MWPD-2020 Task 1: Product Matching Based on Deep Entity Matching Frameworks

Cheng Fu[1,3], Tianshu Wang[1,3], Hao Nie[1,3], Xianpei Han[1,2] and Le Sun[1,2]

[1] Chinese Information Processing Laboratory, Institute of Software, Chinese Academy of Sciences
[2] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
[3] University of Chinese Academy of Sciences
{fucheng, tianshu2020, niehao2016, xianpei, sunle}@iscas.ac.cn

Abstract. This paper describes our product matching system developed for Semantic Web Challenge on Mining HTML-embedded Product Data 2020 (Task 1). Product matching is the task of identifying product offers deriving from different websites that refer to the same real-world product, which is a typical scenario of entity matching (EM). In our system, we implement four state-of-the-art deep learning-based entity matching models and integrate their results to get the final product matching predictions. Competition results show that, our system obtains promising performance on this task.

Keywords: Product Matching, Entity Matching, Semantic Web.

1 Introduction

Product matching is the task of identifying product offers deriving from different websites that refer to the same real-world product, which is a typical scenario of entity matching (EM). In this task, product matching is handled as a binary classification problem: given two product offers decide if they describe the same product (matching) or not (non-matching). It is critical for many downstream applications such as product knowledge graph construction, product search, product recommendation etc.
Entity matching has been extensively studied since the 1950s [7], thus a variety of methods for solving the EM problem have been proposed [8, 9]. The existing EM approaches can be roughly divided into two categories: rule-based, and machine learning-based. Rule-based approaches resolve entity record pairs using matching rules given by domain experts [10] or automatically learned from labeled examples [11, 12, 13]. Machine learning (ML)-based approaches usually treat entity resolution as a classification problem [14]. Traditional ML approaches include SVM-based models [15], Markov logic-based methods [16s], active learning-based solutions [17], etc. Recently, some deep learning-base methods were also proposed for EM. One main advantage of such approaches is that they can better capture semantic similarity between textual attributes, and can efficiently reduce human cost in EM pipeline [1, 2, 5, 6, 18, 19].

In our system, we implement four state-of-the-art deep learning-based entity matching models (MPM, Seq2SeqMatcher, HierMatcher and DITTO), and integrate results output by them to get the final product matching predictions.

2 System Overview

As shown in Fig. 1, our system mainly consists of three pipeline modules, which are respectively pre-processing module, entity matching module and post-processing module. Pre-processing module is to normalize all attribute values and complete product entity information by extracting some new information. Entity matching module is to predicate whether the two offers refer to the same product using end-to-end entity matching frameworks. Post-processing module is to refine the matching results produced by the entity matching module via some heuristic rules.

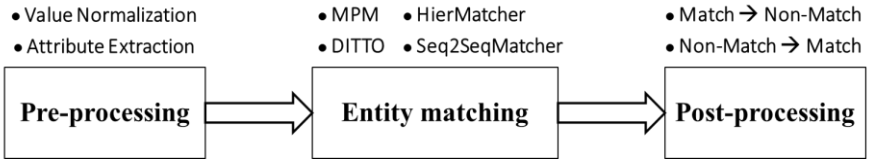

Fig. 1. Overview of our proposed system.

2.1 Pre-processing

Value Normalization. For inputs of different subsequent modules, we use different pre- normalization strategies. In most cases, given a raw textual value, we remove non-alphanumeric characters and stopwords using NLTK, and then lowercase all tokens it contains. But when preparing data for model extraction, we don't do the lowercase operation and keep some special non-alphanumeric characters (such as "-" and "/") which can be considered as important model extraction features.

Attribute Extraction. To complete product entity information, we attempt to extract two key types of attribute values for each product offer: brand and model. In raw datasets, brand is an existing attribute, but its value coverage rate is low (eg. 57.4% in the official dataset). Model is a new attribute that can usually help in entity Matching. For the brand attribute, we use a vocabulary-based extraction approach. Specifically, we first built a brand vocabulary based on the WDC Product Data Corpus [20] (Computers & Accessories domain in its English version). Then use an exact matching strategy to get brand value for each product offer via the vocabulary. When constructing the brand vocabulary, we use the following existing attributes in the corpus: "*brand*", "*brand name*", "*merk*", "*manufacturer*" and "*marca*". For the model attribute, we use two strategies. The first one is vocabulary-based, which is similar to the extraction of brand. For model vocabulary construction, we use "*Part Number*" and

"*SKU*" attributes in the WDC Product Data Corpus [20] (Computers & Accessories domain in its English version). The second one is pattern-based, in which we use about twenty Python regular expressions to filter model candidates from the existing title and description attributes. After get a candidate set of model values for each offer, we use TF-IDF to choose the final one.

2.2 Entity Matching

In this module, we use four recently proposed EM models (MPM, Seq2SeqMatcher, HierMatcher and DITTO) to predict whether two offers refer the same product. Then use a voting mechanism to integrate their prediction results for each product offer pair. Specifically, in Round 1 of this competition, we integrate results from MPM, Seq2SeqMatcher and HierMatcher. In Round 2 of this competition, we integrate results from MPM, HierMatcher and DITTO. Detailed introduction of the four models used in our system are as follows.

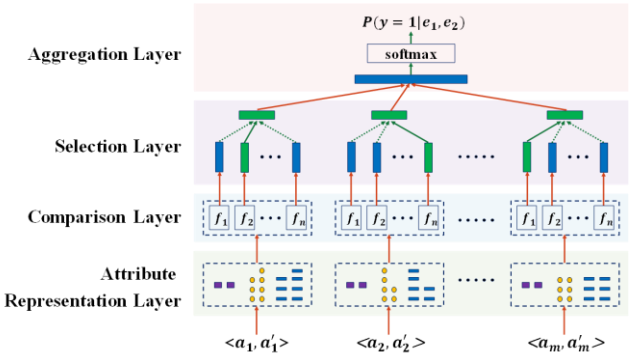

Fig. 2. Framework of the MPM model used in our system.

MPM [1]. MPM is an end-to-end multi-perspective matching model for entity resolution, which can adaptively select the optimal similarity measures for heterogenous attributes, and jointly learn and select similarity measures in an end-to-end way. As shown in Fig. 2, it uses a "*compare-select-aggregate*" neural framework, which first compares aligned attribute values in multiple perspectives using different similarity measures, then adaptively selects the optimal similarity measure for each attribute by designing a gate mechanism, finally aggregates the comparison results of the selected similarity measures from all attributes to make EM decision.

In our system, we use 4 attributes of each product offer as input： *brand, model, title, price*. For each attribute, we use eight similarity measures of three types (the same as in [1]) to get multi-pespective comparison results, then adaptively select the optimal one. We use the pretrained FastText 300-dimensional word embedding [3] for the two DL based similarity measures: *rnn_sim* and *hybrid_sim*. Hidden size of each GRU layer is set 256.

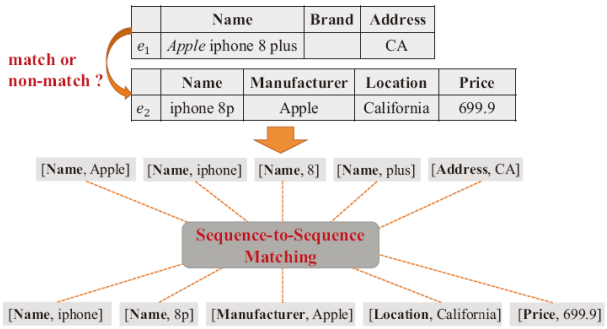

Fig. 3. Framework of the Seq2SeqMatcher model used in our system.

Seq2SeqMatcher [5]. Seq2SeqMatcher is a deep learning-based entity matching model aiming to effectively solve the heterogeneous and dirty cases by modeling EM as a token-level sequence-to-sequence matching task. Fig. 3 shows its architecture, in which each record is linearized as a token sequence, and each token is a pair of the form <*attribute, word*>. From the figure we can see that: 1) it compares records in token-level instead of attribute level, where no attribute alignment knowledge is needed, therefore can naturally solve the heterogeneous schemas; and 2) tokens can be compared across attributes and contribution of each token to the final EM decision is automatically learned, therefore the dirty cases can be effectively solved.

For this model, we use 4 attributes of each product offer as input: *brand, model, title, price*. And we use the same word embedding and parameter settings as the originally paper [5] in our system.

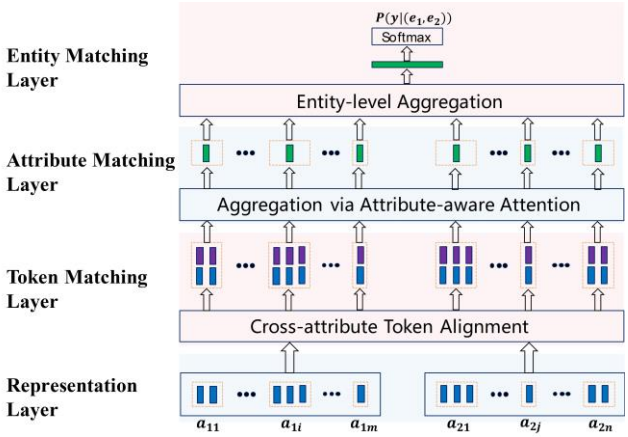

Fig. 4. Framework of the HierMatcher model used in our system.

HierMatcher [6]. HierMatcher is a hierarchical matching network also designed to resolve heterogeneous and dirty entity matching problems. As shown in Fig. 4, it can jointly model entity matching at three levels (token, attribute, and entity) in a unified neural framework. At the token level, it constructs a cross-attribute token alignment module. By selecting comparison objects for all tokens across all attributes, it can effectively address the schema heterogeneity and the misplaced-type dirty data problems. At the attribute level, it uses an attribute-aware attention mechanism, which can learn to identify important information for different attributes, therefore can effectively resolve the redundant-type and noisy-type dirty data problems. Furthermore, by obtaining matching evidence level by level, i.e., aggregating comparison results from token level to attribute level, and then to entity level, it can fully take advantage of hierarchical structure information of entities.

For this model, we use 4 attributes of each product offer as input: *brand*, *model*, *title*, *price*. And we use the same word embedding and parameter settings as the originally paper [6] in our system.

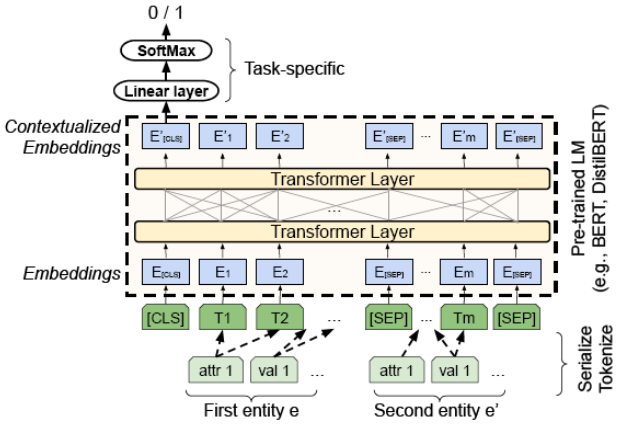

Fig. 5. Framework of the DITTO model used in our system.

DITTO [2]. DITTO is a novel entity matching system based on pretrained Transformer-based language models. It fine-tunes and casts EM as a sequence-pair classification problem to leverage such models with a simple architecture. As shown in Fig. 5, given two entities, DITTO serializes them as one sequence and feeds it to the model as input. The model consists of (1) token embeddings and Transformer layers from a pre-trained language model (e.g., BERT) and (2) task-specific layers (linear followed by softmax). Conceptually, the [CLS] token "summarizes" all the contextual information needed for matching as a contextualized embedding vector $E'_{[CLS]}$ which the task-specific layers take as input for classification.

In our system, we fine-tune our EM model on an uncased 12-layer DistilBERT [4] pre-trained model. We fix the learning rate to be 1e-5 and the max sequence length to

be 512. For each product offer, we use the following 4 attributes for matchings: *brand, model, title, price*.

2.3 Post-processing

Post-processing module is used to correct some obviously error of prediction results output by the entity matching module using heuristic rules. For example, those pairs with exactly the same title, but be predicted to be non-matched, and those pairs with different brands but be predicted to be matched.

3 Data

In this competition, compared with test set, official released training and validation sets are much easier. Furthermore, they do not cover the product offers contained in the test set well. Therefore, for the competition, we construct an extended training dataset and an extended validation dataset, which are harder and have better coverage of the product offers in the test set. The size of our extended training set is 138,461, which contains 68,461 pairs from official training set and 70,000 new hard pairs. The size of our extended validation set is 6,100, which consists of 1,100 pairs from official validation set and 5,000 new hard pairs. Additional hard instances (product offer pairs) mentioned before are obtained by the following two steps.

Step 1: Initial candidate set construction. We first sample 10,000 clusters from subset of the WDC Product Data Corpus (English version), each product of which belongs to the Computers & Accessories category. Then we construct a large dataset containing 914,878 product offer pairs using the same strategy as the one used to construct official training dataset.

Step 2: Hard instance selection. Given each offer pair from the initial candidate set, we then use Jaccard similarity of offer titles to select hard samples. Specifically, for each positive sample, if title similarity of its offers is less than 0.4, we consider it as a hard one. For each negative sample, if title similarity of its offers is more than 0.6, we consider it as a hard one. Finally, we randomly sample corresponding numbers of samples for the extended training set and validation set described before. Positives/negatives ratio in the sampled hard pairs is 3:7.

4 Evaluation

Table 1 reports the results of our systems in the two rounds of this competition. ISCAS-ICIP is our system in Round 1 integrating results of MPM, Seq2SeqMatcher and HierMatcher. ISCAS-ICIP (R2) is our system in Round 2 integrating results of MPM, HierMatcher and DITTO. From this table we can see that, our systems significantly outperform the baseline system built on DeepMatcher. Specifically, ISCAS-

Table 1. Results of our systems in the competition.

	Systems	Precision	Recall	F1
Base models	MPM	80.43	82.66	81.53
	Seq2SeqMatcher	82.26	81.17	81.71
	HierMatcher	82.61	82.07	82.34
	DITTO	84.48	83.17	83.82
Our systems	ISCAS-ICIP	83.89	81.33	82.59
	ISCAS-ICIP (R2)	**85.77**	**84.95**	**85.36**
Baseline	DeepMatcher	70.89	74.67	72.73

ICIP and ISCAS-ICIP (R2) respectively achieve 9.8 and 12.6 F1 score improvement on the test set. It demonstrates that, our systems can achieve promising performances for the product matching tasks. Besides, both of our systems in the two rounds outperform all base models integrated by them, which demonstrates the effectiveness of the integration strategy used in our systems.

5 Conclusion

This paper describes our product matching system developed for Semantic Web Challenge on Mining HTML-embedded Product Data 2020 (Task 1). In our system, we implement four state-of-the-art deep learning-based entity matching models (MPM, Seq2SeqMatcher, HierMatcher and DITTO), and integrate results from multiple models from them to get the final product matching predictions. Competition results show that, our system obtains promising performance on this task.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants no. U1936207 and 61772505, the National Key Research and Development Program of China under Grant No.2017YFB1002104, and Beijing Academy of Artificial Intelligence (BAAI2019QN0502).

References

1. Cheng Fu, Xianpei Han, Le Sun, Bo Chen, Wei Zhang, Suhui Wu, and Hao Kong. End-to-end multiperspective matching for entity resolution. In Proceedings of the IJCAI, pages 4961–4967. AAAI Press, 2019.
2. Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. Deep Entity Matching with Pre-Trained Language Models[J]. arXiv, 2020: arXiv: 2004.00584. Conference Name: ACM Woodstock conference.
3. Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov. Enriching Word Vectors with Subword Information. CoRR abs/1607.04606 (2016).

4. V. Sanh, L. Debut, J. Chaumond, and T. Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In Proc. EMC2 ’19, 2019.s
5. Hao Nie, Xianpei Han, Ben He, Le Sun,Bo Chen, Wei Zhang, Suhui Wu, and Hao Kong. Deep sequence-to-sequence entity matching for heterogeneous entity resolution. In Proceedings of CIKM, pages 629–638, 2019
6. Cheng Fu, Xianpei Han, Jiaming He, Le Sun. Hierarchical Matching Network for Heterogeneous Entity Resolution. In Proceedings of the IJCAI,2020.
7. Howard B Newcombe, James M Kennedy, SJ Axford, and Allison P James. Automatic linkage of vital records. Science, 130(3381):954–959, 1959.
8. AnHai Doan and Alon Y Halevy. Semantic integration research in the database community: A brief survey. AI magazine, 26(1):83–83, 2005.Conference Location:El Paso, Texas USA
9. Nick Koudas, Sunita Sarawagi, and Divesh Srivastava. Record linkage: similarity measures and algorithms. In Proceedings of the ACM SIGMOD, pages 802–803. ACM, 2006.ISBN:978-1-4503-0000-0/18/06
10. Mauricio A Hern´andez and Salvatore J Stolfo. The merge/purge problem for large databases. In ACM Sigmod Record, volume 24, pages 127–138. ACM, 1995.
11. Surajit Chaudhuri, Bee-Chung Chen, Venkatesh Ganti, and Raghav Kaushik. Exampledriven design of efficient record matching queries. In PVLDB, pages 327–338. VLDB Endowment, 2007.
12. Jiannan Wang, Guoliang Li, Jeffrey Xu Yu, and Jianhua Feng. Entity matching: How similar is similar. PVLDB, 4(10):622–633, 2011.
13. Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quian´e-Ruiz, Armando Solar-Lezama, and Nan Tang. Synthesizing entity matching rules by examples. PVLDB, 11(2):189–202, 2017.
14. Ivan P Fellegi and Alan B Sunter. A theory for record linkage. Journal of the American Statistical Association, 64(328):1183–1210, 1969.
15. Mikhail Bilenko and Raymond J Mooney. Adaptive duplicate detection using learnable string similarity measures. In Proceedings of the ACM SIGKDD, pages 39–48. ACM, 2003.
16. Parag Singla and Pedro Domingos. Entity resolution with markov logic. In ProcSunita Sarawagi and
17. Anuradha Bhamidipaty. Interactive deduplication using active learning. In Proceedings of the ACM SIGKDD, pages 269–278. ACM, 2002.eedings of the ICDM, pages 572–582. IEEE, 2006.
18. Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. Distributed representations of tuples for entity resolution. PVLDB, 11(11):1454–1467, 2018.
19. Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. Deep learning for entity matching: A design space exploration. In Proceedings of the ICDM, pages 19–34. ACM, 2018.
20. Primpeli, A., Peeters, R., & Bizer, C.: The WDC training dataset and gold standard for large-scale product matching. In: Companion Proceedings of the 2019 World Wide Web Conference. pp. 381-386 ACM (2019).