Integrating Product Data using Deep Learning

Ralph Peeters
Data and Web Science Group
University of Mannheim

Christian Bizer
Data and Web Science Group
University of Mannheim

Abstract

Product matching is the task of deciding whether two product descriptions refer to the same real-world product. Product matching is a central task in e-commerce applications such as online market places and price comparison portals, as these applications need to find out which offers refer to the same product before they can integrate data from the offers or compare product prices. Product matching is a non-trivial task as merchants describe products in different ways and as small differences in the product descriptions matter for distinguishing between different variants of the same product. A successful approach for dealing with the heterogeneity of product offers is to combine deep learning-based matching techniques with large amounts of training data which can be extracted from Web corpora such as the Common Crawl. Training deep learning methods involving millions of parameters for use cases such as product matching requires access to large compute resources. In this extended abstract, we report how we trained different RNN- and BERT-based models for product matching using the bwHPC infrastructure and how this extended training allowed us to reach peak performance. Afterwards, we describe how we use the bwHPC infrastructure for our ongoing research on table representation learning for data integration.

1 Introduction

The meteoric rise of deep learning techniques over the recent years has touched many different research fields and led to large improvements on various tasks. This development has especially been driven by research in Computer Vision and Natural Language Processing which popularized neural architectures such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and lately Transformers. As the neural networks were constantly increased in size, large amounts of processing power are necessary to train current models. For example, one of the most widely used Transformer models is BERT [3] developed by Google. The BERT architecture consists of multiple encoder stacks, each applying bi-directional self-attention to an input sequence, resulting in a contextualized representation of the sequence and its tokens after the last layer. BERT has ∼340 million parameters and pre-training the model on a large text corpus took Google 256 TPU days. Training such models often exceeds the capabilities of the compute infrastructure of single university research groups and the provision of shared compute resources through initiatives such as bwHPC is thus a necessary prerequisite to enable university groups to conduct research in this space.

In this extended abstract, we discuss the results of our recent research [8, 9, 10] on product matching using deep learning with respect to the employed bwForCluster MLS&WISO compute resources. Afterwards, we give an overview of our ongoing research in the area of table representation learning for data integration for which we use the bwUniCluster 2.0. The presented results show that RNN- and BERT-based deep learning models can significantly outperform previous symbolic matching methods on less structured textual product offers from different online shops but require large amounts of training data and computational power in the form of GPUs in order to achieve their full potential. Our ongoing work requires even more compute resources due to the large scale multi-gpu (pre-)training of Transformer models.

2 Product Matching Experiments

Product matching is the task of deciding whether two product descriptions refer to the same real-world product. Product matching is a central task for e-commerce applications such as online market places and

Proceedings of the 7th bwHPC Symposium (ISBN 978-3-948303-29-7)
Article No. 11; pages 59–62

price comparison portals. For example, price comparison portals need to collect product offers from different online-stores and subsequently perform the product matching step to disambiguate offers for the same products. They can then present this information to their customers to help them find the cheapest offers for a desired product. Product matching is not a trivial task because, for marketing reasons, merchants present products in their e-shops in different ways. This leads to heterogeneous representations of the same product due to e.g. a different focus of the descriptions, different usage of abbreviations, symbols, and units (MB vs GB) or simply due to manually introduced errors like typos or missing information. Product matching has a long history in research and practice. Early approaches to product matching applied rule- and statistics-based methods. Since the early 2000s machine learning-based methods dominate product matching. Due to the successes of deep learning in fields like computer vision and natural language processing, the research focus has shifted towards applying these methods also for product matching [6, 7].

Learning a high quality product matcher requires large amounts of training data in the form of heterogeneous product descriptions originating from different sources. Many e-shops have started to annotate (mark-up) product offers within their HTML pages using standardized terms from the schema.org vocabulary[1] in order to enable search engines to extract offers and display them in the context of e-commerce applications such as Google Shopping. A part of the e-shops also annotate product IDs such as *GTIN* or *MPN* numbers within their HTML pages which allow offers for the same product to be grouped together. The HTML pages of the e-shops are included in public Web corpora such as the Common Crawl[2] and it is thus possible to extract large amounts of product offers from this corpus and group them into training pairs using the product IDs [1]. For the experiments presented in this paper, we use the training, validation and test sets from the WDC Product Data Corpus for Large-scale Product Matching (WDC LSPC) [11] which has been extracted from the Common Crawl by our group. We use the training, validation, and test sets for the four categories *computers*, *cameras*, *shoes* and *watches*. The training sets are available in four sizes, labeled *small*, *medium*, *large* and *xlarge*, ranging from ∼2,000 to ∼70,000 product offer pairs. We use the attributes *brand*, *title*, *description* and *specTableContent* for all experiments. The three latter attributes are highly textual and contain longer sequences of words. Further statistics about the datasets can be found on the WDC LSPC website[3].

We use these datasets to perform product matching experiments with various baselines and neural models. In our first set of experiments, we applied the Magellan [5] entity matching framework and a word co-occurrence baseline as well as the Deepmatcher [7] framework to the problem of product matching. The Deepmatcher framework offers a deep neural network architecture for the problem of entity matching and a selection of submodules like RNNs or attention mechanisms that can be applied. In the corresponding paper [10], presented at WIMS2020, we investigate the performance of all models along various dimensions. These are (i) training set size and (ii) feature selection for all models and specifically for Deepmatcher we further experiment with (iii) all available modules, (iv) using pre-trained or self-trained embeddings and (v) allowing end-to-end training by also updating the embedding layer during training. Additionally, we optimized hyperparameters for the computers xlarge dataset to showcase maximum possible performance. The result of these experiments showed that an end-to-end trained RNN-based Deepmatcher model using pre-trained fastText embeddings is able to consistently outperform the baselines across all training set sizes. For the largest training set, the model is able to reach F1s above 90%, up to nearly 96% for some product categories.

The Deepmatcher models require GPUs to train in a reasonable amount of time. We train every configuration three times and average the results to account for the inherent randomness of the training process. Due to the large amount of experiments (overall ∼4000 distinct runs), in addition to the resources available at our university we made heavy use of the bwForCluster MLS&WISO Production, specifically the GPU nodes with 16 Haswell CPU cores and 2 NVIDIA Tesla K80 GPUs each. Most of the runs were executed on the bwHPC machines, submitting 4 processing jobs at once, training 2 models per node, one on each GPU. In summary across all models and setups, these experiments required ∼48 days of training time spread over a timeframe of 4 months on the bwHPC machines. CPU utilization was not an issue during these experiments, as the 16 cores of the Haswell machines were more than enough for supplying the GPUs with the preprocessed data during the batching process without interruptions. The experiments did not have specific requirements regarding storage, as the only operations happening were reading the dataset once at the beginning and periodically writing results to a text file. Using the standard workspace disks was sufficient in this regard.

[1] https://schema.org/
[2] https://commoncrawl.org/
[3] http://webdatacommons.org/largescaleproductcorpus/v2/

Around the same time as these experiments were performed, the Transformer [13] architecture and especially BERT [3] were strongly impacting the NLP community and researchers from other fields also started applying this new type of neural architecture to their problems. In our research [9] presented at the DI2KG workshop at VLDB2020, we experimented with intermediately training and fine-tuning these Transformer models for the task of product matching, showing that these models were able to exceed the results of Deepmatcher on the WDC LSPC datasets by a minimum of 2% F1 using the larger training sets up to a maximum of 20% F1 for the smallest training set size. These results show the training data efficiency of these models and the pre-training / fine-tuning paradigm when applied to the product matching task.

In our most recent work [8] presented at VLDB2021 we continued this line of experiments with BERT-based models and investigated the question of improving matching results by not only treating the product matching task as a binary classification but additionally requiring the model to predict the identifiers of the distinct products which form a pair in the form of an additional multi-class objective. This can be achieved by training the BERT model for multiple tasks at once using a combined loss function. This multi-task training can be more beneficial for each task if both tasks are similar than training only for a single task. In this paper we showed that this multi-task training using the JointBERT architecture can improve on the performance of a BERT model trained via a single binary task by 1-5% F1 reaching up to 98.5% F1 if enough training data is available for both tasks and the considered set of products, which is usually the case in the multi-source setting of product matching between online-stores. A more fine-grained analysis on a set of matching challenges revealed that JointBERT improves on BERT on all challenges regarding products seen during training and is especially good at detecting matches and non-matches in noisy product pairs missing words or containing typos.

3 Ongoing Research on Table Representation Learning

An open research question that currently gets a lot of attention in the data integration community is whether the performance on Transformers for data integration tasks can be improved further by pre-training them using large corpora of structured tables in addition to less structured textual input [2, 4, 12, 14]. The models that are pre-trained on the table corpora are afterwards fine-tuned for specific integration tasks such as annotating tables with entities, types and relations from Knowledge bases, or matching entities and columns across different tables. Our current research focuses on hierarchical Transformer models for table-related tasks as well as pre-training such Transformers on large table corpora. As we need to embed complete tables instead of only pairs of product offers, the computational requirements, especially with regards to VRAM have increased significantly. We are now also able to leverage multi-gpu training to parallelize the training procedure. The previously used machines with K80 GPUs are no longer suitable for this, so we are currently making use of the 4 to 8 GPU machines of the bwUniCluster 2.0. The available NVIDIA V100 GPUs are significantly more capable than the older K80 and the available VRAM of 32GB is just enough to fit very small batches of tables. Most recently, we were able to pre-train a complex hierarchical table transformer on a set of $\sim$570K tables from Wikipedia for 100 epochs in 8 days and 6 hours using parallel training on one machine with 8 V100 GPUs. Training models at this scale is clearly impossible using our local compute infrastructure.

4 Conclusions

Training high-performance deep learning models is quite data and resource intensive but has been shown to achieve state-of-the-art results on many tasks including product matching. This is especially true for textual data, where we can constantly reach F1 values above 90% given enough training examples. With the advent of Transformer models and the trend of training models with an increasingly large amount of parameters - currently numbering in the billions for the largest models - conducting research in this area requires access to state-of-the-art hardware which is often not available in the required quantities at university level. The bwHPC initiative offers access to such hardware and has enabled us to do research using models that were previously only usable for large companies or research institutions.

References

[1] C. Bizer, A. Primpeli, and R. Peeters. Using the Semantic Web as a Source of Training Data. *Datenbank-Spektrum*, 19(2):127–135, 2019.

[2] X. Deng, H. Sun, A. Lees, Y. Wu, and C. Yu. TURL: Table understanding through representation learning. *Proceedings of the VLDB Endowment*, 14(3):307–319, Nov. 2020.

[3] 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*, pages 4171–4186, June 2019.

[4] H. Iida, D. Thai, V. Manjunatha, and M. Iyyer. TABBIE: Pretrained Representations of Tabular Data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3446–3456, Online, June 2021. Association for Computational Linguistics.

[5] P. Konda, S. Das, P. Suganthan G. C., A. Doan, A. Ardalan, et al. Magellan: Toward Building Entity Matching Management Systems. *Proceedings of the VLDB Endowment*, 9(12):1197–1208, 2016.

[6] Y. Li, J. Li, Y. Suhara, A. Doan, and W.-C. Tan. Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment*, 14(1):50–60, 2020.

[7] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, et al. Deep Learning for Entity Matching: A Design Space Exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34, 2018.

[8] R. Peeters and C. Bizer. Dual-objective fine-tuning of BERT for entity matching. *PVLDB*, 14(10):1913–1921, 2021.

[9] R. Peeters, C. Bizer, and G. Glavaš. Intermediate training of BERT for product matching. In *CEUR Workshop Proceedings*, volume 2726, pages 2–6, 2020.

[10] R. Peeters, A. Primpeli, B. Wichtlhuber, and C. Bizer. Using schema.org Annotations for Training and Maintaining Product Matchers. In *Proceedings of the 10th International Conference on Web Intelligence, Mining and Semantics*, pages 195–204, June 2020.

[11] A. Primpeli, R. Peeters, and C. Bizer. The WDC Training Dataset and Gold Standard for Large-Scale Product Matching. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 381–386, May 2019.

[12] N. Tang, J. Fan, F. Li, J. Tu, X. Du, G. Li, S. Madden, and M. Ouzzani. RPT: Relational pre-trained transformer is almost all you need towards democratizing data preparation. *Proceedings of the VLDB Endowment*, 14(8):1254–1261, Apr. 2021.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, et al. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, Dec. 2017.

[14] D. Wang, P. Shiralkar, C. Lockard, B. Huang, X. L. Dong, and M. Jiang. TCN: Table Convolutional Network for Web Table Interpretation. In *Proceedings of the Web Conference 2021*, WWW '21, pages 4020–4032, New York, NY, USA, Apr. 2021. Association for Computing Machinery.