Matching Products with Deep NLP Models

Leonidas Akritidis
School of Science and Technology
International Hellenic University
Thermi, Thessaloniki, Greece
Email: lakritidis@ihu.gr

Panayiotis Bozanis
School of Science and Technology
International Hellenic University
Thermi, Thessaloniki, Greece
Email: pbozanis@ihu.gr

***Abstract*—Following the explosive data growth that is presently taking place on the Web, the eCommerce industry has evolved towards enterprises and services that collect product-oriented data from multiple external sources. Since the majority of these sources are usually uncontrolled and independent of each other, they provide their information in a diverse manner, rendering the identification of products a difficult task. The problem becomes particularly challenging by considering the native sparseness and high dimensionality of text, the specific peculiarities of product data, the large data volume and the dynamic nature of the involved applications. Despite the uncontested significance of the problem, there is a lack of works in the relevant literature that confront it by taking into account all the aforementioned challenges. In this paper, we summarize these challenges and provide some useful insights on how they can be effectively tackled. We also present several components of a preliminary system that is being developed to accurately and efficiently identify product entities in diverse data originating from multiple external sources.**

***Index Terms*—product matching, entity matching, record linkage, NLP, BERT**

I. Introduction

The problem of precisely recognizing the product identity from a descriptive excerpt of text is of crucial importance for numerous eCommerce applications. For example, the large-scale product comparison platforms require effective and scalable solutions to allow their users aggregate hundreds of offers from various merchants. Similarly, on-line auction houses that accept offers by individual owners must be able to match an offer with an existing product entity. Even small-sized applications managed by local retailers require similar mechanisms to enrich their electronic catalogs with data feeds coming from different wholesalers.

The significance of the problem has attracted the attention of a significant number of researchers. Earlier works employed simple string similarity (or distance) metrics, such as the cosine and edit distance measures [1]. More recent approaches compute the similarity between two strings by taking into consideration their semantics [2]. However, string similarity measures are not suitable for the problem of product matching for two reasons: First, a high (low) similarity value between two products does not necessarily infer that these products are identical (different) [3], [4]. Frequently, highly similar titles refer to different products and vice versa. Second, their computation is expensive, despite some works proposed strategies for their acceleration [5].

To address the problem of sparseness, several researchers suggested the utilization of external sources of information like search engines and social networks [3], [4]. The most serious drawback of these approaches is that communicating and retrieving information from such sources is a costly procedure. In addition, a portion of these sources does not provide unlimited access, and usage restrictions may hold. In [6] a system for matching product titles was proposed. However, this solution depends on the existence of key elements in the titles, such as product codes and manufacturer names.

In a previous line of research we introduced UPM, a multi-step method that confronts the problem by adopting a clustering approach [7], [8]. After identifying important parts in the title of an offer (e.g., model name, attribute values, measurement units), UPM generates low dimensional vector embeddings by constructing all possible word combinations. Then, the embeddings are appropriately scored and the offers sharing common highly-scored embeddings are placed together in a product cluster. The output is further improved by applying post-processing filters, ensuring that offers of different categories or of different sources cannot coexist within the same cluster. Although UPM achieves high matching quality and it can handle offers of brand new products (i.e., without any matches), it is not appropriate for dynamic systems.

During the past few years, the introduction of the Transformer model yielded huge improvements in natural language classification [9]. BERT (Bidirectional Encoder Representations from Transformers) is considered as one of the most effective, Transformer-based language representation models [10]. However, the original BERT is not suitable for the problem of similarity matching that we examine here, due to its large computational overhead [11].

Several researchers have proposed modifications that confront the problem more efficiently. In particular, Sentence-BERT employs siamese and triplet networks on a pre-trained BERT model to derive semantically meaningful sentence embeddings [11]. In [12], eComBERT adopted the logic of Sentence-BERT for matching product offers. Moreover, JointBERT combines binary matching and multi-class classification, forcing the model to predict the entity identifier in addition to the match/non-match decision [13].

The huge success of the Transformer-based models in hard NLP tasks renders them an attractive approach in the product matching domain. In addition, the dynamic nature of the

eCommerce systems and the large volume of the involved data pose significant challenges that require special treatment. In this paper, we describe these challenges, we provide the necessary background knowledge and we propose a partitioning architecture for improving the efficiency of this task.

II. Challenges

During this study, we encountered multiple factors that render the problem of product identification particularly challenging. We summarize them in the following list:

- *Text Sparseness:* Consisting of only a few terms, a product title is considered as a form of short text. As such, it exhibits a significant degree of sparseness that blurs the semantic similarity with other titles. It derives from various factors, such as the absence of important words from the titles, the usage of different words to describe the same entity, the inclusion of irrelevant terms, the large vocabulary sizes and so on.
- *Latent similarity:* A product title is not simply another form of short text. Unlike natural language text, highly similar titles may refer to different products, and, vice versa, diverse titles may represent the same product [3], [4]. Hence, the over-simplistic usage of similarity measures or similarity-based algorithms may lead to inaccurate results and poor matching quality.
- *High dimensionality and representation:* The traditional Bag-Of-Words (BOW) text representations generate prohibitively long vectors that may easily overwhelm a classifier, even at moderate scales. The aforementioned BERT-based models, like Sentence-BERT, address this issue by representing a text sequence with fix-sized low-dimensional vectors. However, several issues still remain:
  - *Text cleaning* is of crucial importance. How do we treat the irrelevant words in an offer (e.g., *"free shipping"*, *"discount of x%"*, etc.)?
  - *Title enrichment* is another point of research. Should we inject specific "words of interest" (like categories, technical specifications, etc.) in a product title? Are there any important words that are missing from an offer title? How can we handle the abbreviations?
- *Large data volumes:* The warehouse of a typical medium-sized enterprise stores tens, or hundreds of thousands of products. On the other hand, large-scale auction houses and product comparison platforms are 1-3 orders of magnitude larger. Each individual product is a candidate class for an incoming offer. Training and using classifiers directly in datasets of such scales is clearly infeasible with commodity hardware. A robust solution must be scalable and include sophisticated mechanisms to efficiently sustain these tremendous workloads.
- *High data velocity:* New offers may arrive in rates that exceed one million per day. Updating the offer database within reasonable times is a challenging task. This includes insertions of new products, deletions of the obsolete ones, and linkage with the product entities that pre-exist in the system's database.

III. Background

We consider the universe of all products P that consists of product entities $p \in P$ with three properties: a title t_p, a category c_p and a set of attributes A_p. Therefore, p can be expressed as a triplet of the form (t_p, c_p, A_p). The title t_p is always known and it is unique for every p in the collection. The category c_p is also always available, but it depends on the adopted taxonomy C. The set of attributes A_p may include technical specifications (accompanied by their respective measurement units), dimensions, materials, colors, and so on. It may be unknown, or partially known; but even if it is fully known, data quality problems frequently arise, such as completeness, reliability, consistency and integrity [14].

We call a system that maintains a database with P products as a *Product Management System (PMS)*. As mentioned earlier, examples of PMSs include on-line auction houses, product comparison platforms, retailers, and so on. Apart from its underlying product database, a PMS receives product offers O, originating from external data sources; e.g., independent users, merchants, APIs, third parties, and so on.

Similarly to the records of P, an offer $o \in O$ is always described by a title t_o, whereas it may also possess a category c_o and a set of attributes A_o. Notice that, since o originates from an external uncontrolled source, c_o and A_o may be blank or unreliable. In addition, c_o may be incompatible to the taxonomy C that is adopted by the PMS. It is also common that some of the attributes of o are embedded into its title t_o.

The automatic product matching mechanism of a PMS must effectively perform the following operations:

1) Analyze the incoming offers and, for each offer $o \in O$, identify the product $p \in P$ that it represents. Then, create a match between o and the corresponding product p, and potentially update the availability, price, and attributes of p.
2) In several cases, o may refer to a product that is not present in P. This usually occurs when o concerns a new release, or when it promotes very old unsupported products. In such cases, a new record p' must be inserted to P, and a match between o and p' must be generated.
3) Optional: Strictly speaking, categorization is not a requirement of the product matching problem. Nonetheless, in this work we consider it extremely important for two reasons: i) it facilitates category-based navigation and enhances search robustness in a PMS, and ii) it improves both matching quality and execution speed as described in Subsection IV-A.
4) Optional: Several real-world PMS applications require that the products of P that match no incoming offers to be deleted, deactivated, or marked as unavailable.

IV. System Overview

This section describes the architecture of our multi-stage approach for processing the incoming set of offers O. An indicative block diagram is illustrated in Figure 1. To perform the four aforementioned operations, an offer $o \in O$ passes through the three following stages:

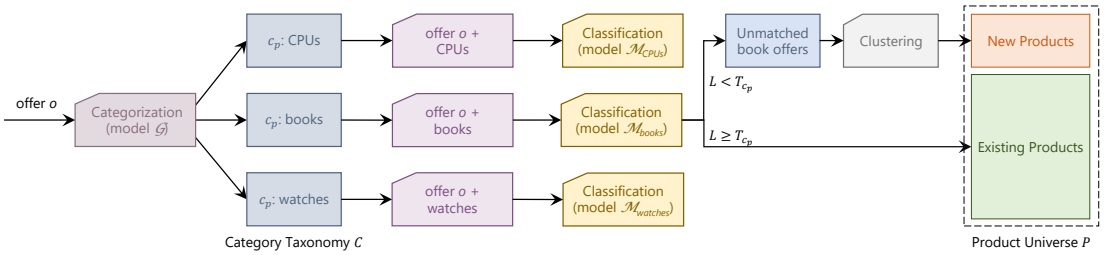

Fig. 1. The execution flow of the product matching procedure

- *Categorization:* Given that a PMS operates a pre-existing product taxonomy C, a deep learning model $\mathcal{G}$ assigns a category $c \in C$ to o.
- *Classification:* This stage matches o with a single product $p \in P$. It is based on a model set $\mathcal{M}$ that includes $|C|$ classifiers, one per category. The classification is performed by a picking a model $\mathcal{M}_c \in \mathcal{M}$ that has been trained with the products belonging to the category c of o. Apart from the matching product p, $\mathcal{M}_c$ also outputs an uncertainty value L that is indicative of the classification trustworthiness.
- *Clustering of the unmatched offers:* In case L is smaller than a category-specific threshold T_c, we assume that o matches none of the products $p \in P$. Now this stage creates $|C|$ pools of such unmatched offers, also one per category $c \in C$. Then, it applies a clustering algorithm to create new clusters and place the similar offers there. The cluster labels are subsequently utilized to create new products and append them to P.

More details on these stages are described in Subsections IV-A, IV-B, and IV-C.

A. Product Categorization

Determining the category c of an incoming offer o is of particular importance for numerous reasons. First, a categorized set of product entities P improves user experience and allows extended search capabilities. In the context of product matching, we leverage categorization to limit the pool of candidate matches to a subset $P_c \in P = \{p \mid c_p = c\}$ that includes only the products belonging to same category c as the offer o. This approach has several beneficial implications that are described in details in the next subsection.

The categorization process begins by applying a number of preprocessing filters to the offer titles. The filters perform case folding (i.e., conversion of all characters to lower case) and punctuation removal. Several symbols, like dots and dashes, are significant for recognizing the identify of a product. Hence, dots are left intact in the offer titles, whereas the dashes are replaced by space characters. An additional cleaning filter is also applied for noise (irrelevant words) removal. In this Work-In-Progress paper we utilize a simple dictionary-based strategy. However, the Denoising Autoencoders offer an attractive alternative that will be examined during our future work.

Regarding short-text representation, the literature includes a broad variety of state-of-the-art techniques. However, as mentioned earlier, the well-established approaches that generate word embeddings are not suitable in our case, because it is the entire sequence that must be vectorized and not just a single word. The methods that simply average the embeddings of the included words to derive the short text embedding do not perform well, especially when the embeddings have been constructed by bidirectional Transformers [11].

For this reason, a set of BERT modifications have been proposed in the literature with the aim of generating high quality sentence embeddings. Sentence-BERT generates low-dimensional embeddings of the input text sequences by fine-tuning BERT with siamese and triplet networks [11]. In this article, we draw inspiration from Sentence-BERT: we fine-tune the standard BERT model (110 million parameters) by using the output state of a BERT model trained for the masked language modelling (MLM) task.

Regarding the classifier $\mathcal{G}$, we employ a deep learning model that is fed with the constructed short text embeddings. $\mathcal{G}$ includes a Birectional Long-Short Term Memory (BiLSTM) layer with 768 units utilizing the Rectified Linear Unit (ReLU) for activation. A Dropout layer with a rate equal to 0.2 is attached to the output of the BiLSTM units with the aim of avoiding overfitting. The output of $\mathcal{G}$ is given by the softmax function of a fully connected layer. The model is trained with Adam that optimizes the categorical cross-entropy loss function in 50 epochs with and batches of 128 examples.

B. Product Matching

After the determination of the category c of an offer o, another mechanism attempts to identify a product p that best matches o. The process utilizes a set $\mathcal{M}$ of $|C|$ classifiers. Each classifier $\mathcal{M}_c \in \mathcal{M}$ has been trained on the subset of the products that belong to the respective category c. Product matching is then performed by using the model $\mathcal{M}_c$ to classify o into one of the products of $P_c \subseteq P$.

This strategy avoids querying the entire database P, leading to two beneficial outcomes: i) it limits the possibility of false matches, since it searches only among products of the same category as the offer o, and ii) the entire matching task is much faster because it is deployed on a subset of the original data. In the illustrated example, o is categorized as a "book". Hence, we can safely discard all "CPUs" and "Watches" and work with the "books" category only.

$\mathcal{M}_c$ may be any classifier; in this Work-In-Progress article we employ another BiLSTM architecture, similar to the one that was used in the categorization process. Hence, the model comprises a BiLSTM layer, followed by a Dropout layer to

prevent overfitting. This configuration is attached to a fully connected layer, and the entire model is trained in batches of 32 and in 50 epochs. The categorical cross-entropy loss function is minimized by utilizing the Adam optimizer.

By properly selecting the activation function of the output layer of $\mathcal{M}_c$ (e.g. softmax, logistic, etc.), the generated output can be interpreted as a probability. We denote this probability with L and we set a threshold T_c for each category $c \in C$. This threshold determines the matching strength between o and p, or else, whether the classification (match) of o in p is reliable or not. We adopt the multiple threshold approach because our experiments have revealed that for different product categories, the classifier $\mathcal{M}_c$ must output probabilities of different magnitude in order to safely match o with a product.

Now, if $L \geq T_c$, the output of $\mathcal{M}_c$ is sufficiently large, so it is considered reliable. We create a match between o and p and we optionally perform several actions with p, like update its price, availability, stock, etc. On the other hand, if $L < T_c$, then the output of $\mathcal{M}_c$ is weak and probably represents an incorrect match. In this case, the output of the classifier is ignored and the methodology of the next subsection is applied.

C. Creating New Products

The offers that have been unsuccessfully classified into one of the existing products of P are grouped according to their category in a set of $|C|$ buckets B. The grouping is done in such a manner, that each bucket $b \in B$ stores unmatched offers belonging to the same category (see Figure 1).

Since these offers match no product entities, we consider that they represent new releases that have not been encountered before. Thus, P must be expanded to include them. To achieve this goal, we apply a text clustering algorithm to each bucket b, with the aim of grouping together the offers that represent identical products. Given that the vector embeddings of the offer title have already been created during the categorization phase, a simple algorithm like k-Means or Agglomerative clustering can be applied directly by using these representations. Alternatively, UPM from [7], [8] is also applicable.

After the clustering phase is completed, a new product for each generated cluster is appended to P. The titles of the new products derive by utilizing either the cluster label, or the title of the representative (i.e., center, or clustroid) element.

V. Preliminary Conclusions and Future Work

In this Work-In-Progress paper we introduced a deep learning approach to the problem of product matching in e-Commerce systems. The proposed method can effectively match an incoming offer to a product entity, whereas it is also capable of handling offers of new products that do not match any of the existing entries. On its first stage, the vector representation of an offer title is obtained by fine-tuning BERT with the output of another BERT model trained with the masked language modeling. Then, the offer embeddings are fed into a classifier that identifies the category of the offer.

On the second stage, the system employs a set of classifiers and selects the one that has been trained on the products belonging to the same category as the offer. In case the output probability is adequately large, then a match between the input offer and the predicted product class is formed. Otherwise, new products are created by applying a clustering algorithm that is also category-agnostic. This category-based approach is designed to improve both matching quality and efficiency.

Our current work is mainly oriented towards the proper selection of the categorization and classification models and the design of their architecture. Additional research is also conducted towards the identification of category-based aspects that will further improve the effectiveness of our method.

Acknowledgment

This research is co-financed by Greece and the European Union (European Social Fund-SF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning 2014-2020" in the context of the project "Support for International Actions of the International Hellenic University" (MIS 5154651).

References

[1] W. H. Gomaa, A. A. Fahmy *et al.*, "A survey of text similarity approaches," *International Journal of Computer Applications*, vol. 68, no. 13, pp. 13–18, 2013.

[2] W. Hua, Z. Wang, H. Wang, K. Zheng, and X. Zhou, "Short text understanding through lexical-semantic analysis," in *Proceedings of the 31st IEEE Int'l Conference on Data Engineering*, 2015, pp. 495–506.

[3] V. Gopalakrishnan, S. P. Iyengar, A. Madaan, R. Rastogi, and S. Sengamedu, "Matching product titles using web-based enrichment," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, 2012, pp. 605–614.

[4] N. Londhe, V. Gopalakrishnan, A. Zhang, H. Q. Ngo, and R. Srihari, "Matching titles with cross title web-search enrichment and community detection," *Proceedings of the VLDB Endowment*, vol. 7, no. 12, pp. 1167–1178, 2014.

[5] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang, "Efficient similarity joins for near-duplicate detection," *ACM Transactions on Database Systems*, vol. 36, no. 3, pp. 1–41, 2011.

[6] H. Köpcke, A. Thor, S. Thomas, and E. Rahm, "Tailoring entity resolution for matching product offers," in *Proceedings of the 15th Int'l Conference on Extending Database Technology*, 2012, pp. 545–550.

[7] L. Akritidis and P. Bozanis, "Effective unsupervised matching of product titles with k-combinations and permutations," in *Proceedings of the 14th IEEE International Conference on Innovations in Intelligent Systems and Applications*, 2018, pp. 1–10.

[8] L. Akritidis, A. Fevgas, P. Bozanis, and C. Makris, "A self-verifying clustering approach to unsupervised matching of product titles," *Artificial Intelligence Review*, vol. 53, no. 7, pp. 4777–4820, 2020.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[11] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," *arXiv preprint 1908.10084*, 2019.

[12] J. Tracz, P. I. Wójcik, K. Jasinska-Kobus, R. Belluzzo, R. Mroczkowski, and I. Gawlik, "BERT-based similarity learning for product matching," in *Proceedings of Workshop on Natural Language Processing in E-Commerce*, 2020, pp. 66–75.

[13] R. Peeters and C. Bizer, "Dual-objective fine-tuning of bert for entity matching," *Proceedings of the VLDB Endowment*, vol. 14, pp. 1913–1921, 2021.

[14] L. Akritidis, A. Fevgas, and P. Bozanis, "Effective products categorization with importance scores and morphological analysis of the titles," in *Proceedings of the 30th International Conference on Tools with Artificial Intelligence*, 2018, pp. 213–220.