Intermediate Training of BERT for Product Matching

Ralph Peeters
Data and Web Science Group
University of Mannheim
Mannheim, Germany
ralph@informatik.uni-mannheim.de

Christian Bizer
Data and Web Science Group
University of Mannheim
Mannheim, Germany
chris@informatik.uni-mannheim.de

Goran Glavaš
Data and Web Science Group
University of Mannheim
Mannheim, Germany
goran@informatik.uni-mannheim.de

ABSTRACT

Transformer-based models like BERT have pushed the state-of the-art for a wide range of tasks in natural language processing. General-purpose pre-training on large corpora allows Transformers to yield good performance even with small amounts of training data for task-specific fine-tuning. In this work, we apply BERT to the task of product matching in e-commerce and show that BERT is much more training data efficient than other state-of-the-art methods. Moreover, we show that we can further boost its effectiveness through an intermediate training step, exploiting large collections of product offers. Our intermediate training leads to strong performance (>90% F1) on new, unseen products without any product-specific fine-tuning. Further fine-tuning yields additional gains, resulting in improvements of up to 12% F1 for small training sets. Adding the masked language modeling objective in the intermediate training step in order to further adapt the language model to the application domain leads to an additional increase of up to 3% F1.

CCS CONCEPTS

• **Information systems → Entity resolution**; **Electronic commerce**; • **Computing methodologies → Neural networks**.

KEYWORDS

e-commerce, product matching, deep learning

1 INTRODUCTION

Product matching is the task of deciding if offers originating from different web-shops refer to the same real-world product. This is a central task for e-commerce applications such as online market places, price comparison portals, as well as for the construction of product knowledge graphs [36] such as the one currently built by Amazon [10]. Different merchants present their products in different ways, leading to heterogeneity among offers of the same product, which makes product matching a challenging task.

In natural language processing (NLP), deep Transformer networks [33], pre-trained on large corpora via language modeling objectives [7, 8, 22, *inter alia*] significantly pushed the state-of-the-art in a variety of downstream tasks [15, 34], including a number of sentence-pair classification tasks, e.g. paraphrase identification [9]. Recent studies [4, 21] also demonstrate the effectiveness of Transformer models like BERT [8] for the task of entity matching.

In this work, we show that fine-tuning BERT for product matching is much more training data efficient than the state-of-the-art framework *Deepmatcher* [24]. Fine-tuning BERT results in 15-20% higher F1 scores in settings with small- and medium-sized training sets. Even for large training sets, fine-tuning BERT still yields a 2% improvement over *Deepmatcher*.

Inspired by findings that intermediate training on large training sets for related tasks [28, 30] improves downstream performance, we next introduce an intermediate training step before the final fine-tuning of the model for specific products. In this step, we train BERT on product data from thousands of e-shops and show that intermediate training leads to high performance (>90% F1) and good generalization to new products, even without any product-specific fine-tuning. Poor generalization to new products is the main weakness of *Deepmatcher* [24], as shown in our previous work [26]. Our intermediate training is particularly beneficial for fine-tuning setups with limited training data: it leads to improvements of up to 12% F1 on new products with small training datasets, compared to direct fine-tuning (i.e. without any intermediate training). Finally, we show that adding domain-specific (self-supervised) language modeling to the intermediate training leads to further gains of up to 3% F1 in downstream product-matching tasks.

All code and data of our experiments is available on GitHub[1] which makes all results reproducible.

2 BERT FOR PRODUCT MATCHING

Deep Transformer-based models like BERT [8] use stacked encoder layers based on a self-attention mechanism [33], which allows every (sub-)word to attend to every other (sub-)word in a sequence, enabling mutual semantic contextualization of words. The deep architecture, i.e. stacking of attention layers, allows for modeling of syntactic and semantic compositionality of the language that stems from word interactions [14]. Unlike static word embeddings [3, 23, 27], where each word has one fixed vector regardless of the context, pre-trained Transformers produce context-specific vector representations of words, allowing, *inter alia*, to capture different word senses (e.g. *bank* would have very different representations in contexts in which it denotes a *financial institution* from those in contexts where it denotes a *river bank*). BERT is pre-trained on a large corpus of text (concatenation of Wikipedia and BookCorpus) using two pre-training objectives: (1) The masked language modeling objective (MLM) aims to reconstruct (i.e. predict) words that have been masked out in the input text from the context; (2) The next sentence prediction (NSP) objective predicts if two sentences are adjacent to each other in text or not – contributing to downstream performance of text-pair classification tasks. The input to the BERT model has the following format: `[CLS] Sequence 1 [SEP] Sequence 2 [SEP]`. Two sequences, comprising (sub-)word

[1] https://github.com/Weyoun2211/productbert-intermediate

Table 1: Test and training set statistics

	# products w/ pos (overall)	# Pos. Pairs	# Neg. Pairs	# Comb. Pairs
Test set				
computers	150 (745)	300	800	1,100
Training sets				
xlarge	745	9,690	58,771	68,461
large	745	6,146	27,213	33,359
medium	745	1,762	6,332	8,094
small	745	722	2,112	2,834

tokens, are separated using `[SEP]` tokens; the sequence start token `[CLS]` serves to capture the representation of the whole text-pair.

After the pre-training step, it it possible to either use the output representations of each word in downstream tasks (feature-based approach) or to fine-tune the BERT model itself for these tasks (fine-tuning-based approach), with the latter generally leading to better performance. In this work, we adopt the standard fine-tuning for sentence-pair classification: we feed the transformed representation of the sequence start token `[CLS]`, $\mathbf{x}_{CLS}$ into a simple logistic regression classifier: $y = \sigma(\mathbf{x}_{CLS}\mathbf{W}_{cl} + b_{cl})$, with $\mathbf{W}_{cl}$ and $\mathbf{b}_{cl}$ as well as BERT's parameters being optimized during fine-tuning.

2.1 Datasets

In our experiments, we use the training, validation and gold standard (test) datasets from the *computers* category of the WDC Product Corpus for Large-Scale Product Matching [26]. These datasets are derived from schema.org annotations from thousands of webshops extracted from the Common Crawl. Relying on schema.org annotations of product identifiers like GTINs or MPNs allows us to directly create binary (matching or non-matching) labels for our classification task, without the need for laborious manual annotation. All labels of the test set used for final evaluation have been manually checked. Previous experiments with these datasets have shown that using schema.org ids as distant supervision results in clean enough labels for training high-performance product matchers [26].

The *computers* test set encompasses positive pairs for 150 unique products. The negative pairs for these products contain offers for 595 additional products. The corresponding training sets contain both positive and negative pairs for the same products. For more details on the construction of the product corpus as well as the training and test sets, we refer the reader to [29] and to the project website[2]. To test the efficiency of the classifiers w.r.t. training size, we experiment with training sets of varying size: *small*, *medium*, *large*, *xlarge*. Table 1 shows statistics of the training sets and test set.

2.2 Fine-tuning Setup

We cast product matching as a binary classification task, i.e. given two offers, we predict if they represent the same real-world product. Input for BERT (Sequence 1 and 2) is then the concatenation of the product data of each offer. To this end, we first concatenate all attributes of each product offer into one string. We use the attributes *brand*, *title*, *description* and *specification table content* and concatenate them in this order.

Experimental setup. We conduct all our experiments with PyTorch [25] using BERT's implementation[3] from the HuggingFace Transformers library [35]. All hyperparameters are set to their defaults if not stated otherwise. We minimize the binary cross-entropy loss using Adam [17] as optimization algorithm. BERT allows for input sequences of maximal length of 512 tokens: we first constrain each attributes length to *5* (*brand*), *50* (*title*), *100* (*description*) and *200* (*specification table content*) words respectively, dropping any words outside that range, and further truncate long product offers by removing tokens from their end until we satisfy BERT's constraint. We fine-tune all layers for 50 epochs with a linearly decaying learning rate with warm-up over the first epoch. We use the validation set for model selection and early stopping: if the F1 score on the validation set does not improve over 10 consecutive epochs, we stop the training. We use a fixed batch size of 32 and sweep learning rates in the range [5e-6, 1e-5, 3e-5, 5e-5, 8e-5, 1e-4]. We train three model instances for each hyperparameter configuration and report the average performance.

Baselines. We compare BERT-based product matching with several baselines. First, we evaluate a simple word co-occurrence based approach, where we feed binary bag-of-words features of the two product offers to traditional classification algorithms. We also test the Magellan framework [18] for entity resolution which generates string- and numeric-similarity based features. Magellan constructs these features depending on the data types of the input attributes. We combine both the Magellan and the word co-occurrence feature creation methods with *XGBoost*, *Random Forest*, *Decision Tree*, *linear SVM*, and *Logistic Regression* as classification methods and apply randomized search over the respective hyperparameter spaces. Finally, we compare against *Deepmatcher* [24], a state-of-the-art neural entity resolution framework using pre-trained word embeddings as input. *Deepmatcher* computes attribute-wise similarities between two records and then combines these as features for the matching decision. For *Deepmatcher*, we use fastText embeddings trained on the English Wikipedia[4] as input and allow for the fine-tuning of word embeddings, which, albeit not part of the original implementation, has been shown to improve performance [26]. We train all *Deepmatcher* instances for 50 epochs with default parameters and only search for the optimal learning rate. For *Deepmatcher* and BERT we use the method specific tokenizers for pre-processing, for the other baselines we lower-case all attributes before further processing.

2.3 Fine-tuning Results

Table 2 compares the results of fine-tuning BERT to the baselines. BERT outperforms all three baselines in all settings. The gains from BERT-based product matching become larger the smaller the training dataset is: for the smallest training set, BERT outperforms *Deepmatcher* by 20 F1 points. Even for the largest training set, we

[2] http://webdatacommons.org/largescaleproductcorpus/v2/

[3] We used the following pre-trained BERT instance: `bert-base-uncased`.

[4] https://fasttext.cc/docs/en/pretrained-vectors.html

Table 2: BERT compared to baselines

	Word Cooc.			Magellan			deepmatcher			BERT			Li et al. [21]
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	F1
xlarge	86.59	79.67	82.99	71.44	56.89	63.33	89.63	94.78	92.12	95.99	93.00	**94.47**	95.45
large	79.52	77.67	78.58	67.67	63.67	65.60	85.70	91.22	88.38	91.64	95.00	**93.29**	91.70
medium	65.83	78.33	71.54	48.99	81.56	61.20	66.39	82.78	73.67	84.89	94.22	**89.31**	88.62
small	53.98	74.67	62.66	50.86	71.22	59.17	54.86	69.56	61.20	75.62	89.33	**81.89**	80.76

obtain 2.3% F1 gain over *Deepmatcher*. Our results are in line with the findings of Li et al. [21], though not fully comparable, as the authors use DistilBERT [31] and apply additional data augmentation techniques. Overall, we can conclude that fine-tuning BERT is a promising technique for product matching, especially in settings with limited training data.

Table 3: Intermediate training set statistics

	# products w/ pos (overall)	# pos. pairs	# neg pairs	# comb. pairs
computers only	60,030 (286,356)	409,445	2,446,765	2,856,210
4 categories	201,380 (838,317)	858,308	2,665,056	3,523,364

3 INTERMEDIATE TRAINING ON DOMAIN-SPECIFIC DATA

BERT has been pre-trained on a general-purpose natural language corpus, whose language as well as topics are rather different from product descriptions. We thus test the intuitive assumption that intermediate in-domain training – after BERT's original pre-training and before fine-tuning for specific products – can improve matching performance. For the intermediate training we use training data covering a wide range of products from thousands of e-shops.

3.1 Building Intermediate Training Sets

We leverage the WDC Product Corpus for Large-Scale Product Matching [29] and its product-cluster structure to build wide coverage training sets consisting of millions of offer pairs. The corpus consists of clusters containing offers for the same product. The clusters have been derived using schema.org annotated ids as weak supervision (see Section 2.1). In order to have an unbiased evaluation, the clusters contained in the test set and fine-tuning training sets are removed from the corpus prior to building the intermediate training sets.

We compare the effects of intermediate training on two structurally different training sets. The first intermediate training set contains only offer pairs for the category *computers*: this allows us to introduce more *computer* information into BERT and have the Transformer network detect relevant linguistic phenomena for recognizing matches between *computer* offers. The second training set contains pairs from four categories – *computers*, *cameras*, *watches* and *shoes* – with fewer training pairs per product: this offers a wider selection of products (i.e., more versatile information about what constitutes a product match for the model), but less in-depth information for each product/category.

We build the training sets as follows: for positive instances, we select only clusters containing more than one offer, from which we can build at least one positive pair. We restrict ourselves to clusters of size ≤80 after observing that very large clusters contain more noise and may lead to degradation of performance. For each offer in each cluster we build up to 15 (computers) or 5 (4 categories) positive pairs with the other offers from that cluster. Half of those are *hard positives*, created by a) applying cosine similarity between bag-of-words vectors of concatenation of title and the first 5 words of description and b) sorting offer pairs by cosine similarity and selecting pairs with the lowest scores. The remaining 50% are selected by randomly pairing offers from the same cluster. We create negative pairs in a similar fashion: for each offer taken for positives pairs, we create the same amount of negatives pairs using offers from other clusters of the same category. *Hard negatives* (50%) are pairs of offers from different clusters with the highest cosine similarity; the other half are randomly sampled pairs of offers from different clusters. Table 3 displays the statistics of the resulting intermediate training sets.

3.2 Intermediate Training Procedure

For the first set of experiments, the intermediate training is performed with a single objective, the binary product matching task. The architecture is exactly the same as for the fine-tuning experiments. One model is trained for each of the training sets from Table 3. After intermediate training, we evaluate the model with and without final product-specific fine-tuning. We run the intermediate training for 40 epochs with a linearly decaying learning rate (starting from 5e-5) with 10,000 warmup steps and a batch size of 256. Due to the long training times we train the first 90% of epochs on sequences of length 128 and only the last 10% on the full sequences of 512 tokens to speed up training, similar to the original BERT training procedure [8].

In the second set of experiments, we add the MLM objective to the product matching objective and jointly optimize both in the intermediate training step. We follow the original masking procedure: we randomly select 15% of tokens for replacement; in 80% of the cases, we replace the token with the [MASK] token, in 10% of the cases with a random vocabulary token, and in the remaining 10% we keep the original token (i.e., we give up the replacement). As in the original work, we train the Transformer network by minimizing the cross-entropy loss over predictions of masked tokens. After the intermediate training, we again evaluate two model variants: with and without the final product-specific matching fine-tuning.

Table 4: Intermediate training with PM objective

	intermediate training							
	computers category				4 categories			
	P	R	F1	Δ only fine-tune	P	R	F1	Δ only fine-tune
xlarge	95.58	93.67	94.61	0.14	95.45	95.44	95.45	0.98
large	92.68	95.56	94.09	0.80	91.34	96.00	93.61	0.32
medium	94.01	95.78	94.88	5.57	91.59	95.67	93.59	4.28
small	94.38	93.11	93.73	11.84	90.39	90.89	90.64	8.75
none	94.41	90.00	92.15	-2.32 (xl)	88.24	95.00	91.49	-2.98 (xl)

Table 5: Intermediate training with PM and MLM objective

	intermediate training - PM + MLM				
	P	R	F1	Δ only fine-tune	Δ interm. only PM
xlarge	98.20	96.56	97.37	2.90	2.76
large	94.99	96.67	95.82	2.53	1.73
medium	96.05	97.11	96.58	7.27	1.70
small	95.64	97.44	96.53	14.64	2.80
none	94.31	94.00	94.16	-0.31 (xl)	2.01

3.3 Intermediate Training Results

Table 4 shows the results of the intermediate training procedure. We compare the intermediate training on the computers training set against the intermediate training on the training set comprising 4 product categories. We observe that even without final fine-tuning (row 'none' in Table 4), we achieve a very good matching performance of 92% F1. This suggests that through the intermediate training we inject category-specific knowledge into BERT's parameters, as it is evidently able to make good matching predictions for products for which it had not seen any training examples. Once the intermediate model is subjected to further fine-tuning on offer pairs from the training sets, we observe further improvements in all settings, with gains being most prominent for the smallest training set. Intermediate training followed by fine-tuning on small training sets reaches a performance of ~94% F1, which, without intermediate pre-training (see Table 2), we previously obtained only on the largest training set. Training on category-specific data (*computers*) generally yields marginally better performance than training on the mix of 4 categories..

Table 5 shows the results of adding the MLM objective to the product matching objective in the intermediate training step using the *computers* intermediate training set. Compared to the corresponding settings in which the intermediate training did not include MLM (see left half of the Table 4), the performance (with fine-tuning) increases by up to 3% F1, yielding a new top overall matching performance (>97% F1 for the largest training set and 96% F1 for all other training sizes). This confirms the findings from other application domains [2, 20] pointing to benefits of domain-specific MLM pre-training. The original pre-training data likely only contains few instances of product-specific vocabulary, as it covers a wide range of topics. Applying intermediate MLM training on domain-specific data allows for adaptation of the vocabulary embeddings to the domain, resulting in better downstream performance.

In summary, subjecting BERT to an intermediate training step with large amounts of product data leads to a model that generalizes well to new unseen products from the same category and can be easily fine-tuned with small amounts of product-specific training data to further increase the performance for these products. Depending on the structure of the intermediate training set, more training data for a single category can lead to a small increase in performance compared to a more heterogeneous training set encompassing a larger set of products from several categories. Adding the MLM objective to the intermediate training results in further improvements in matching performance, suggesting that domain-specific language modeling indeed successfully adapts BERT's parameters to the product domain.

4 RELATED WORK

Product matching, a task with rich history and large body of work in both research and industry, can be seen as a special case of *entity resolution*, which concerns itself with the disambiguation of entity representations to their respective real-world entity [5, 6]. Early approaches applied rule- and statistics-based methods [12]. Since the early 2000s, machine learning based methods have taken the focus due to their strong performance [19]. In recent years, due to the successes of deep learning in fields like computer vision and natural language processing, researchers working on entity-matching started to shift their attention towards these methods as well [1, 11, 13, 16, 24, 32, 37]. Recently, Transformer-based architectures [8, 33] were shown to produce state-of-the-art results [4, 21].

5 CONCLUSION

Transformer-based language models like BERT have had a tremendous impact in the field of NLP, improving the state-of-the-art performance in a wide variety of tasks. In this work, we demonstrate the utility of BERT for product matching in e-commerce, showing that it is much more training data efficient than *Deepmatcher*. Performing intermediate training of BERT with large amounts of product data from thousands of e-shops leads to a model with high generalization performance (>90% F1) for new (i.e. unseen) products. We show that, if submitted to intermediate training, BERT reaches peak performance with less product-specific training data than without intermediate training. We achieve the best performance if intermediate training combines two jointly-trained objectives: (1) binary product-matching and (2) masked language modeling. Category-specific intermediate training yields only slightly better performance than intermediate training on cross-category data. While intermediate product-matching training alone brings substantial gains, adding the masked language modeling objective to the intermediate training gives an additional performance edge of up to 3% F1 in all setups. This is in line with observations from other domains, such as scientific text [2, 20], that domain-specific language modelling improves the performance of BERT for in-domain downstream tasks.

REFERENCES

[1] Luciano Barbosa. 2019. Learning Representations of Web Entities for Entity Resolution. *International Journal of Web Information Systems* 15, 3 (2019), 346–358.

[2] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. 3606–3611.

[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[4] Ursin Brunner and Kurt Stockinger. 2020. Entity Matching with Transformer Architectures - a Step Forward in Data Integration. In *Proceedings of the International Conference on Extending Database Technology, 2020*. 463–473.

[5] Peter Christen. 2012. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer-Verlag, Berlin Heidelberg.

[6] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. 2019. End-to-End Entity Resolution for Big Data: A Survey. *arXiv:1905.06397 [cs]* (2019).

[7] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-Training Text Encoders as Discriminators Rather Than Generators. *arXiv:2003.10555 [cs]* (March 2020). arXiv:cs/2003.10555

[8] 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*. 4171–4186.

[9] William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing*. 9–16.

[10] Xin Luna Dong, Xiang He, Andrey Kan, Xian Li, Yan Liang, et al. 2020. AutoKnow: Self-Driving Knowledge Collection for Products of Thousands of Types. *arXiv:2006.13473 [cs]* (June 2020).

[11] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed Representations of Tuples for Entity Resolution. *Proceedings of the VLDB Endowment* 11, 11 (2018), 1454–1467.

[12] Ivan P. Fellegi and Alan B. Sunter. 1969. A Theory for Record Linkage. *J. Amer. Statist. Assoc.* 64, 328 (1969), 1183–1210.

[13] Cheng Fu, Xianpei Han, Le Sun, Bo Chen, Wei Zhang, et al. 2019. End-to-End Multi-Perspective Matching for Entity Resolution. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. 4961–4967.

[14] John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 4129–4138.

[15] Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, et al. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. In *Proceedings of the International Conference on Machine Learning*. 7449–7459.

[16] Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. 2019. Low-Resource Deep Entity Resolution with Transfer and Active Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 5851–5861.

[17] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]* (Dec. 2014). arXiv:cs/1412.6980

[18] Pradap Konda, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, et al. 2016. Magellan: Toward Building Entity Matching Management Systems. *Proceedings of the VLDB Endowment* 9, 12 (2016), 1197–1208.

[19] 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 (2010), 484–493.

[20] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, et al. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36, 4 (2020), 1234–1240.

[21] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *arXiv:2004.00584 [cs]* (April 2020).

[22] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, et al. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692* (2019).

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Conference on Neural Information Processing Systems*. 3111–3119.

[24] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, et al. 2018. Deep Learning for Entity Matching: A Design Space Exploration. In *Proceedings of the 2018 International Conference on Management of Data*. 19–34.

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. 8024–8035.

[26] Ralph Peeters, Anna Primpeli, Benedikt Wichtlhuber, and Christian Bizer. 2020. Using schema.org Annotations for Training and Maintaining Product Matchers. In *Proceedings of the 10th International Conference on Web Intelligence, Mining and Semantics*.

[27] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1532–1543.

[28] Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088* (2018).

[29] Anna Primpeli, Ralph Peeters, and Christian Bizer. 2019. The WDC Training Dataset and Gold Standard for Large-Scale Product Matching. In *Workshop on e-Commerce and NLP (ECNLP2019), Companion Proceedings of WWW*. 381–386.

[30] Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, et al. 2020. Intermediate-Task Transfer Learning with Pretrained Models for Natural Language Understanding: When and Why Does It Work?. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 5231–5247.

[31] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. *arXiv:1910.01108 [cs]* (2020).

[32] 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 Association for Computational Linguistics, Volume 3 (Industry Papers)*. 8–15.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, et al. 2017. Attention Is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 6000–6010.

[34] Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R Thomas McCoy, et al. 2019. Can You Tell Me How to Get Past Sesame Street? Sentence-Level Pretraining Beyond Language Modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4465–4476.

[35] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, et al. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv* abs/1910.03771 (2019).

[36] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Product Knowledge Graph Embedding for E-Commerce. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 672–680.

[37] Dongxiang Zhang, Yuyang Nie, Sai Wu, Yanyan Shen, and Kian-Lee Tan. 2020. Multi-Context Attention for Entity Matching. In *Proceedings of The Web Conference 2020*. 2634–2640.