Block-SCL: Blocking Matters for Supervised Contrastive Learning in Product Matching

Mario Almagro*
mario.almagro@nielseniq.com
NielsenIQ - Innovation
Madrid, Spain

David Jiménez*
david.jimenez@nielseniq.com
NielsenIQ - Innovation
Madrid, Spain

Diego Ortego*
diego.ortego@nielseniq.com
NielsenIQ - Innovation
Madrid, Spain

Emilio Almazán*
emilio.almazan@nielseniq.com
NielsenIQ - Innovation
Madrid, Spain

Eva Martínez Garcia
eva.martinezgarcia@nielseniq.com
NielsenIQ - Innovation
Madrid, Spain

ABSTRACT

Product matching is a fundamental step for the global understanding of consumer behavior in e-commerce. In practice, product matching refers to the task of deciding if two product offers from different data sources (*e.g.* retailers) represent the same product. Standard pipelines use a previous stage called "blocking", where for a given product offer a set of potential matching candidates are retrieved based on similar characteristics (*e.g.* same brand, category, flavor, *etc.*). From these similar product candidates, those that are not a match can be considered hard negatives. We present **Block-SCL**, a strategy that uses the blocking output to make the most of Supervised Contrastive Learning (SCL). Concretely, Block-SCL builds enriched batches using the hard-negatives samples obtained in the blocking stage. These batches provide a strong training signal leading the model to learn more meaningful sentence embeddings for product matching. Experimental results in several public datasets demonstrate that Block-SCL achieves state-of-the-art results despite only using short product titles as input, no data augmentation, and a lighter transformer backbone than competing methods.

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**; **Language models**; **Near-duplicate and plagiarism detection**.

KEYWORDS

product matching, blocking, e-commerce, contrastive learning, transformers

---

*M. Almagro, D. Jiménez, D. Ortego and E. Almazán contributed equally to this research. E. Martínez contributed to the writing of the document and previous discussions.

---

1 INTRODUCTION

Nowadays e-commerce websites offer hundreds of millions of products as a result of the online expansion of retailers' businesses. In this context, performing product matching successfully, *i.e.* automatically finding offers of the same product from different data sources is key to providing good quality of experience. The product matching task is particularly challenging due to a large number of existing products, their high heterogeneity, missing product information, and varying levels of data quality [24]. An example of product offers to match from different sources extracted from WDC dataset [17]: *"Kingston DataTraveler microDuo 3C 32GB Zilver - Prijzen Tweakers"* and *"Kingston Technology GmbH LambdaTek|USB flash drives"*. Product matching, then, requires a substantial language understanding and domain-specific knowledge [11] to differentiate between positive and negative pairs. We refer the reader to [20] for a detailed overview of product matching challenges.

The current trend in product matching is to use the Transformer paradigm [11, 16, 24], which relies on robust pre-trained Transformer models to serve as starting point for solving downstream tasks [4, 32]. For example, the authors in [1, 11, 28] concatenate the two product descriptions and subsequently process them using a Transformer encoder, *e.g.* BERT [4], and add a classification head on top to solve product matching as a binary classification task.

Alternatively, R. Peters and C. Bizer [16] process the product descriptions individually and estimate separate description embeddings, which are then compared in a metric learning fashion and classified as match or not match. Metric learning frameworks are very popular in natural language processing (NLP) for a variety of applications including: text classification [2], sentence representation [6], named entity recognition [22] or text retrieval [27]. These metric learning methods work by pulling together representations of the same concept and pushing apart representations for different concepts. Currently, the methods based on Supervised Contrastive Learning (SCL) are leading the progress in metric learning [9]. SCL proposes a training objective that, within a batch of training samples, considers positives for each sample those of the same class and as negatives the remaining ones. This vanilla version of SCL has been adapted or improved in different contexts, *e.g.* text classification [2], product matching [16], learning with label noise [14] or long-tailed classification [3]. In particular, authors in [16] use vanilla-SCL for product matching, where positive/negative pairs are simply positive/negative pairs of product offers. We adopt a

similar approach and focus on a key aspect overlooked in [16]: the hard negatives.

Product matching usually involves a blocking pre-processing step to discard entries that are unlikely to be matches, thus reducing the matching problem to the resulting blocking entries. Moreover, hard negatives play a key role in achieving better similarity learning [26, 29], thus we believe that the blocking is a natural source of hard negatives that can boost metric learning for product matching.

In this paper, we propose to use blocking information to enhance supervised contrastive learning for product matching in e-commerce. The main contributions of the paper are listed below:

- We demonstrate that considering hard negatives from the blocking information during the batched optimization benefits the learning process and boosts product matching performance.
- We evaluate our approach in 6 public datasets comparing results with top-performing methods. Moreover, in the ablation study, we show the impact of adding more positive/ negative samples in the blocking.
- We demonstrate that solely using hard negatives for SCL without using larger backbones and tricks from related methods obtains the best overall results while improving the efficiency of the models considerably.

2 RELATED WORK

The problem of product matching refers to the task of finding product offers in a large dataset given a textual query for product offers. The main challenge lies in the fact that query and candidate offers come from different domains *e.g.* retailers. In practice, this problem is approached with a two-stage pipeline:

(1) Blocking: a light comparison on the entire dataset is conducted with the goal of identifying a subset of descriptions that are potential matches for the product query. It is expected to obtain a subset without false negatives.
(2) Matching: the descriptions from the "blocking" stage are classified as match or not match for the input query.

Most works found in the literature deal only with the second stage in isolation. In particular, the problem is treated as a pair classification problem, where each pair is independent of the other. However, we believe that there is relevant information in the blocking output that we can use to improve the matching stage.

Early solutions for product matching were based on hand-crafted features and rules that provided a high level of interpretability at the cost of involving human expert knowledge [5, 21]. Deepmatcher [13] can be considered the first deep learning based model for product matching. This method generates independent features for each description using Recurrent Neural Networks, which are then compared and classified as match or not match.

More recently, Transformer-based approaches have shown good performance on solving this task [1, 11, 15, 28]. These methods usually concatenate pairs of textual product offers and feed them to a large pre-trained language model (*e.g.* BERT) that is fine-tuned with a classification head that provides matching/not matching predictions.

Nevertheless, the performance of Transformer-based models is directly related to the quality of the sentence embeddings used.

Since BERT-based models do not produce independent single sentence embeddings, they are generated by averaging the outputs of the BERT model or by using the special [CLS] token. However, these sentence embedding methods have shown limitations to properly capture sentence semantics [12, 18, 31]. Also, the pre-training task significantly impacts the embeddings quality [7]. Many works leverage the information from BERT models to build sentence embeddings [10, 19], and those using self-supervised contrastive learning approaches achieve the state-of-the-art results. Some of them focus on building positive sentence pairs [6, 30], or using prompt learning [8, 25] to reduce token embeddings biases and make the BERT layers capture better sentence semantics that help the network distinguish between soft and hard negative examples.

To the best of our knowledge, there is only one recent work that learns sentence embeddings for product matching via contrastive learning [16]. We name this approach Vanilla-SCL as they follow the original SCL [9] random strategy for assembling the batches. Nevertheless, they ignore the information from the blocking, which is a source of hard negatives. Recent works [24] have shown the benefits of using hard negatives and positives to obtain better products embeddings. We identify the potential benefits of using hard negative examples enclosed in the information from the blocking filtering when applying supervised contrastive learning and propose to include that information during batch construction.

3 APPROACH

The proposed approach relies on two steps: pre-training using SCL with blocking information and product matching classification. First, we train the backbone and projection network to produce robust sentence representations so that offers of the same product are separable from the rest. Then, we freeze the backbone and replace the projection head with a linear layer to perform product matching classification using pairs of product offers.

3.1 Pre-training

Broadly speaking, we adopt the SCL approach from [9] proposed for visual representation learning and adapt it to learn sentence embeddings for product matching. This is achieved by pulling offers of the same product closer in the feature space and pushing away those of other products during training. Figure 1 presents an overview of the pre-training proposal.

First, we use the sampling strategy depicted in Sec. 3.1.2 to select a batch $\mathcal{B}_i$ of product offers x_i, where $i \in [1, |\mathcal{B}_i|]$ and $|\mathcal{B}_i|$ is the batch size. The purpose of using blocking information relative to each product is to identify the product offers that should lie closer in the feature space and thus force the model to learn representations that differentiate them. Every product offer is then mapped to a low-dimensional representation z_i by learning a transformer encoder network f_θ and a multi-layer perpectron projection head g_ϕ with parameters θ and ϕ, respectively. In particular, the sentence embedding $v_i = f_\theta(x_i)$ is generated by average pooling of token embeddings produced by f_θ. Subsequently, the sentence embedding is transformed into a low-dimensional representation $w_i = g_\phi(v_i)$, which is later L_2-normalized into $z_i = w_i/\|w_i\|_2$. Then, the contrastive loss function described in Sec. 3.1.3 is calculated.

$$L_{BSC}$$

Blocking 1		Blocking 'n'	
$z_{p1}^1 \dots z_{pk}^1$	$z_{n1}^1 \dots z_{nq}^1$	○ ○ ○	$z_{p1}^n \dots z_{pk}^n$	$z_{n1}^n \dots z_{nq}^n$

Projection head

Pooling

| $T_{[CLS]}$ | T_{tok_1} | ○ ○ ○ | T_{tok_M} | $T_{[SEP]}$ |

Transformer encoder

| [CLS] | tok_1 | ○ ○ ○ | tok_M | [SEP] |

Blocking-based sequential batch

Blocking-based sequential batch construction

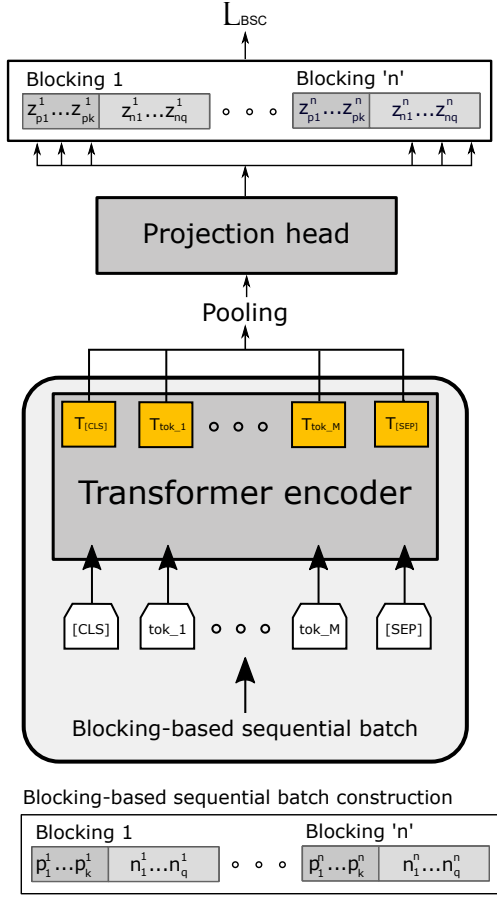| Blocking 1		Blocking 'n'	
$p_1^1 \dots p_k^1$	$n_1^1 \dots n_q^1$	○ ○ ○	$p_1^n \dots p_k^n$	$n_1^n \dots n_q^n$

Figure 1: Diagram of the proposed pre-training strategy and the blocking-based batch construction, where 'p' and 'n' refers to positives and negatives offers within the same blocking, respectively.

3.1.1 Data preparation. Most of the datasets in product matching (*e.g.* WDC [17]) consist of m pairs of relatively similar product offers labeled to denote if they refer to the same product (match) or not (non-match). Then, each data pair includes a product offer on the left (left$_i$), a product offer on the right (right$_i$), and a binary matching label y_i. The nature of the data is that matching/non-matching offers correspond to the same/different product, sharing all of them a high degree of textual similarity given that they come from the same blocking. Therefore, we can assume that non-matching pairs provide a source of hard negative samples.

Although product and blocking identifiers are not always provided, such information can be generated using the set of offer pairs. Those pairs sharing some offer would constitute a blocking, while only matching offers would have the same product id. Therefore, we propose to use a match-based graph to assign unique product identifiers to all matching offers, while assigning the same blocking identifier to all non-matching products compared with a specific product.

3.1.2 Blocking-based batch construction. Batches are built to contain product offers x_i from n different blockings, so offers with the same product id are considered positive samples, and offers not sharing the product id but sharing the blocking id are assumed to be hard negative samples. To do so, data is sampled in such a way that we obtain a set of product ids in every batch for which we have a set $\mathcal{P}_i$ of positive offers and a set $\mathcal{N}_i$ of hard negative ones. The cardinality of $|\mathcal{P}_i| \leq k$ and $|\mathcal{N}_i| \leq q$, being k and q hyper-parameters that denote the number of positive and negative samples in a blocking to be selected for each product id, *e.g.* $k = 2$ selects 2 positive offers for the product id associated to sample x_i.

Note that the sum of positive and negative samples in the batch $|\mathcal{P}_i| + |\mathcal{N}_i| = |\mathcal{B}_i| - 1$, where -1 denotes the self-contrast case. This batch creation strategy forces the network to distinguish between positive and negative pairs that have similar text sequences as they belong to the same blocking, as well as unrelated offers coming from the remaining blockings.

3.1.3 SCL loss. The supervised contrastive learning loss is defined as follows:

$$\mathcal{L}_{SCL} = \frac{1}{|\mathcal{B}_i|} \sum_{i \in \mathcal{B}_i} \frac{1}{|\mathcal{P}_i|} \sum_{p \in \mathcal{P}_i} - \log \frac{\exp\left(z_i \cdot z_p / \tau\right)}{\sum_{b \in \mathcal{B}_i \setminus x_i} \exp\left(z_i \cdot z_b / \tau\right)}, \quad (1)$$

where τ is a temperature scaling constant and both the numerator and denominator compute inner products between pairs of offers embeddings. The former does it in-between positive offers, while the latter normalizes it using all offers in the batch $\mathcal{B}_i$ except the current i offer. Minimizing Eq. 1 implies adjusting θ and ϕ to pull together the feature representations z_i and z_p of positive product offers, while pushing apart z_i from all negative product offers in $\mathcal{B}_i \setminus \mathcal{P}_i$. The gradient analysis of $\mathcal{L}_{SCL}$ in [9] reveals this training objective focuses on hard positives/negatives rather than easy ones. Introducing the blocking information via $\mathcal{P}_i$ and $\mathcal{N}_i$ promotes data-to-data relations between hard positive/negative samples.

3.2 Product matching classification

This last stage computes binary matching predictions for offer pairs by learning a linear layer on top of the frozen pre-trained transformer backbone. Figure 2 shows an overview of the product matching classification. The classification head receives the concatenation of the sentence embeddings from the two offers to match, v_i and v_j, together with the concatenation of the Euclidean distance and cosine similarity of vi and v_j. Then, the classifier layer input is $(v_i, v_j, |v_i - v_j|, v_i * v_j)$. For simplicity the two last terms are referred as *similarity* in Figure 2. Note that the concatenation of this type of information is a standard practice in other works [10, 16] to help converge to top matching performance, something we have also observed in our experiments. Furthermore, the previously mentioned strategy assumes that v_i goes before v_j in the concatenation classifier, while there is no reason not to use the reversed order. Therefore, we also add the reversed concatenation order $(v_j, v_i, |v_i - v_j|, v_i * v_j)$ and average the logits of both orders, which we experimentally saw to slightly help in boosting matching performance.
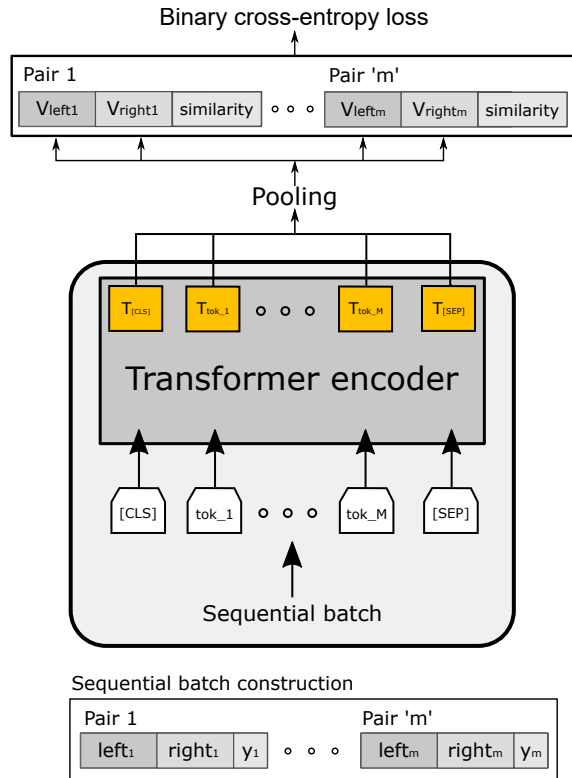
Figure 2: Diagram of the proposed downstream strategy and the sequential pair-based batch construction.

4 EXPERIMENTS

In this section we describe first, the datasets used to validate our approach, then the experimental setup and the results obtained compared to several related works. Finally, we present an ablation study on the influence of the number of positives and negatives in the construction of the blocking within the batch.

Table 1: Datasets general statistics: train split.

Dataset	Size	#pos. pairs	#neg. pairs	#offers	#prods.
Abt-Buy	def.	822	6, 837	2, 112	1, 084
Amzn-Goog.	def.	933	8, 234	3, 445	2, 279
WDC-Comp.	small	722	2, 112	2, 790	745
	med.	1, 762	6, 332	3, 846	745
	large	6, 146	27, 213	4, 238	745
	xlarge	9, 690	58, 771	4, 307	745

4.1 Datasets

We conduct all our experiments using 6 public datasets that are commonly used in the field of e-commerce and product matching: Amazon-Google [13], Abt-Buy [13] and WDC [17] (all variants

Table 2: Datasets general statistics: test split.

Dataset	#pos. pairs	#neg. pairs	#prods.
Abt-Buy	206	1, 710	921
Amzn-Goog.	234	2, 059	1, 962
WDC-Comp.	300	800	745

Table 3: Datasets blocking statistics: train set.

Dataset	Size	Average block. size	Average pos/block.	Average neg/block.
Abt-Buy	def.	11.9 ± 13.4	1.2 ± 1.0	10.6 ± 13.1
Amzn-Goog.	def.	15.0 ± 125.5	2.8 ± 41.1	12.1 ± 84.8
WDC-Comp.	small	6.1 ± 2.6	1.5 ± 0.8	4.5 ± 2.4
	med.	17.3 ± 6.7	3.7 ± 1.8	13.5 ± 6.1
	large	71.6 ± 30.3	13.1 ± 9.4	58.4 ± 25.3
	xlarge	146.8 ± 92.4	20.7 ± 22.7	126.0 ± 77.7

of the computers subset, *i.e. small*, *medium*, *large* and *xlarge*). Tables 1 and 2 shows general statistics of the train and test splits. Unlike other methods [16], we only use the *title* as an input feature (or equivalently the shortest textual information used to describe the product offer), which greatly simplifies the training, *i.e.* fewer memory requirements, and computations, while achieving better performance.

Additionally, we inspect the statistics related to the blocking information as shown in Table 3. In the table, we present three main columns: *Avg. block. size*, which refers to the number of product textual descriptions that are candidates to be a match for each query product description. The *Avg. pos/block* representing the match descriptions out of the candidates and, the *Avg. neg/block* for the non-match descriptions. Note that each query product might have different blocking sizes as well as the number of positive and negative descriptions, thus we present the average statistics across all query products. We can observe significant differences in these three aspects between datasets, for example, a $\sim 30\times$ increase in the number of negatives per blocking in the *xlarge* version compared to the *small* version. Note that the specific characteristics of the blocking in each dataset might yield different optimal configurations of our method. For instance, for datasets with more number of negatives in the candidates' list we might want to select more negatives per blocking to increase variability in the batch.

4.2 Experimental setup

We use a pre-trained BERT medium (BERT-med)[1] as the backbone (6 layers, 8 attention heads, and 512 hidden dimension size) in all the experiments and a cosine annealing scheduler for the learning rate with 5% of warm-up period and maximum value of $5 \cdot 10^{-5}$.

For the supervised contrastive learning pre-training, we use a projection head composed of one linear layer of dimension 512 followed by *gelu* activation, dropout with probability 0.1, *LayerNorm*, and a final linear projection to a low-dimensional representation

---

[1]https://huggingface.co/google/bert_uncased_L-6_H-512_A-8

Table 4: F1-score results on the test set of each dataset. The symbol ♣ denotes that we run our implementation of the method. The results marked with * where taken from [16]. The † symbol indicates the rows used to aggregate results. Bold denotes the best performing results and underlined the second best.

| Backbone | #param. | Approach | Abt-Buy | Amzn-Goog. | WDC-Computers | | | | Avg. |
					small	medium	large	xlarge	
RoBERTa	110M	Ditto baseline [11]	91.05*	65.92	86.37*	91.90*	94.68*	94.73*	88.80*
		Ditto [11]	89.33	75.58	-	-	-	-	86.90†
		Vanilla SCL (w/ tricks) [16]	**93.70**	79.28	**93.18**	**97.66**	**98.16**	**98.33**	<u>93.38</u>
BERT-base	110M	JointBERT [15]	-	-	77.55	88.82	96.90	97.49	-
DistilBERT	66M	Ditto [11]	-	-	80.76	88.62	91.70	95.45	86.90†
BERT-med	35.3M	Fine-tuned LM	10.43	21.09	10.03	26.86	70.73	79.16	36.38
		Vanilla SCL (w/o tricks) [16]♣	90.95	<u>80.69</u>	87.26	92.63	94.67	94.67	90.14
		Blocking SCL (Ours)	<u>93.30</u>	**86.61**	<u>90.49</u>	<u>96.97</u>	<u>98.00</u>	<u>97.83</u>	**93.86**
Bi-LSTM	-	DeepMatcher	62.8	70.70	61.22	69.85	84.32	88.95	72.97

of 256. We train for 300 epochs in Abt-Buy, Amazon-Google, and WDC-small-computers, 400 epochs in WDC-medium-computers, 600 epochs in WDC-large-computers, and 800 epochs in WDC-xlarge-computers. Regarding the hyper-parameters related to the blocking, for the smaller datasets Abt-Buy, Amazon-Google and WDC-small-computers we found the optimal configuration with 1 positive and 16 negatives while, for the rest we used 2 positives and 16 negatives. In general we have observed stability of the models on the blocking parameters, however, a more rigorous analysis will be required to understand the relation between dataset and the selection of these hyper-parameters. After training, we select the model checkpoint with the lowest validation loss for the subsequent fine-tuning.

To estimate a product matching prediction, we replace the projection head learned during pre-training with a linear layer that maps the 512 dimensional sentence embedding from the Transformer backbone to the matching/not-matching space. We train for 50 epochs freezing the backbone and use early-stopping if the validation loss does not decrease in 10 epochs. For evaluation, we use the checkpoint with the highest F1-score in validation.

4.3 Results

Table 4 reports the resulting F1 scores of our experiments along with top-performing methods in 6 commonly used public datasets. The compared methods are DITTO [11], Vanilla-SCL [16], Joint-BERT [15] and DeepMatcher [13]. We name the recent method in [16] as Vanilla-SCL given that it performs random sampling of negatives following [9], as opposed to our Block-SCL method that uses negative mining from blocking information. For Vanilla-SCL we report the results from [16] using RoBERTa-base backbone, which we name Vanilla-SCL with tricks *(w/ tricks)* due to using data augmentation and serialization of multiple input features (title, description and additional information) with additional tagging of the data field and value using special tokens [COL] and [VAL]. We further reproduce Vanilla-SCL without tricks *(w/o tricks)*, *i.e.* no

data augmentation and using only the title with no tagging strategy, and run it ourselves using BERT-med backbone to enable fair comparison with our Block-SCL.

Our method surpasses by a large margin (↑ 7.3) the results of the previous top-performing method [16] in the Amazon-Google dataset. This dataset is the less saturated one in terms of performance, giving sufficient room for improvement, unlike the other 5 datasets, where related work performance ranges from 93.16 up to 98.1 F1-score. In the remaining datasets, our method achieves comparable results to [16] using a model 3× smaller and a more modest training strategy *e.g.* smaller batch-sizes and input sequences. Regarding efficiency, we conducted a preliminary evaluation of training times simulating the configuration proposed by [16], the results are presented in Table 6. Our configuration is significantly more efficient, having a 4× and 2× speed up with BERT-med and RoBERTa, respectively. These results suggest that building complex batches with hard negatives brings benefits during training, leading the convergence of the model to a more optimal minimum, without requiring large architectures.

The fine-tuned LM approach in Table 4 refers to using the pre-trained BERT-med backbone from a general domain and entirely fine-tuning it on the downstream task of product matching following the approach described in section 3.2. This strategy performs poorly, thus demonstrating that the pre-trained model is key to learning meaningful sentence embeddings for product matching.

Due to computational restrictions, we were not able to run our method with RoBERTa-base and the same configuration proposed in [16]. However, we should expect some improvement when using RoBERTa-base (110M) compared to BERT-med (35.3M), as bigger architectures, trained on larger datasets and batch sizes usually come with performance gains in supervised contrastive learning [9]. Moreover, data augmentation in NLP is known to be hard, so we do not apply it given that using labels, *i.e.* product identifiers in our scenario, reduces the importance of data augmentation in contrastive learning [23].

4.4 Ablation studies

A key component of our approach is the construction of the batches, as described in 3.1.2. In this stage, we sample a certain number of positive and negative pairs from each blocking. However, it is unclear how many positives and negatives we should select. Table 5 shows the results of not using the blocking in the batches, as in Vanilla-SCL, and presents Block-SCL performance when varying the number of positives and negatives. Regarding the impact of including hard-negatives in the batch we report an improvement of 2.8 points with respect to not using hard-negatives, *i.e.* random sampling. Additionally, we observe a clear tendency for improvement as the number of negatives increases in the batch. These results verify the effectiveness of our approach and our initial hypothesis that the use of hard negatives from the blocking examples within the same batch should lead the model convergence to a more optimal solution. Finally, we do not observe any impact of the number of positive samples chosen from each blocking, which suggests that positives within a blocking are relatively redundant and provide a very similar training signal. Nevertheless, further analysis need to be conducted to gain a more solid understanding of these parameters influence.

The experiments in this section are run on the WDC Computers medium dataset using the same configuration except for the hyperparameters stated in the Table 5. Concretely, we used the BERT-med architecture, the training was optimized with AdamW using a warm-up of 5% followed by a cosine annealing learning rate decay strategy. For the pre-training stage we used a batch size of 256, and trained for 200 epochs with an initial learning rate of 10^{-4}. In the product matching classification, we used a batch size of 64 during 50 epochs with an initial learning rate of $5 \cdot 10^{-5}$. Finally, we report the average of 3 runs along with the standard deviation, which demonstrates the stability of Block-SCL.

Table 5: Influence of the number of positives/negatives in WDC-medium. We report the average and standard deviation of the F1-score values after running 3 repetitions of Block-SCL with different seeds.

Blocking	#pos	#neg	F1
No	1	0	93.15 ± 0.52
Yes	1	8	95.96 ± 0.20
	2	8	95.91 ± 0.46
	3	8	96.32 ± 0.15
	6	8	96.17 ± 0.13
Yes	1	1	93.88 ± 0.31
	1	2	94.89 ± 0.33
	1	4	95.01 ± 0.52
	1	16	96.13 ± 0.22
Yes	2	16	96.53 ± 0.15

Table 6: Computational time between different settings.

Approach	Backbone	it/s	epoch/min
Block-SCL (Ours)	BERT-med	9.30	2
	RoBERTa	2.84	0.80
Vanilla SCL [16]	BERT-med	2.31	0.66
	RoBERTa	1.25	0.18

5 CONCLUSIONS

In this paper, we tackle the problem of product matching in e-commerce and focus on learning robust representation of products using supervised contrastive learning as training loss. This learning objective based on metric learning suits the product matching task, which essentially measures similarities between textual product offers. Building on top of this intuition, we incorporate the blocking information within the optimization loop in a way that every batch randomly samples several disjoint blockings (unique product ids). We empirically demonstrate that negative samples within the blocking introduce a strong signal in the supervised contrastive loss formulation, obtaining high-quality product embeddings.

In the experimental section, we demonstrate that our proposed Block-SCL method is able to learn more discriminative product embeddings by measuring the product matching performance based on learning a product matching classifier on top of those embeddings. We outperform most related work methods by a large margin and clearly improve over Vanilla-SCL [16] when compared in a fair setup without tricks. Nevertheless, considering Vanilla-SCL with tricks (data serialization of multiple product fields and data augmentation) and a larger transformer backbone, we perform on par, while being more than 4 times faster.

We leave for future work an in-depth analysis on the impact of the number of blockings in a batch, and positives and negatives within each blocking, as well as experimenting with larger backbones and tricks whose impact is not clear in previous work.

REFERENCES

[1] U. Brunner and K. Stockinger. 2020. Entity matching with transformer architectures-a step forward in data integration. In *International Conference on Extending Database Technology (ICEDT)*.

[2] Q. Chen, R. Zhang, Y. Zheng, and Y. Mao. 2022. Dual Contrastive Learning: Text Classification via Label-Aware Data Augmentation. *arXiv: 2201.08702* (2022).

[3] Jiequan Cui, Zhisheng Zhong, Shu Liu, Bei Yu, and Jiaya Jia. 2021. Parametric Contrastive Learning. In *IEEE/CVF International Conference on Computer Vision (ICCV)*.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics (ACL)*.

[5] W. Fan, X. Jia, J. Li, and S. Ma. 2009. Reasoning about record matching rules. In *International Conference on Very Large Data Bases (VLDB)*.

[6] T. Gao, X. Yao, and D. Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[7] Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *North American chapter of the Association for Computational Linguistics (NAACL)*.

[8] Ting Jiang, Shaohan Huang, Zihan Zhang, Deqing Wang, Fuzhen Zhuang, Furu Wei, Haizhen Huang, Liangjie Zhang, and Qi Zhang. 2022. PromptBERT: Improving BERT Sentence Embeddings with Prompts. *arXiv preprint arXiv:2201.04337* (2022).

[9] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. 2021. Supervised Contrastive Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[10] Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from bert for semantic textual similarity. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 9119–9130.

[11] Y Li, J. Li, Y. Suhara, A. Doan, and W.-C. Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. In *International Conference on Very Large Data Bases (VLDB)*.

[12] Chandler May, Alex Wang, Shikha Bordia, Samuel R Bowman, and Rachel Rudinger. 2019. On measuring social biases in sentence encoders. *arXiv preprint arXiv:1903.10561* (2019).

[13] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. 2018. Deep Learning for Entity Matching: A Design Space Exploration. In *ACM International Conference on Management of Data (ICDM)*.

[14] D. Ortego, E. Arazo, P. Albert, N.E O'Connor, and K. McGuinness. 2021. Multi-Objective Interpolation Training for Robustness to Label Noise. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[15] R. Peeters and C. Bizer. 2021. Dual-Objective Fine-Tuning of BERT for Entity Matching. In *International Conference on Very Large Data Bases (VLDB)*.

[16] R. Peeters and C. Bizer. 2022. Supervised Contrastive Learning for Product Matching. *arXiv: 2202.02098* (2022).

[17] A. Primpeli, R. Peeters, and C. Bizer. 2019. The WDC Training Dataset and Gold Standard for Large-Scale Product Matching. In *World Wide Web Conference (WWWC)*.

[18] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the Behaviors of BERT in Ranking. *arXiv preprint arXiv:1904.07531* (2019).

[19] N. Reimers and I. Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

[20] K. Shah, S. Kopru, and J.-D. Ruvini. 2018. Neural Network based Extreme Classification and Similarity Models for Product Matching. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), Industry Papers)*.

[21] R. Singh, V. Meduri, A. Elmagarmid, S. Madden, P. Papotti, J.-A. Quiané-Ruiz, A. Solar-Lezama, and N. Tang. 2017. Generating concise entity matching rules. In *ACM International Conference on Management of Data (ICDM)*.

[22] S. Snigdha Sarathi Das, A. Katiyar, R.J. Passonneau, and R. Zhang. 2022. CON-TaiNER: Few-Shot Named Entity Recognition via Contrastive Learning. In *Association for Computational Linguistics (ACL)*.

[23] Y. Tian, C. Sun, B. Poole, P. Krishnan, C. Schmid, and P. Isola. 2020. What Makes for Good Views for Contrastive Learning?. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[24] J. Tracz, P.I. Wójcik, K. Jasinska-Kobus, R. Belluzzo, R. Mroczkowski, and I. Gawlik. 2020. BERT-based similarity learning for product matching. In *Association for Computational Linguistics (ACL), Workshop on Natural Language Processing in E-Commerce*.

[25] Hao Wang, Yangguang Li, Zhen Huang, Yong Dou, Lingpeng Kong, and Jing Shao. 2022. SNCSE: Contrastive Learning for Unsupervised Sentence Embedding with Soft Negative Samples. *arXiv preprint arXiv:2201.05979* (2022).

[26] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl. 2017. Sampling Matters in Deep Embedding Learning. In *IEEE International Conference on Computer Vision (ICCV)*.

[27] L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P.N. Bennett, J. Ahmed, and A. Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *International Conference on Learning Representations*.

[28] S. Xu, S.E. Li, and Y. Yang Xiang. 2020. A Pre-trained Matching Model Based on Self- and Inter-ensemble For Product Matching Task. In *International Semantic Web Conference (ISWC), Semantic Web Challenge on Mining the Web of HTML-embedded Product Data*.

[29] H. Xuan, A. Stylianou, X. Liu, and R. Pless. 2020. Hard negative examples are hard, but useful. In *European Conference on Computer Vision (ECCV)*.

[30] Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. Consert: A contrastive framework for self-supervised sentence representation transfer. *arXiv preprint arXiv:2105.11741* (2021).

[31] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675* (2019).

[32] L. Zhuang, L. Wayne, S. Ya, and Z. Jun. 2021. A Robustly Optimized BERT Pre-training Approach with Post-training. In *Chinese National Conference on Computational Linguistics (CCL)*.