Computing and Informatics, Vol. 43, 2024, 1397–1415, doi: 10.31577/cai_2024_6_1397

ENHANCING SEMANTIC WEB ENTITY MATCHING PROCESS USING TRANSFORMER NEURAL NETWORKS AND PRE-TRAINED LANGUAGE MODELS

Mourad Jabrane, Abdelfattah Toulaoui, Imad Hafidi

Sultan Moulay Slimane University
Laboratory of Process Engineering, Computer Science and Mathematics
Bd Beni Amir, BP 77
Khouribga, Morocco
e-mail: mourad.jabrane@usms.ac.ma

Abstract. Entity matching (EM) is a critical yet complex component of data cleaning and integration. Recent advancements in EM have predominantly been driven by deep learning (DL) methods. These methods primarily enhance data accuracy within structured data that adheres to a high-quality and well-defined schema. However, these schema-centric DL strategies struggle with the semantic web's linked data, which tends to be voluminous, semi-structured, diverse, and often noisy. To tackle this, we introduce a novel approach that is loosely schema-aware and leverages cutting-edge developments in DL, specifically transformer neural networks and pre-trained language models. We evaluated our approach on six datasets, including two tabular and four RDF datasets from the semantic web. The findings demonstrate the effectiveness of our model in managing the complexities of noisy and varied data.

Keywords: Entity matching, record linkage, linked data, deep learning, transformer neural networks

1 INTRODUCTION

Entity matching (EM), also called duplicate detection or record linkage, was first defined by Newcombe in 1959 [1]. The challenge is identifying effectively the duplicates that correspond to the identical real-world entity. In the context of Big

Data, EM has gradually shifted toward noisy, large, semi-structured, and highly heterogeneous data. Consequently, EM methods keep progressing and with each EM problem that is solved with sufficient performance, more challenging problems arise. So, more advanced solutions will always be needed. Current state of the art when it comes to EM in textual and tabular data uses methods based on deep learning, due-to those methods offering a more powerful framework for handling EM problem. The latest of these methods make use of the power of pre-trained language and transformers models. These models bring a level of natural language processing, so that the framework can use the meaning of the text to achieve the task, rather than just semantic similarity. In the previous generation, the domain of EM was confined to structured data, whereas in recent times, it has extended to semi-structured and unstructured data, i.e., linked data (web data), predominantly created by users. The current landscape of data exhibits a high degree of schema noise and heterogeneity, which are characterized by loose schema bindings with uncertain semantics, thereby posing unprecedented challenges to data matching. For this type of data the current state-of-the-art EM methods rarely make use of machine learning, let alone deep learning. The present scenario is dominated by the prevalence of rule-based techniques and iterative algorithms, which are hampered by inadequate scalability, substantially low time efficiency, and lengthy execution times even for datasets comprising merely a few thousand entities.

For this reason, in this paper we propose an innovative, loosely schema-aware approach based on recent advancements in deep learning to perform EM on linked data. Particularly, transformer neural networks and pre-trained language models.

The summary of the contributions made by our work is presented as follows: Firstly, we compared the performance of five deep learning models, including DeepMatcher, HierMatcher, CorDEL, and DITTO on three pairs of datasets from the author and product domains, namely Amazon-Google (dirty), Amazon-Walmart (dirty), and Abt-Buy (textual). Secondly, we propose an innovative, loosely schema-aware approach based on the DITTO model to handle linked data effectively. Finally, we conducted an extensive evaluation of the proposed approach on six datasets, which included two tabular datasets, namely Amazon-Google and Amazon-Walmart, and four RDF datasets, namely SPIMBench, Person1, Person2, and Anatomy.

This paper is organized as follows: Section 2 discusses the related work in the areas of deep learning for EM. Section 3 explains our schema-aware approach to handle linked data. Section 4 presents the experimental evaluation of our approach and comparison to related work. Section 5 presents the findings of the study. Section 6 provides a discussion about the results. Section 7 summarizes the key findings and proposed some possible directions for future work.

2 RELATED WORK

2.1 DeepMatcher

One of the most important contributions to EM came with DeepMatcher [2]. DeepMatcher uses pre-trained word embedding to calculate the vector representations of tokens in the entity, then aggregates those representations for each attribute to find the vector representation of the entire attribute. Finally, the vector representations are aggregated one last time to find entity representations, which are concatenated and passed to the decision network which is composed of a two-layer MLP with residual connections and a softmax function at the end (Figure 1). The authors have open-sourced their work as the *deepmatcher* Python package. The authors suggest using GloVE or FastText for the word embedding, but FastText performed better in their testing. They also suggest using simple averaging, RNNs, or Attention layers for vector aggregation, with performance varying between data sets.

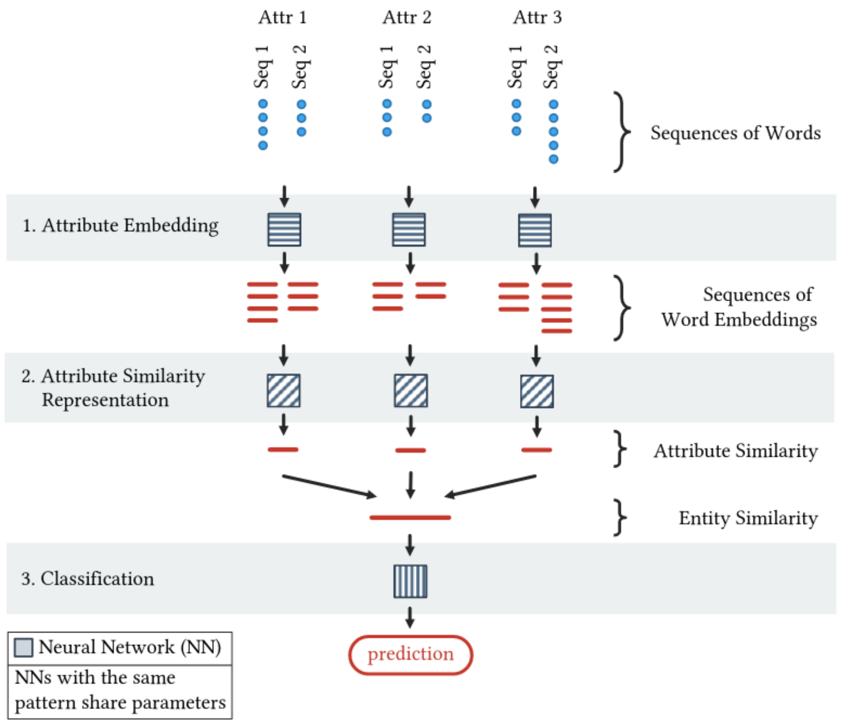

Figure 1. The general structure of DeepMatcher model

DeepMatcher was improved upon by DeepMatcher+ [3], which is used as a benchmark for entity matching.

2.2 Auto-EM

Auto-EM [4] uses multiple models for entity matching: an attribute type detection model, type-specific matchers, a general matcher, and a table-level matcher. The authors also pre-trained 49 type-specific matchers using very large KBs so the models can be fine-tuned for new types and specific tasks.

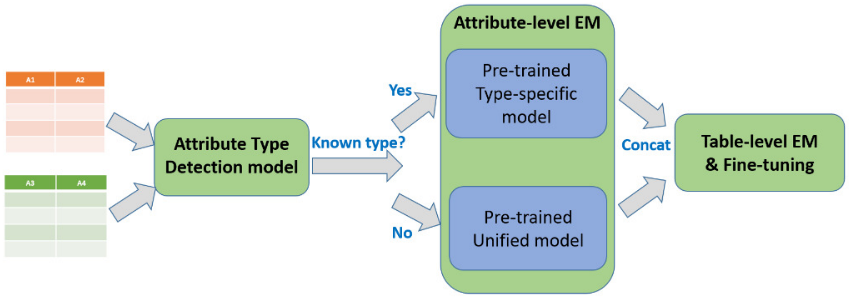
A1
A2
A3
A4
Attribute Type Detection model
Known type?
Yes
No
Attribute-level EM
Pre-trained Type-specific model
Pre-trained Unified model
Concat
Table-level EM & Fine-tuning

Figure 2. The general structure for Auto-EM model

The flow as described in Figure 2: Firstly, the attribute type detection model is used to predict the type of each attribute from the table, these predictions are then used to choose the correct model for type-specific matchers. The type-specific matchers take two inputs of the same type (Person names, addresses, phone numbers, movies, books, organizations, etc. . .) and output a similarity score. For attributes that are of unknown types, a general matching model is used. Finally, the scores are concatenated, and the table-level matcher is used to calculate the final matching score.

All the type-specific matchers and the general matcher have the exact same architecture (Figure 3), and the difference between them stems from their training data. The matchers use a hierarchical architecture. A character level BiGRU is used to determine how the characters within the words are connected, and an character level attention layer is used to align the characters between the attributes, and then the representations of characters within the words are aggregated via a weighted sum. Finally, the same process is used again on the word-level instead of on a character-level to calculate the attribute representation. The final attribute representations are concatenated and passed to an MLP to retrieve the final score.

This method offers a great deal of flexibility for entity matching. The paper also provides a way to pre-train the classifier and matchers to reduce the training time and data required for new tasks. However, it can only process tabular data, and the schemata need to be aligned in order to use it.

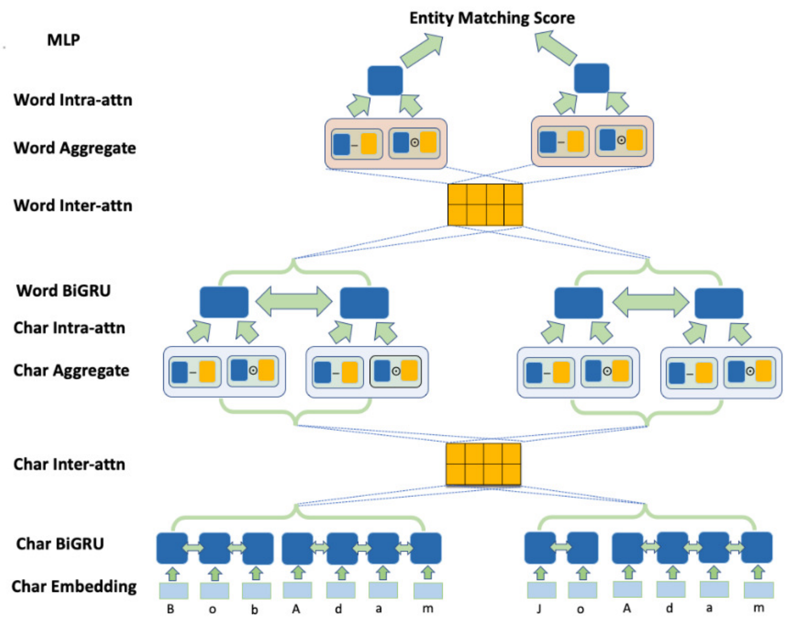

Figure 3. The hirerarchical model used for attribute-level matching

2.3 HierMatcher

Researchers here [5] conceived a hierarchical matching model based on attention layers (Figure 4). A hierarchical model passes the information through different sub-models with different context information, each one extracts an aspect of the information relative to the context.

The model starts by taking the attributes in the form of tokens, and a vector representation for each token is calculated using a BiGRU RNN. The token representations are aligned with the similar tokens in the opposite entity. With H_2 the matrix where each column is the representation of a token in the second entity, and h_{1i} the representation of the target token repeated to match the dimension of H_2, the similarity is calculated as follows:

$$\begin{aligned} C_{1i} &= h_{1i} - H_2, \\ G_{1i} &= \mathit{HighwayNet}(C_{1i}), \\ v_{1i} &= \mathit{softmax}(wG_{1i} + b), \end{aligned}$$

where *HighwayNet* is a two layer MLP with residual connections, and w and b are learnable parameters. The final output of this layer is the column of C_{1i} whose index has the highest value in v_{1i}.

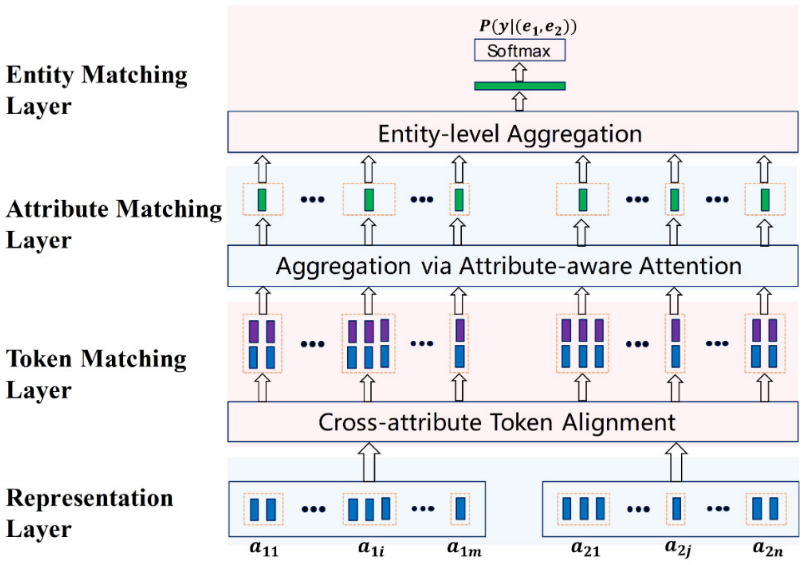

Figure 4. The architecture of the Hierarchical Matcher proposed in [5]

Next, the token representations for each attribute are aggregated via an attention layer to get a single vector representation of each attribute, and the same is done again to aggregate the attribute vectors for each entity. Finally, the two vectors are concatenated and fed into a two-layer MLP with a ReLU activation for the first one, and a Softmax activation to make the final decision.

This model is very flexible, and can handle heterogeneous data. It could also theoretically be extended to handle data that is unstructured or does not have a fixed schema. However, since it uses attention layers, it may require large amounts of data to reach optimality. Another possible problem is that the model may be too simple to handle more complex problems.

2.4 CorDEL

CorDEL [6] was proposed as a model to do entity matching on systems without using too much in term of resources. It uses a simple process to separate tokens into tokens unique to each entity (Figure 5), and tokens common between them in a LIM (Local Interaction Module). Given t_{1j} and t_{2j} the tokens have an attribute j in the first and second entities, respectively.

$$s_j = t_{1j} \cap t_{2j},$$

$$u_{1j} = t_{1j} \setminus s,$$

$$u_{2j} = t_{2j} \setminus s.$$

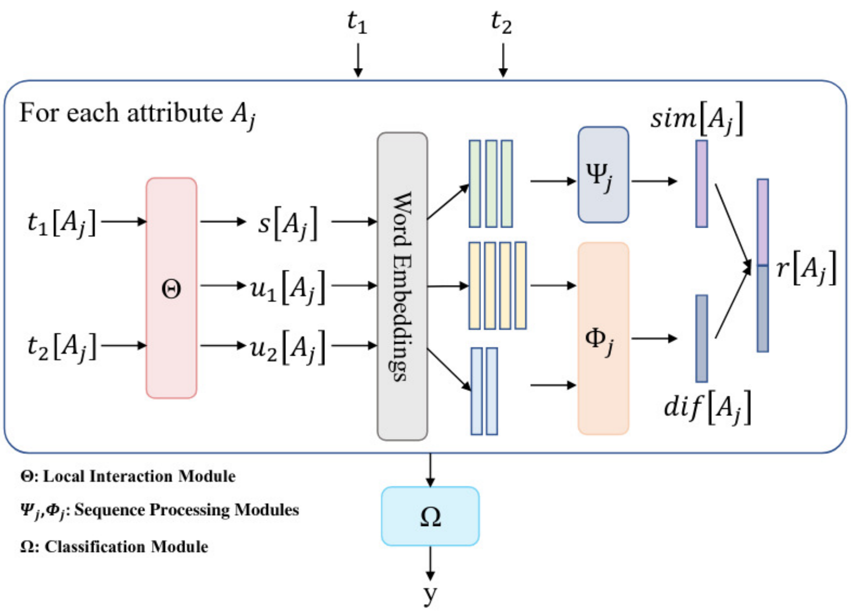

Figure 5. The architecture of the CorDEL model

After the LIM, CorDEL passes the tokens to a pre-trained embedding layer, which calculates the representative vectors for each token. Both sets of tokens (s_j and $u_{1j} \cup u_{2j}$) are passed to their own sequence processing model (ψ_j and ϕ_j, respectively). The intuition being that ψ_j can calculate the evidence of similarity between the two attributes, while ϕ_j can calculate the evidence of dissimilarity. Finally, the two evidence vectors are concatenated and passed to the model Ω, which takes all the outputs from the attribute level comparisons and outputs a single similarity score y.

One thing to take into consideration is that the model needs a hyper-parameter in the form of an aggregation for the sequence processing model. The paper suggests using either the sum of vectors or an RNN, depending on the target system.

The model is very simple yet powerful, and can handle heterogeneous data very well. However, it needs aligned schemata to function, which is not always optimal. The model also cannot handle unstructured data.

2.5 DITTO

DITTO [7] is an entity matching model that makes use of pre-trained language models to process textual data. The paper was published simultaneously with another paper [8] that had the same idea. The model uses a pre-trained language model (specifically BERT or its derivatives), and converts the entities to match into a text sequence. The text sequence is then fed into the language model, and the first output vector is taken as the input of a binary classification layer.

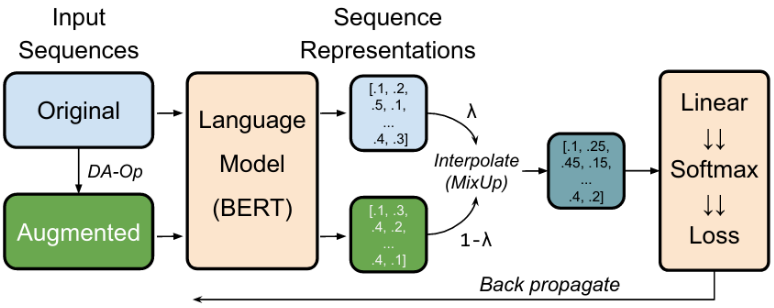

Figure 6. The architecture for the DITTO model with data augmentation

The authors [7] suggest serializing the data into the format:

```
[START]
[COL]col11[VAL]value11[COL]col12[VAL]value12...
[SEP]
[COL]col21[VAL]value21[COL]col22[VAL]value22....
[END]
```

Which gives the model the clear structure of the attributes in the entities. This is desirable if the attribute names encode useful information for the entity as a whole.

The authors also defined a process for data augmentation that could be optionally used to help the model generalize. Augmented data is processed the same way during training of the model, and the result vector is interpolated with the original vector before the classification layer.

The model makes use of the latest technologies in NLP, and it offers a serialization that can make use of the attribute names as part of the decision process. One of this model's strong points is its use of pre-trained model, which can cut down on training time and required data set size.

2.6 Comparison

2.6.1 Training Time

The DL-based approaches consistently outperform a number of recognized benchmark datasets [9]. Despite their great performance, they are very inefficient with time, taking hours of training even for datasets containing a few thousand entities [8], and require specialized hardware to run large models efficiently.

The relevant approaches attempt to reduce the complexity of the learned models, which are highly dependent on the parameters which must be refined during training. Still, the majority of DL models contain a substantial number of parameters: over 22 million for the hybrid model of Deep-Matcher, over 700 000 for DeepMatcher-SIF and CorDel-Sum, and over 50 000 for CorDel-Attention [6]. Additionally, special attention should be taken to constructing stable, resilient, and hence dependable models, such that different training runs produce negligible changes in performance [6].

DITTO is the only model that cuts down on training time and required labeled data set size by using pre-trained language models.

2.6.2 Performance

For performance evaluation of the methods described above, we use three pairs of datasets from the author and the product domains. The three datasets cover two types of EM problems (textual and dirty).

- Amazon-Google (dirty): e-commerce datasets include product entities, described with the following four common attributes: product name, product description, manufacturer, and price. The provided ground truth contains 8 442 (7 142 negative and 1 300 positive) correspondences between Amazon and Google product entities.
- Amazon-Walmart (dirty): contains product information along the Amazon and Walmart electronics product catalogues. The common attributes are: title, category, brand, modelno, price. The provided ground truth contains 10 242 (9 280 negative and 962 positive) correspondences between the Amazon and Walmart product entities.
- Abt-Buy (textual): benchmark e-commerce datasets, contain textual attributes. A ground truth of 7 164 (6 067 negative and 1 097 positive) correspondences between product entities of the two datasets. The common attributes are: product name, product description and product price.

We take the best case results in the case where the model has multiple variations. Table 1 presents the results of the models performance in terms of F1 score.

	Amazon-Google	Amazon-Walmart	Abt-Buy
DeepMatcher+	70.7	53.8	63.8
HierMatcher	74.9	68.5	–
CorDEL	70.2	51.2	64.9
DITTO	75.58	85.69	89.33

Table 1. The performance of the methods on three datasets

We noticed that DITTO has the highest performance across the board. This is due to the fact that its base model (BERT) has a good generalization of natural language.

3 PROPOSED APPROACH

This section will discuss the proposed new serialization scheme to handle the semantic web's linked data. We will also discuss the datasets used for testing. This was done on the model with varying amounts of data to get an idea on the absolute minimum required to get an acceptable score. This section also introduces a few possible modifications to improve and stabilize the performance of the model in low data circumstances.

3.1 The Base Model

DITTO offers a powerful and efficient solution for entity matching tasks that can help improve the quality of data integration and cleaning processes by:

- Improved accuracy: By using pre-trained language models to generate embedding for entities, DITTO can improve the accuracy of entity matching tasks compared to traditional methods. Pre-trained language models have been shown to be effective at capturing semantic and contextual information in text, which can help identify matching entities even when there are variations in spelling, word order, or phrasing.
- Time and cost savings: By automating the entity matching process, DITTO can save time and reduce the costs associated with manual data cleaning and integration. This can be particularly valuable for organizations that deal with large and diverse datasets that require frequent updates and maintenance.
- Scalability: DITTO is a scalable solution that can handle large volumes of data and can be deployed in distributed environments. It can also be integrated with existing data pipelines and workflows to support real-time data processing and analytics.
- Flexibility: DITTO is a flexible framework that can be customized with different pre-trained language models, embedding methods, and matching algorithms.

Evaluating the degree of semantic similarity between two profiles in linked data is a difficult problem in the field of entity matching. Improved performance was achieved by approaches based on language models such as T5 [10], XLNet [11], ERNIE 2.0 [12], ALBERT [13], RoBERTa [14], TinyBERT [15], however these approaches either require high computational resources (T5, XLNet, ERNIE, ALBERT, RoBERTa) or have low performance (TinyBERT). Hence, we opted to use DistilBERT as the base language model for our approach. DistilBERT has been distilled using the Knowledge Distillation method [16] to compress the knowledge contained in a large model into a smaller deep learning model, by having the original model act as a teacher for the new one. DistilBERT was trained using the BERT model as a teacher, and provides a similar degree of language generalization to the original BERT model, while having a much smaller number of parameters (and therefore lower computational cost). Hence, DistilBERT provides smaller download size and faster training and inference times (practically provides 97 % of the performance of BERT, while only taking 40 % of the size and being 60 % faster [17]).

To make this model compatible with our target, we change the serialization scheme for input items to handle the semantic web's linked data.

3.2 Serialization Scheme

The proposed serialization scheme is described below:

- Put the token *[PROP]* and the suffix of each predicate URI as the column name, followed by the token *[VAL]* and value of the property for each literal property.
- For each resource, put the token *[NEI]* apply the same serialization to the properties of each neighbor and concatenate it to the serialization.
- Optionally, prepend the suffix of the URI ID of the item, this is beneficial if the URI contains useful information.

3.2.1 Example

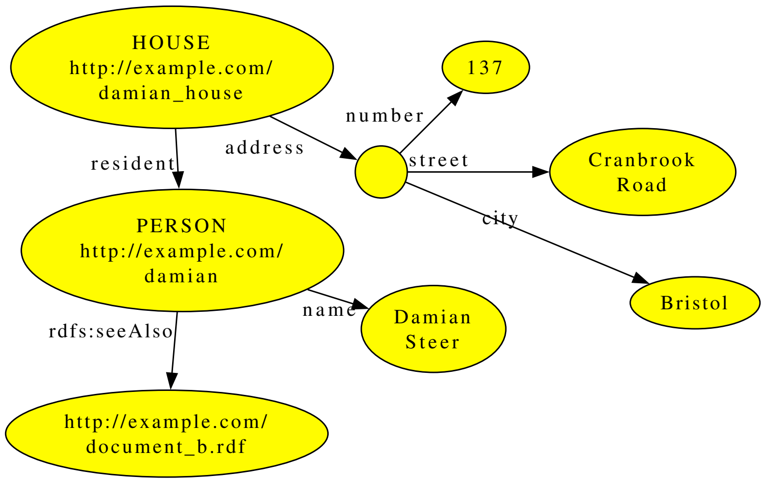

Figure 7. A simple linked data graph (RDF item) as an example

Take the linked data graph in figure 7 (RDF Format), it will be serialized as follows:

```
damian_house
[NEI] address
[VAL] address1
```

```
 [PROP] number [VAL] 137
 [PROP] street [VAL] Cranbrook Road
 [PROP] city [VAL] Bristol
[NEI] resident
 [VAL] damian
 [PROP] name [VAL] Damian Steer
```

This serialization takes into account how some resources will have pertinent information nested under several levels of relations, like the address in the previous example.

To account for variations in the structure of datasets, we propose a three-level this serialization scheme:

Shallow: Only serialize the literal properties of the item in question, ignoring all of its neighbors;

Medium: Serialize the literal properties of the item and the URIs of its direct neighbors;

Deep: Serialize the literal properties of the item along with the literal properties of its direct neighbors.

3.3 Training Modifications

3.3.1 Label Smoothing

Label Smoothing is a technique to reduce over-fitting in DL. It adds a small degree of randomness to each target label y, by applying the following simple transformation:

$$y' = (1 - a)y + a \times r.$$

With r sampled uniformly from $[0, 1]$, and a a parameter chosen by the user, a is commonly chosen to be a small number, with the interval $[0.05, 0.2]$ being the most commonly used interval to not disturb the values of y too much.

Then, y' is used as the target label instead of y. This ensures that the model does not fit the expected results exactly during training, because that could lead to over-fitting.

In this model, the value of $a = 0.1$ gave the best results.

3.3.2 Linear Learning Rate

During training, the model was given a number of 'Warm-up Steps', those are steps where the learning rate goes from 0 to the Target learning rate linearly (Figure 8). After that, the learning rate is slowly decreased linearly over the period of training.

Changing the learning rate like this gives the parameters of the model the chance to fall into their respective optimal ranges before the learning rate reaches a large value. If we start with the learning rate at its starting value directly, it could lead to some of the parameters overshooting.

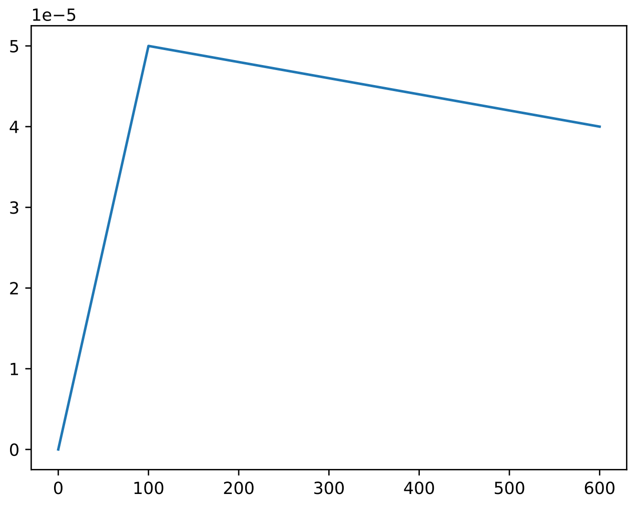

Figure 8. Warm-up steps of 100

3.3.3 Model Initialization

To ensure that the model learns optimally, the weights of the classifier layer were initialized using Kaiming normal. Kaiming normal [18] is a method for sampling weights from a distribution that is conceived to ensure that, given inputs sampled from a normal distribution, the layer will produce outputs that follow a normal distribution $\mathcal{N}(0, 1)$. Kaiming normal has been tested in practice and was proven to improve the convergence time of deep learning models. The weights are sampled in Kaiming normal following a normal distribution with a mean of 0 and a standard deviation of $\sqrt{\frac{2}{D}}$ – i.e. $\mathcal{N}\left(0, \sqrt{\frac{2}{D}}\right)$ – with D being the dimension of the inputs.

4 EXPERIMENTAL EVALUATION

The model was trained on Kaggle, with GPU acceleration. At the time of writing, Kaggle offers a *Tesla P100* with 16 GB of GPU RAM. As previously mentioned, the code was written in Python 3, and was run on Kaggle's Jupyter notebook interface.

The implementation uses PyTorch version 1.11.0, which is the long-term-support version as of the time of writing. Additionally, to access the library of pre-trained language models, the code makes use of HugginFace Transformers version 4.18.0 which is used to download and run pre-trained models that were shared by other base language modeler people and groups.

4.1 Training

As proposed in the original paper [7], the model was trained for 20 epochs. The learning rate was set to 2×10^{-5} which is the recommended rate for transfer learning using language transformers. The training was done using a batch size of 32.

Depending on the size of the dataset, the training took 2 to 3 hours using all fractions of the datasets.

4.2 Datasets

Six datasets were used to test the model, two tabular datasets and four RDF datasets:

- Amazon-Google [19] and Amazon-Walmart, which had been discussed previously.
- SPIMBench is part of AOEI-2021, it provides two graphs that represent creative works. The first graph contains the original works, and the second one is the same graph, but with schema and content transformations applied. This synthetic dataset can be seen as a test to investigate if the model can handle corrupted data.
- Person1 and Person2 are two RDF datasets from AOEI-2010. The datasets contain entity matching tasks, they each contain two graphs representing people and need to match the people in the two graphs. Each dataset has two RDF files representing the two graphs, and a third RDF file representing the ground truths for the matching task.
- Anatomy is a dataset from AOEI-2021. It represents a real world task of matching the anatomy of mice to the human anatomy represented as a knowledge graph.

The negative pairs for training and testing were generated using a blocking algorithm, specifically token blocking, which groups together items that have the same tokens (continuous alphanumeric strings) in their properties or the properties of their neighbors. This was done to avoid having too many trivial negative pairs in the training process. For our purposes, we take the pairs that exist in the smallest blocks while discarding the large blocks which have very common, and therefore less useful, tokens such as "www" which exists in all URIs. For each dataset we take negative pairs to aim for at least a 1 to 7 ratio of positive to negative pairs. We use a train/test split of 70 %/30 %.

For all datasets, the model was tested using 10 %, 25 %, 50 %, 80 %, and 100 % of the training data, respectively. This will allow us to test the model's ability to adapt to applications with limited labeled data, and its ability to achieve an acceptable results in those cases.

Dataset	**# Items**	**# Positives**	**Type**
Amazon-Google	11 460	1 167	Tabular
Amazon-Walmart	10 242	962	Tabular
SPIMBench	2 885	299	RDF
Person1	3 000	500	RDF
Person2	3 200	400	RDF
Anatomy	6 048	1 516	RDF

Table 2. Properties of the datasets used in testing

5 RESULTS

The model yielded the results shown in Table 3. As expected, the larger the training data, the better the model performs. These results are different from the results stated in the paper, this could be due to a combination of factors, one possible factor is that the training data does not have the same split as the data used in the original article. Part of the difference could be partially attributed to the modifications made to the training process.

	10 %	**25 %**	**50 %**	**80 %**	**100 %**
Amazon-Google	43.72	63.56	71.03	71.73	73.83
Amazon-Walmart	33.33	50.98	69.05	91.33	91.67

Table 3. F-scores of the model applied to tabular data

With RDF data, we get the results shown in Table 4. The *level* indicates the serialization level we chose for each dataset, it could be shallow, medium or deep.

Our approach was trained using different fractions of the input datasets. The model was tested with 10 %, 25 %, 50 %, 80 %, and 100 % of the training data to test the model's behavior with different amounts of training data.

	Datasets			
	Person 1	**Person 2**	**SPIMBench**	**Anatomy**
Level	deep	deep	medium	deep
MinoanER	1.0	0.8888	1.0	0.0705
DT10 %	1.0	0.9296	0.9176	0.7690
DT25 %	1.0	0.9515	0.9663	0.8444
DT50 %	1.0	0.9770	0.9780	0.9017
DT80 %	1.0	1.0	0.9945	0.9079
DT100 %	1.0	1.0	1.0	0.9259

Table 4. F_1-scores of the model applied to RDF data compared to MinoanER

The plot in Figure 9 shows the results for each dataset.

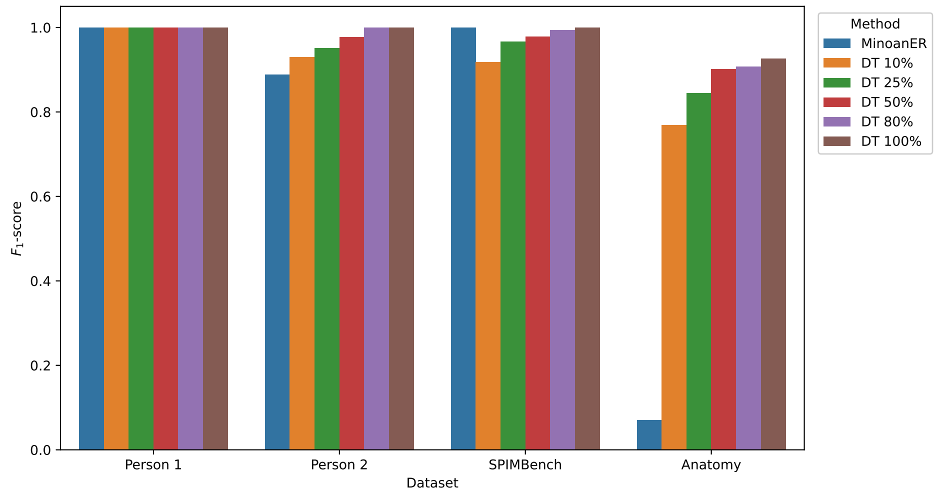

Figure 9. A plot comparing the performance of our method to MinoanER

6 DISCUSSION

We observe an increase in the performance of our method with more training data, which is advantageous for datasets that already have a number of labeled pairs, as that would yield better results.

For simple datasets, MinoanER performs reasonably well, yielding a perfect score on Person 1 and SPIMBench, but slightly less than our method on Person 2. Where MinoanER struggles is on the Anatomy dataset, which is a much more difficult task.

Our approach consistently performs the same as MinoanER or better when provided enough training data, the performance gap is especially clear on the Anatomy dataset which represents a real world case that requires a certain level of knowledge of anatomical structures to perform the matching. This could prove useful for other real world cases with domain-specific language models.

7 CONCLUSION AND FUTURE WORK

Entity matching is one of the most researched areas of data science. It can vastly improve data quality for many applications and create more useful data. In recent years, many approaches have been proposed for entity matching on linked data, but not many of them utilise the power of deep learning and pretrained language models.

In this work, deep learning has proven to be a powerful tool for handling linked data, exceeding state-of-the-art performance on entity matching. And it is still especially underutilized in this type of data, which has proven to be a great resource for knowledge storage and distribution.

In future works it may be beneficial to make use of deep learning to either improve the current state-of-the-art performance in other applications relating to linked data and entity resolution. In which case, entity matching could also serve as the entry point for using deep learning to reason on the basis of linked data, which is still under-explored at the moment. Another possible direction is using Active Learning to further improve the model and minimize its data requirements while improving its performance.

REFERENCES

[1] Newcombe, H. B.—Kennedy, J. M.—Axford, S. J.—James, A. P.: Automatic Linkage of Vital Records: Computers Can Be Used to Extract "Follow-Up" Statistics of Families from Files of Routine Records. Science, Vol. 130, 1959, No. 3381, pp. 954–959, doi: 10.1126/science.130.3381.954.

[2] Mudgal, S.—Li, H.—Rekatsinas, T.—Doan, A.—Park, Y.—Krishnan, G.—Deep, R.—Arcaute, E.—Raghavendra, V.: Deep Learning for Entity Matching: A Design Space Exploration. Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18), ACM, 2018, pp. 19–34, doi: 10.1145/3183713.3196926.

[3] Kasai, J.—Qian, K.—Gurajada, S.—Li, Y.—Popa, L.: Low-Resource Deep Entity Resolution with Transfer and Active Learning. In: Korhonen, A., Traum, D., Màrquez, L. (Eds.): Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019). 2019, pp. 5851–5861, doi: 10.18653/v1/P19-1586.

[4] Zhao, C.—He, Y.: Auto-EM: End-to-End Fuzzy Entity-Matching Using Pre-Trained Deep Models and Transfer Learning. The World Wide Web Conference (WWW '19), ACM, 2019, pp. 2413–2424, doi: 10.1145/3308558.3313578.

[5] Saisubramanian, S.—Kamar, E.—Zilberstein, S.: A Multi-Objective Approach to Mitigate Negative Side Effects. In: Bessiere, C. (Ed.): Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20). 2020, pp. 354–361, doi: 10.24963/ijcai.2020/50.

[6] Wang, Z.—Sisman, B.—Wei, H.—Dong, X. L.—Ji, S.: CorDEL: A Contrastive Deep Learning Approach for Entity Linkage. 2020 IEEE International Conference on Data Mining (ICDM), 2020, pp. 1322–1327, doi: 10.1109/ICDM50108.2020.00171.

[7] Li, Y.—Li, J.—Suhara, Y.—Doan, A.—Tan, W. C.: Effective Entity Matching with Transformers. The VLDB Journal, Vol. 32, 2023, No. 6, pp. 1215–1235, doi: 10.1007/s00778-023-00779-z.

[8] Brunner, U.—Stockinger, K.: Entity Matching with Transformer Architectures – A Step Forward in Data Integration. 23rd International Conference on Extending Database Technology (EDBT 2020), 2020, pp. 463–473, doi: 10.21256/zhaw-19637.

[9] Li, Y.—Li, J.—Suhara, Y.—Doan, A.—Tan, W. C.: Deep Entity Matching with Pre-Trained Language Models. Proceedings of the VLDB Endowment, Vol. 14, 2020, No. 1, pp. 50–60, doi: 10.14778/3421424.3421431.

[10] Raffel, C.—Shazeer, N.—Roberts, A.—Lee, K.—Narang, S.—Matena, M.—Zhou, Y.—Li, W.—Liu, P. J.: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research, Vol. 21, 2020, Art. No. 140, `http://jmlr.org/papers/v21/20-074.html`.

[11] Yang, Z.—Dai, Z.—Yang, Y.—Carbonell, J.—Salakhutdinov, R.—Le, Q. V.: XLNet: Generalized Autoregressive Pretraining for Language Understanding. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (Eds.): Advances in Neural Information Processing Systems 32 (NeurIPS 2019). Curran Associates Inc., 2019, pp. 5753–5763, doi: 10.48550/arXiv.1906.08237.

[12] Sun, Y.—Wang, S.—Li, Y.—Feng, S.—Tian, H.—Wu, H.—Wang, H.: ERNIE 2.0: A Continual Pre-Training Framework for Language Understanding. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, No. 5, pp. 8968–8975, doi: 10.1609/aaai.v34i05.6428.

[13] Lan, Z.—Chen, M.—Goodman, S.—Gimpel, K.—Sharma, P.—Soricut, R.: ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations. 8th International Conference on Learning Representations (ICLR 2020), 2019, doi: 10.48550/arXiv.1909.11942.

[14] Liu, Y.—Ott, M.—Goyal, N.—Du, J.—Joshi, M.—Chen, D.—Levy, O.—Lewis, M.—Zettlemoyer, L.—Stoyanov, V.: RoBERTa: A Robustly Optimized BERT Pretraining Approach. CoRR, 2019, doi: 10.48550/arXiv.1907.11692.

[15] Jiao, X.—Yin, Y.—Shang, L.—Jiang, X.—Chen, X.—Li, L.—Wang, F.—Liu, Q.: TinyBERT: Distilling BERT for Natural Language Understanding. CoRR, 2019, doi: 10.48550/arXiv.1909.10351.

[16] Hinton, G.—Vinyals, O.—Dean, J.: Distilling the Knowledge in a Neural Network. CoRR, 2015, doi: 10.48550/arXiv.1503.02531.

[17] Sanh, V.—Debut, L.—Chaumond, J.—Wolf, T.: DistilBERT, A Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. CoRR, 2019, doi: 10.48550/arXiv.1910.01108.

[18] He, K.—Zhang, X.—Ren, S.—Sun, J.: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1026–1034, doi: 10.1109/ICCV.2015.123.

[19] Köpcke, H.—Thor, A.—Rahm, E.: Evaluation of Entity Resolution Approaches on Real-World Match Problems. Proceedings of the VLDB Endowment, Vol. 3, 2010, No. 1-2, pp. 484–493, doi: 10.14778/1920841.1920904.

Mourad Jabrane is a Ph.D. candidate and Adjunct Professor at the National School of Applied Sciences (ENSA), Sultan Moulay Slimane University, Khouribga, Morocco. He earned his State Engineer degree in software engineering from the same institution in 2021. His research is focused on addressing complex problems in the areas of entity resolution, big data, data mining, optimization, and machine learning. His work aims to contribute to both the theoretical and applied advancements in these fields, with a particular emphasis on developing novel solutions to data-centric challenges in real-world applications.

Abdelfattah Toulaoui is a Ph.D. candidate at the National School of Applied Sciences (ENSA), Khouribga, Sultan Moulay Slimane University, Morocco. He earned his Master's degree in Big Data and decision support from the same institution in 2022. His research focuses on cutting-edge areas such as efficient deep learning, computer vision, and language modeling. He is particularly dedicated to exploring innovative approaches that enhance the applicability and efficiency of machine learning models in real-world scenarios. His work aims to bridge the gap between theoretical advancements and practical applications, especially in the development of resource-efficient deep learning techniques.

Imad Hafidi is Professor at the National School of Applied Science (ENSA) in Khouribga, Morocco. He is the Head of the Department of Mathematics and Computer Engineering and serves as the Director of the Laboratory of Process Engineering, Computer Science, and Mathematics (LIPIM) at ENSA, Khouribga. His current research focuses on several cutting-edge areas, including big data, data mining, machine learning, frequent itemsets mining, heuristics, sentiment analysis, entity resolution, anomaly detection, and emotion recognition. His work contributes to advancements in both theoretical and applied aspects of these domains.