Active triplet fraction for HerBERT initialised model for different negative item selection strategies.

(b) Active triplet fraction for HerBERT and eComBERT initialised model for category hard strategy.

Figure 2: Fraction of active triplets in the initial steps of training for BERT based models on ELECTRONICS dataset. Note: CH was run recomputing products' embeddings every 100 steps.

eComBERT better understands the e-commerce domain and places similar items closer to each other from the very beginning. HerBERT is a general model, and may not understand the similarity of product and offers well. In the later training steps HerBERT yields more active triplets than eComBERT as it yields worse training and test accuracy.

Finally, in Table 3 we report the test accuracy for HerBERT and eComBERT and different batch construction strategies. After 1000 steps the models are not yet converged, but it is already known which strategy gives the best performing model. First, we see that for HerBERT and eComBERT initialized models, the category hard strategy gives the highest accuracy, and the e-commerce eComBERT initialized model performs better. Secondly, we see that the models perform similarly for either batch hard or category random strategy, as those strategies produce little active triplets that may contribute to the change of the model. For the same reason, eComBERT trained with category random or batch hard strategy outperforms HerBERT with those strategies: eComBERT was a stronger model in the beginning, and the models have not changed much during training.

	category random	batch hard	category hard
HerBERT	0.8340	0.8352	0.9096
eComBERT	0.8803	0.8790	**0.9270**

Table 3: Test accuracy of models trained with different strategies for 1000 steps on ELECTRONICS.

4.6 Encoder architectures

BERT pretraining is very costly and its inference time is quite substantial in comparison to simpler models. To alleviate those issues, we ran eComBERT pretraining with 4 BERT layers (small eComBERT) and we pretrained DistilBERT on our own internal data (Distil eComBERT). In Table 4 we report test accuracies for the models on all of our prepared datasets. Those models still achieve competitive results across different domains, when cutting the inference time by half and two thirds, for Distil eComBERT and small eComBERT, respectively.

4.7 Zero-shot performance

In this section, we evaluate the zero-shot performance of our best eComBERT model against the BOW-based model and eComBERT without fine-tuning. To properly compare the performance, we prepare a separate dataset, where the test set consists only of products not seen in training. We use CH batch

	ELECTRONICS	BEAUTY	CULTURE
eComBERT	**0.9429**	**0.9674**	**0.9873**
Distil eComBERT	0.9410	0.9666	**0.9873**
small eComBERT	0.9400	0.9656	0.9865

Table 4: Accuracy of models with different BERT architectures trained for 5k steps with category hard sampling strategy.

construction and standard training procedure as discussed previously. To avoid leakage, we exclude test products from sampling. We report training and test accuracy in table 5. Based on those results, we conclude that the BOW model is not complex enough to overfit the training data and we observe similar results both on training and test data. Moreover, we see how much we improve pretrained embeddings with our finetuning procedure. After 5k update steps BERT starts to overfit to the data, while still generalizing fairly well on D_u.

	Training accuracy	Zero-shot accuracy
BOW	0.7929	0.8016
eComBERT-NFT	0.6519	0.6656
eComBERT	0.9086	**0.8873**

Table 5: Zero-shot performance of models trained for 5k steps using CH batch construction strategy on ELECTRONICS dataset.

5 Related work

E-commerce is inherently bound to products, and many challenges of this area are related to them. Such challenges include product entity resolution. Product entity resolution task is not well defined, and it is often formulated as one of the following problems: understanding which items from various sources (e.g. e-commerce platforms) correspond to the same real-world entity, i.e., a product (Li et al., 2020a; Fu et al., 2019); understanding which items (e.g. offers) from a single platform correspond to the same real-world entity (Shah et al., 2018). Last but not least, products and offers have normally multi-modal representations, i.e., they may be described by partially structured text and images at the same time. The way such representations are used when extracting information is a design choice, that highly influences the chosen algorithms and prepossessing strategies.

The former problem, product entity resolution of items from various sources, usually requires not only text matching but also understanding differently structured textual information. To this end, many solutions were proposed. Here we only mention works that use specific language models (Fu et al., 2019); that apply different machine translation techniques (Li et al., 2018), or other methods for comparison of attributes and titles (Li et al., 2020a). For more references related to this problem, we refer the reader to (Ristoski et al., 2018; Fu et al., 2019; Li et al., 2020a).

The latter problem, product matching, i.e., product entity resolution among items from a single platform, normally includes matching the item to a product defined in some product catalog. Such a problem was considered by (Shah et al., 2018), which compares two approaches to solve this task: classification based on fastText-based (Joulin et al., 2017) features; and similarity learning using BiLSTM neural network trained with contrastive loss. Many other case studies of this problem exist, for example (More, 2017), which describes a combination of learned similarities of texts and images neural networks.

6 Conclusions

In this work, we leverage the transformer architecture combined with the similarity learning approach to solve the product matching task in e-commerce. We show that finetuning even general-purpose transformers yields models that perform significantly better than other similarity learning baselines such as

StarSpace. These models can be further improved by employing BERT models pretrained on domain-specific data. Our experiments emphasize the importance of the employed batch construction strategy. Particularly, in our specific problem setup, a tailored batch construction strategy called *category hard* proved to significantly improve the percentage of active triplets and consequently the model performance.

One particular challenge that we encountered during our work with the product matching problem, and one which we believe has not been adequately addressed in the product matching literature is concept drift. This problem, also called covariate shift or nonstationarity, refers to the change in the relationship between input and output variables over time. In the case of product matching, it ultimately leads to a gradual drop in accuracy of the predicted product matches. Apart from the degradation of the model performance, another important issue arising from concept drift is the difficulty of evaluating new model versions. For a stationary problem, the standard procedure involves preparing a potentially costly human-annotated dataset once and using it for comparing the subsequent model version. Unfortunately, this is not a viable strategy if the test set cannot be fixed. We plan to investigate the concept drift problem in our future work.

References

Ursin Brunner and Kurt Stockinger. 2019. Entity matching on unstructured data: an active learning approach. In *2019 6th Swiss Conference on Data Science (SDS)*, pages 97–102. IEEE.

Ursin Brunner and Kurt Stockinger. 2020. Entity matching with transformer architectures - A step forward in data integration. In *Advances in Database Technology - EDBT*, volume 2020-March, pages 463–473.

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, June. Association for Computational Linguistics.

Cheng Fu, Xianpei Han, Le Sun, Bo Chen, Wei Zhang, Suhui Wu, and Hao Kong. 2019. End-to-End Multi-Perspective Matching for Entity Resolution. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence,* {*IJCAI-19*}, pages 4961–4967. International Joint Conferences on Artificial Intelligence Organization.

Alexander Hermans, Lucas Beyer, and Bastian Leibe. 2017. In defense of the triplet loss for person re-identification. *CoRR*, abs/1703.07737.

Elad Hoffer and Nir Ailon. 2015. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain, April. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Pradap Konda, Sanjib Das, Paul Suganthan GC, AnHai Doan, Adel Ardalan, Jeffrey R Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeff Naughton, et al. 2016. Magellan: Toward building entity matching management systems. *Proceedings of the VLDB Endowment*, 9(12):1197–1208.

Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of entity resolution approaches on real-world match problems. *Proceedings of the VLDB Endowment*, 3(1-2):484–493.

Maggie Yundi Li, Stanley Kok, and Liling Tan. 2018. Don't classify, translate: Multi-level e-commerce product categorization via machine translation. *arXiv preprint arXiv:1812.05774*.

Kai Li, Martin Renqiang Min, and Yun Fu. 2019. Rethinking zero-shot learning: A conditional visual classification perspective. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3583–3592.

Juan Li, Zhicheng Dou Dou, Yutao Zhu, and Ji-Rong Wen Zuo, Xiaochen Wen. 2020a. Deep cross-platform product matching in e-commerce. *Information Retrieval Journal*, 23(2):136–158.

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

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*.

Ajinkya More. 2017. Product matching in ecommerce using deep learning. `https://medium.com/walmartlabs/product-matching-in-ecommerce-4f19b6aebaca`.

Kevin Musgrave, Ser-Nam Lim, and Serge Belongie. 2019. Pytorch metric learning. `https://github.com/KevinMusgrave/pytorch-metric-learning`.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP/IJCNLP*.

Petar Ristoski, Petar Petrovski, Peter Mika, and Heiko Paulheim. 2018. A machine learning approach for product matching and categorization. *Semantic Web*, 9:707–728.

Piotr Rybak, Robert Mroczkowski, Janusz Tracz, and Ireneusz Gawlik. 2020. KLEJ: Comprehensive benchmark for polish language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1191–1201, Online, July. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August. 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, jun. Association for Computational Linguistics.

Andreas Thor. 2010. Toward an adaptive string similarity measure for matching product offers. *INFORMATIK 2010. Service Science–Neue Perspektiven für die Informatik. Band 1*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Chao-Yuan Wu, R. Manmatha, Alexander J. Smola, and Philipp Krähenbühl. 2017a. Sampling matters in deep embedding learning. *CoRR*, abs/1706.07567.

L. Wu, A. Fisch, S. Chopra, K. Adams, A. Bordes, and J. Weston. 2017b. Starspace: Embed all the things! *arXiv preprint arXiv:1709.03856*.

Ledell Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. 2017c. Starspace: Embed all the things! *CoRR*, abs/1709.03856.