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

allegro ML Research

Motivation

In e-commerce platforms, hundreds of millions of items are being listed for sale every day, thus providing a **satisfactory search and purchase experience** brings many challenges. One huge challenge for e-commerce portals is introducing **product-based experience**. From the buyer's perspective this means easy search and price comparability, while merchants benefit by having access to a **high-quality product catalog**, speeding up the listing process and providing more complete product descriptions.

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.

Contributions

- we apply state-of-the-art BERT-based models [1] 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 [7] 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.

Product matching with similarity learning

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. While **offers are vendor listed items** described by title, its text description, attributes, category, and photos, a **product represents a manufacturer's description of a good** and is described similarly. Recent papers mostly focus on using only the information contained in the titles or using both titles and attributes [4]. 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.

To solve the product matching problem with **triplet loss** [3], 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).

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

We choose **the transformer** [9] as based encoders. The transformer architecture gained a lot of attention due to achieving state-of-the-art results on Natural Language Understanding benchmarks [10, 6]. Our BERT usage as an encoder is inspired by [5].

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.

	Available matches	Products
culture	300K	800K
electronics	200K	400K
beauty	300K	200K

Baselines

We compare **eComBERT**, i.e., a standard BERT model with an additional layer of 768 linear units on top pretrained on **domain-specific data**, against the following baselines:

- a modified implementation of the **StarSpace** [12] BOW encoder, a commonly used neural embedding baseline for similarity learning problems,
- **non-finetuned HerBERT** [6], a BERT-based encoder trained on a big Polish language corpus,
- **finetuned HerBERT**, with an additional 768 dimension linear layer on top,
- **non-finetuned eComBERT**.

Since language-specific BERT models perform better than general-purpose English models [6], we do not include the latter among the baselines. To make a fair comparison, we apply the same sampling strategy and objective for all the baseline 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**

Encoder architectures

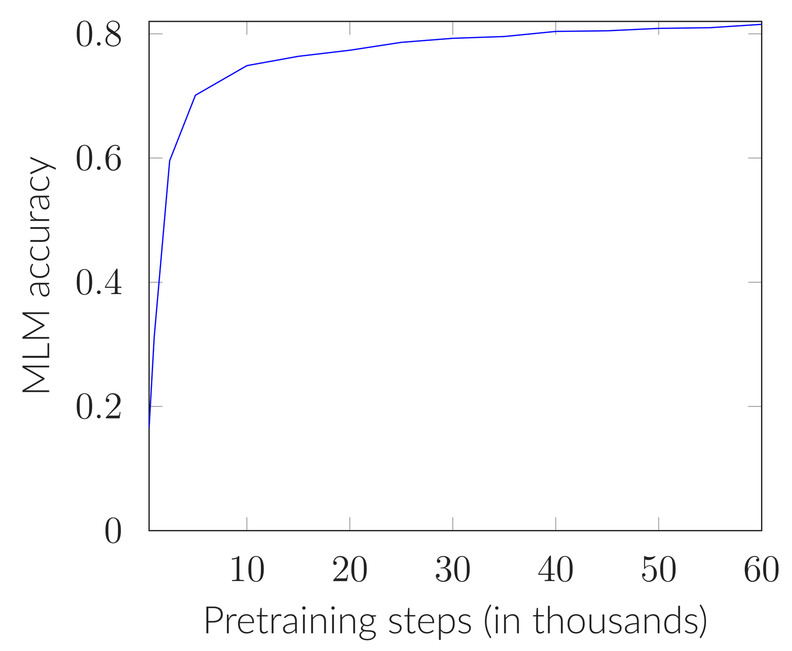
BERT pretraining is very costly and its inference time is quite substantial in comparison to simpler models. To alleviate those issues, we ran eComBERT pretraining with 4 BERT layers (small eComBERT) and we pretrained DistilBERT on our own internal data (Distil eComBERT). In Table 1 we report test accuracies 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.

	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

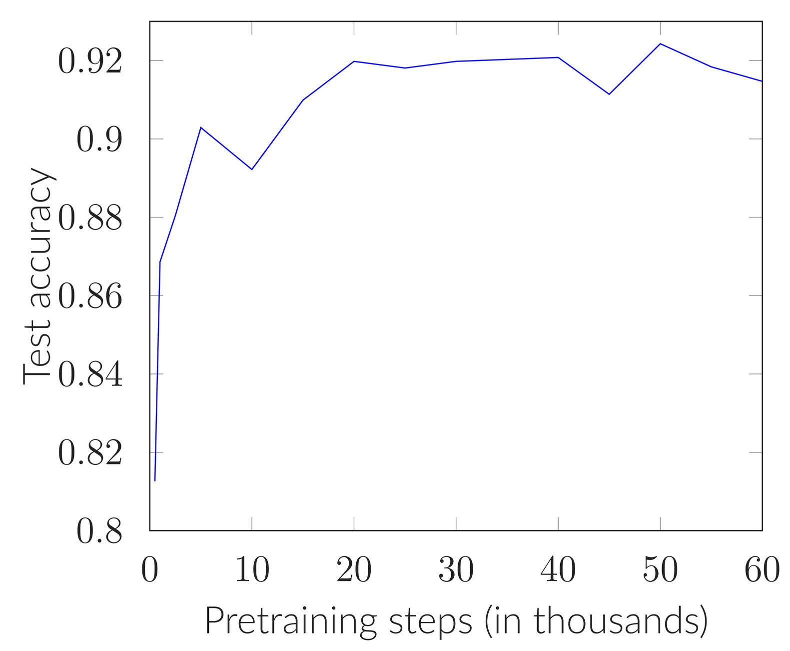
Table 1. Accuracy of models with different BERT architectures trained for 5k steps with category hard sampling strategy.

Pretraining steps vs performance

Standard BERT model requires **extensive pretraining**. Since MLM objective is not strictly related to our task, we check how the number of pretraining affects the downstream task performance. We observe **significant gains of test accuracy early on in pretraining**, but after around 20k steps product matching task performance fluctuates and further pretraining seems to be unnecessary.



(a) Masked language model prediction accuracy.

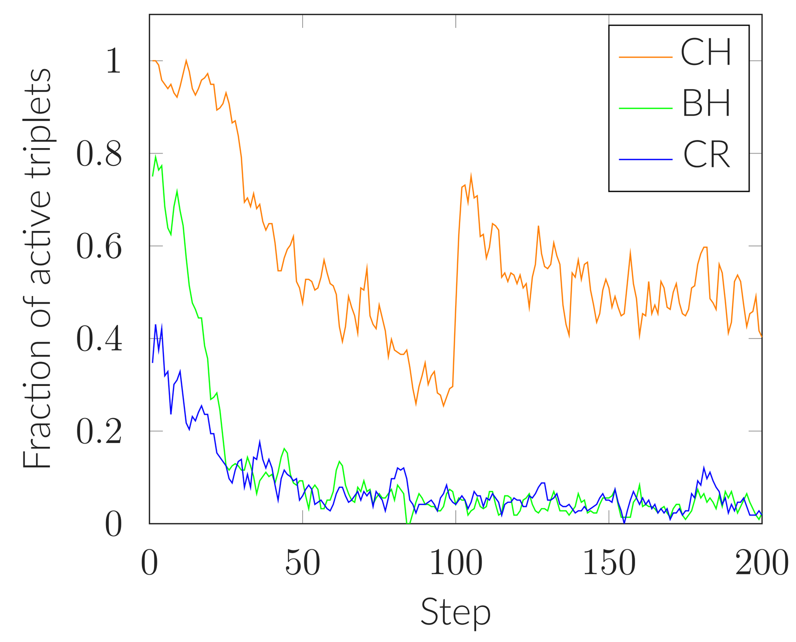


(b) Performance of BERT-based fine-tuned model (eComBERT) trained with CR batch construction strategy.

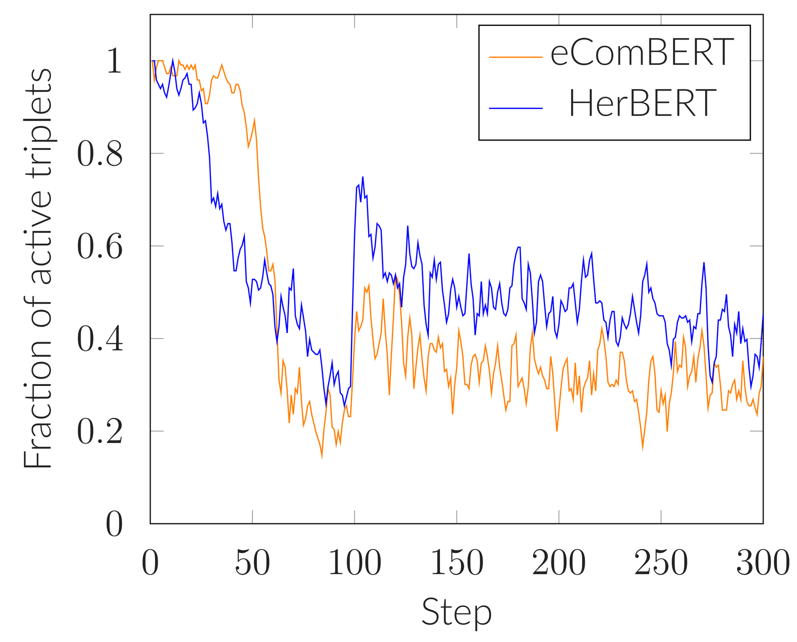
Batch construction strategy

The strategy of **choosing batch triplets** heavily impacts the learning curve and the final performance of the model. In metric learning, plenty of other strategies for selecting triplets exist, for example, batch hard [2], semi-hard [8], or distance weighted sampling [11]. In this work **we evaluate triplets batch construction**, including specifically tailored for offer-product matching problem:

- **category random (CR)** - randomly selects a negative from the non-matching products in the category of the anchor,
- **batch hard (BH)** - for each anchor selects the least similar matching item,
- **category hard (CH)** - our contributed sampling strategy, similar to category random strategy, but selects negatives most similar to the anchor offer.



(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.

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

Table 2. Test accuracy of models trained with different strategies for 1000 steps on electronics.

Conclusions

- BERT-based models combined with appropriately adopted similarity learning obtain high accuracy on offers with either observed or zero-shot products.
- Pre-training BERT-based models on domain specific data improves the model performance.
- Smaller BERT architectures can achieve comparable results to bigger model with significant increase in inference time.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 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 2019. Association for Computational Linguistics.

[2] A. Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification. *CoRR*, abs/1703.07737, 2017.

[3] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015.

[4] J. Li, Z. D. Dou, Y. Zhu, and J. W. Zuo, Xiaochen Wen. Deep cross-platform product matching in e-commerce. *Information Retrieval Journal*, 23(2):136–158, 2020.

[5] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP/IJCNLP*, 2019.

[6] P. Rybak, R. Mroczkowski, J. Tracz, and I. Gawlik. 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 2020. Association for Computational Linguistics.

[7] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[8] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[10] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[11] C. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl. Sampling matters in deep embedding learning. *CoRR*, abs/1706.07567, 2017.

[12] L. Wu, A. Fisch, S. Chopra, K. Adams, A. Bordes, and J. Weston. Starspace: Embed all the things! *CoRR*, abs/1709.03856, 2017.