RECORD LINKAGE WITH A DEEP LEARNING APPROACH

Simone Albero 552166
Giordano Andreola 546572
Elisa Catena 547007
Maria De Domenico 547210

1 Introduction

Society worldwide is generating more and more data giving rise to the "data deluge" problem. This leads to manage a large amount of duplicate data, so records (from two or more data sources) that refer to the same real-world entity need to be linked [3].
Record linkage is the task of quickly and accurately identifying records corresponding to the same entity from one or more data sources [1]. For example, given two files A and B, it aims to link them using quasi-identifiers such as name, address, date of birth, and other fields [2].
In many cases, datasets do not share a unique identifier, thus the process of linking records needs to be performed by matching their corresponding attributes. This becomes challenging due to issues such as different data formats, language ambiguity and abbreviations.
The application of Machine Learning (ML) offers a promising approach for record linkage. The existing ML-based approaches assume that the data obtained from different sources are structured and represented by overlapping sets of attributes. This is very restrictive in terms of real-world applications, given the increasing number of unstructured data sources. Consequently, using ML methods for record linkage tasks becomes more challenging and it has been limited to structured data only [3].
The record linkage strategy follows a three steps pipeline:

1. blocking: efficiently create small blocks of similar records;
2. pairwise matching: compares all record pairs in a block;
3. clustering: groups sets of records into entities.

In this report, we are going to delve into the pairwise matching step. In particular, we present two solutions: the first using Multilayer Perceptron and the other introducing Autoencoders. Various models and improvements have been created for both approaches.

2 Machine learning based approach for record linkage

In this section, we will present a general approach to record linkage using machine learning, highlighting its primary challenges, as this method has become increasingly prevalent in recent years.
At the state-of-the-art, most of the existing works refer to structured data sources. Formally, for two structured data sources with overlapping attributes, the similarity between any two records can be determined by comparing their corresponding attributes using a similarity measure. Actually, given a pair of records

$$s = (s.f_1, \ldots, s.f_N), \quad t = (t.f_1, \ldots, t.f_N)$$

(where $f_1, \ldots, f_N$ represent the overlapping attributes), a similarity measure m quantifies the similarity between two attributes values of s and t and can be formulated as:

$$m : F_i \times F_i \rightarrow [0, 1]$$

where F_i denotes the domain of attribute f_i. The similarity measure returns a numeric value ranging between 0 and 1, referred to as a similarity value [3].
In machine learning, a statistical model and an algorithm are used to "learn" the optimal set of parameters. This learning algorithm relies on a training data set, which, in the context of record linkage, consists of curated data with labeled true and false matches [4]. So, in this case, the RL task can be then considered as a comparison vector classification problem. Notationally:

$$[S, T, m] \Rightarrow \mathrm{LR} : \vec{V} \rightarrow \{0, 1\}$$

where $\vec{V} = \{m(s_i, t_j) : s_i \in S, t_j \in T\}$ is the set of comparison vectors generated for each record pair from $S \times T$ [3].
Finally, once the optimal parameters are determined from the training data, the predictive model can be applied to unlabeled data to identify new links[4].
The key challenge of ML based approaches to RL is the fact that they cannot handle unstructured and heterogeneous data sources. Moreover, the quality of the training data set is critical; the model is only as good as the data it is trained on. After an initial round of record linkage, a sample of record pairs that are not clearly matches or nonmatches is given to a research assistant who makes the final determination.

3 Unstructured record linkage

In this section we propose four different models, all using Siamese Neural Networks (SNN), to solve the record linkage problem on unstructured documents.

3.1 Siamese Neural Networks (SNN)

Siamese nets were first introduced in the early 1990s by Bromley and LeCun to solve signature verification as an image matching problem. A SNN is composed of two identical networks that take distinct inputs but are joined by a loss function at the top. The function measures distance between the high level representations of the two inputs.
The weights of the two networks are tied so that two similar inputs can not be mapped too far from each by the networks in the future space. The main idea behind SNN is that they can learn useful data descriptors that can be further used to compare between the inputs of the respective subnetworks [5].

3.2 Problem definition

As aforementioned, the major limitation is that the existing machine learning-based approaches can be applied just to structured data. Actually, nowadays data are being obtained in many various formats including structured, semi-structured and unstructured. Consequently, we propose a new solution that includes unstructured data as well.
Consider two datasets of records $S = \{s_1, \ldots, s_n\}$ and $T = \{t_1, \ldots, t_m\}$, where each dataset may contain structured or unstructured records. We further assume that the records across S and T are not represented by the same scheme. For each record from S and T, we use L to denote a pre-trained language model, which can be used to embed the records into numerical vectors of the same dimension.
For unstructured data, a pre-trained language model is applied to vectorize each of the records. The embedding of a record can then be calculated as a combined vector of all attribute embeddings. Here, we address the task of supervised RL, that of leveraging $\{S, T\}$ and a language model L to train a classification model for predicting a pair of records $\{s_i, t_j\}$ as a match or non-match.
Notationally:

$$[S, T, L] \xrightarrow[\text{Learning}]{\text{Supervised}} \text{RL} : S \times T \rightarrow \{0, 1\}$$

The output of RL would be a 0/1 label, where 0 indicates that a pair of records does not refer to the same entity while 1 means that the two records are a match [3].

3.3 A solution with Siamese Multilayer Perceptron

We propose and evaluate two different architectures of SNN using a Siamese Multilayer Perceptron. We will refer to the two architectures as SNN1 and SNN2, respectively.
In both cases, x_1 and x_2 (provided as inputs to the twin networks) represent embeddings of a pair of records. We will discuss in the next section how we calculate embeddings.
Both architectures consist of two identical Multilayer Perceptron (MLP) networks that share the same structure (i.e., the number of layers and neurons) and parameters. Each of the two networks produces a high level feature representation of the input vector, o_1 and o_2, respectively.

3.3.1 Embeddings

Word embedding refers to a set of language modeling and feature learning techniques in natural language processing where words or phrases from the vocabulary are mapped to vectors of real numbers [7].
In [6], they used **word2vec**, a two-layer neural network that processes text by taking in batches of raw textual data, processing them and producing a vector space of several hundred dimensions. Each unique word in the data is assigned a vector positioned close to other vectors if the words share common contexts in the corpus [6].
However, since the embedders are dictionaries with a limited number of elements, they cannot represent words that they do not know.
In fact, we tried to use word2vec, but it is poorly suited to the nature of the dataset of our experiments, as this contains attributes relevant for pairwise matching that are not recognized.
So, we decided to use Bert, which is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers [9].
For all implemented architectures, we used two pre-trained Bert models with different sizes:

- $BERT_{BASE}$: composed of 12 encoders;
- $BERT_{LARGE}$: composed of 24 encoders.

3.3.2 SNN1

Architecture. The architecture of the first proposed model is presented in Figure 1.

With SNN1 the final layer computes the Euclidean distance between the outputs of the two twin networks.
The training goal of the model is to learn a new representation of the records that minimise the Euclidean distance between matching records and maximise it for non-matching records.

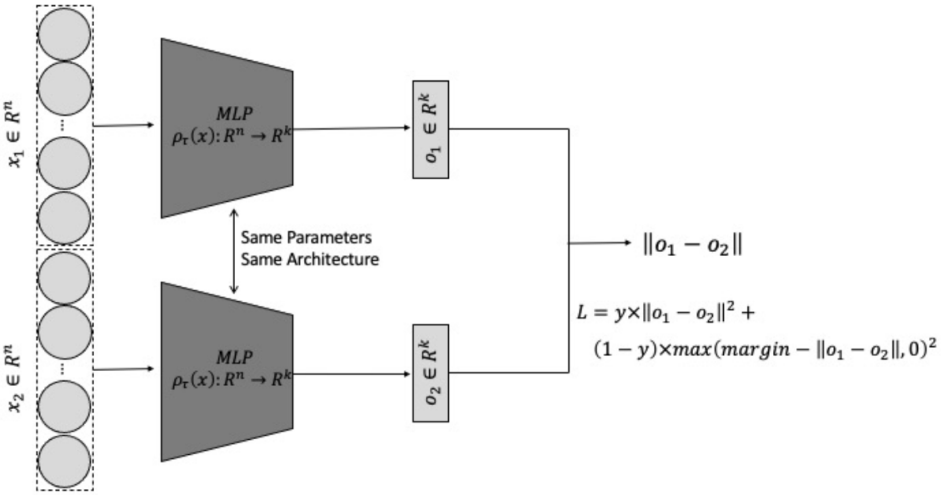

Fig. 1. Siamese Multilayer Perceptron with contrastive loss function applied for the RL task (SNN1).

Loss function. In the training process the *contrastive loss function* is used. It is a distance-based function calculated on pairs and it tries to ensure that semantically similar examples are embedded close together. Given two vectors, x_1 and x_2, and a distance metric d (in this case the Euclidean distance), the loss is calculated as:

$$L = y \times \|\rho(x_1) - \rho(x_2)\|^2 + (1-y) \times max\ (margin - \|\rho(x_1) - \rho(x_2)\|, 0)^2$$

where y is the ground truth relation between the original records r_1 and r_2. In this case $y = 1$ if r_1 and r_2 represent embeddings of two matching records and $y = 0$ otherwise. The *margin* is used to tighten the constraint in the learning process. If two records do not represent the same entity (i.e. are not matching) then the distance between them should be at least the *margin* [5].

3.3.3 SNN2

Architecture. The architecture of the second proposed model is presented in Figure 2.

With SNN2 the absolute values of the differences between the corresponding entrances of the output vectors are passed to a single sigmoidal output unit. More precisely, the output value of the SNN2 is given as:

$$\varphi\left(\sum_j \alpha_j \mid o_1^j - o_2^j \mid\right)$$

where φ is the sigmoidal activation function.
This final layer induces a metric on the learned feature space and determines similarity between the two output vectors o_1 and o_2.

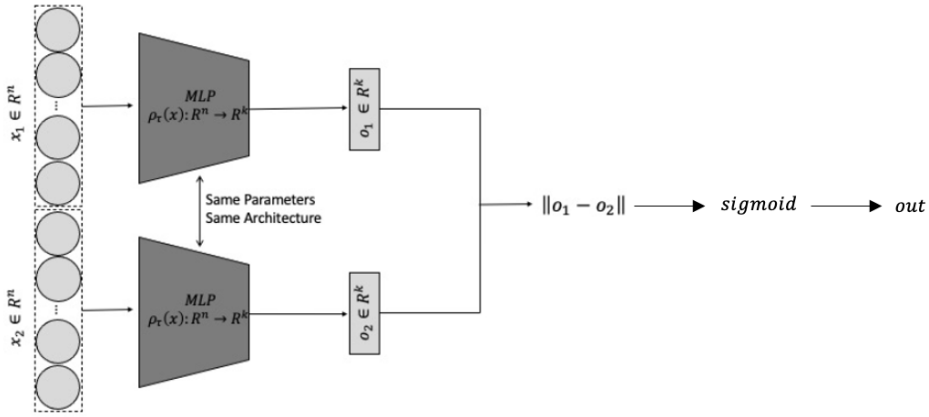

Fig. 2. Siamese Multilayer Perceptron with sigmoidal output unit and cross-entropy loss function (SNN2).

Loss function. In the training process a *cross-entropy loss function* of the following form is used:

$$L(o_1, o_2) = -\left(y \log(out) + (1 - y) \log(1 - out)\right)$$

where p represents the predicted probability that the original records r_1 and r_2 refer to the same entity.

3.3.4 Classification Model

In both cases a training dataset composed of record pairs labelled as match/non-match is required. In the first step, embedding of each record from the training set is obtained with Bert. Following this, SNN1 and SNN2 are trained with a set of embedding pairs labelled as match or non-match.

The SNN1 model is trained so that for each embedding pair fed as input, their high level feature representations are generated and the distance between them is calculated as the output. Following the training process, we calculate the average of the SNN outputs for all matching and non-matching pairs in the training dataset (p_m, p_{nm}). For a new record pair their embeddings are obtained and then fed to SNN. Finally, the pair is classified as a match if the output of SNN is closer to p_m that it is to p_{nm}.
SNN2 is trained so that for each pair of embeddings it gives as an output the probability that the record pair is a match. Hence, for a new pair of records their embeddings are first obtained with Bert. The pair of embeddings is then fed to SNN2. If the probability obtained as output is greater than 0.5 then the pair of records is classified as a match.

3.4 An improvement of the MLP solution: Siamese Autoencoder

We propose and evaluate two different architectures of SNN using a Siamese Autoencoder.

3.4.1 Autoencoder

An Autoencoder is a type of artificial neural network used to learn data encodings in an unsupervised manner. The aim of an autoencoder is to learn a lower-dimensional representation (encoding) of a higher dimensional data, typically for dimensionality reduction, by training the network to capture the most important parts of the input image [8].

3.4.2 A solution with Siamese Autoencoder

Architecture. The architecture of the proposed model is presented in Figure 3.
The model is built with two identical Autoencoders (i.e. sharing the same architecture and parameters) and it requires a training dataset containing record pairs labelled as match/non-match.
The input is composed of two vectors of the same dimension representing embeddings of a pair of records. The goal of the training process is to learn new representations of the vectors, which minimise the distance between matching record pairs and maximise it for non-matching records.

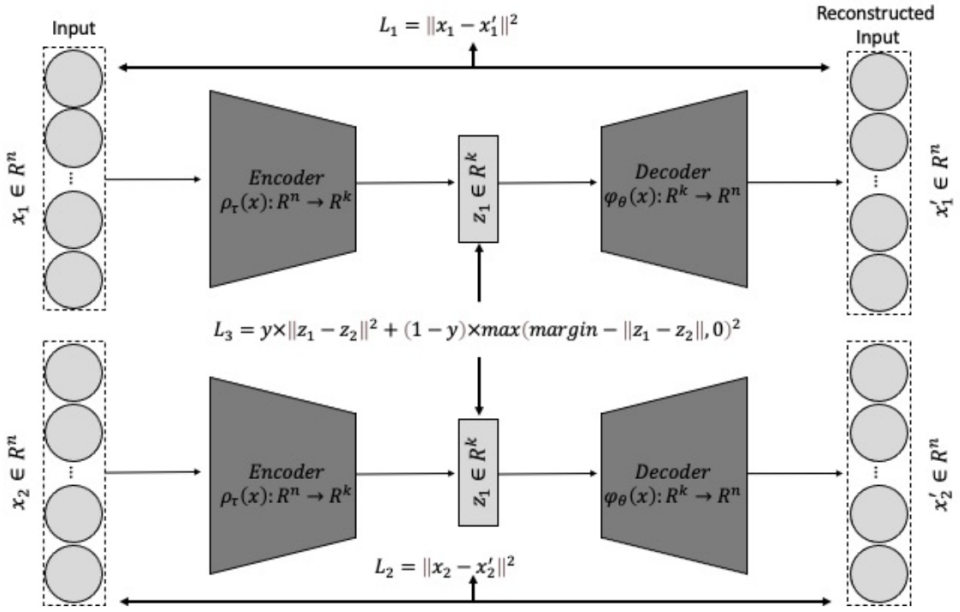

Fig. 3. Siamese Autoencoder applied for the RL task.

Loss function. To minimise the distance between matching record pairs and maximise it for non-matching records, the *contrastive loss* is being calculated on the bottlenecks (encoded inputs) of the two Autoencoders. In addition to this, the *reconstruction losses* of the two Autoencoders are being calculated and included in the final loss function of the Siamese model. Given a pair of embedding vectors $(x1, x2)$, the final hybrid loss function is calculated as:

$$L_{\text{hybrid}} = \alpha \times (\|x_1 - \varphi(\rho(x_1))\|^2 + \|x_2 - \varphi(\rho(x_2))\|^2) + y \times \|\rho(x_1) - \rho(x_2)\|^2 + (1-y) \times max\ (margin - \|\rho(x_1) - \rho(x_2)\|, 0)^2$$

The first component of the loss function represents the reconstruction losses of the two Autoencoders, where ρ and φ represent the functions of the Encoder and the Decoder respectively.
The second component refers to the contrastive loss calculated on the embedded representations of x_1 and x_2 calculated by the twin Autoencoders. α is a parameter indicating the weight of the reconstruction loss. The motivation behind using the hybrid loss function is to downgrade the impact of the contrastive loss function on the training process by combining it with the reconstruction losses [3].

Classification model. Following the training of the model with a set the record pairs labelled as match/non-match, only the Encoder model

is used in the classification process. For a new record pair, their embeddings are first obtained from the language model. Following this, each embedded vector is passed through the Encoder model providing two vectors as output. In the final step, the Euclidean distance is calculated between the output vectors and depending on a pre-defined threshold the pair of records is classified as a match or non-match. Both parameters of the model (i.e. *margin* and *classification threshold*) need to be optimised with the application of a validation dataset.

3.4.3 An hybrid solution: Siamese Autoencoder with Multilayer Perceptron

The above solution is sensitive with the respect to the selection of its two parameters (*margin and classification threshold*).
To avoid this, we propose a hybrid approach based on combination of a Siamese Autoencoder and MLP. In this way we eliminate the need for the *classification threshold* parameter and also make the model much more robust to the selection of the *margin* parameter. A siamese autoencoder is built with two identical Autoencoders.

Architecture. The architecture is presented in Figure 4.
In the first phase, the Siamese Autoencoder is trained with a set of record pairs labelled as match/non-match and the hybrid loss function. Then, a pair of vectors (representing a record pair) is passed through the previously trained Encoder model providing two embedded vectors as an output. The two output vectors are combined into a single vector by computing the absolute values of the differences between their corresponding entrances.
In the second phase, this representation of record pairs with their labels is further applied to train a MLP model for a supervised classification task with binary cross-entropy loss function. At the end, based on the output of the MLP, the record pair is classified as a match or a non-match.
Combining representations of two records into a single vector and training a MLP model to classify such a record pair representation allows us to eliminate the *classification threshold parameter*, which is necessary while classifying a record pair based on their distance. Furthermore, by performing the second training phase we can reduce the impact of the *margin* selection in the training of the Siamese Autoencoder on the final performance of the model.

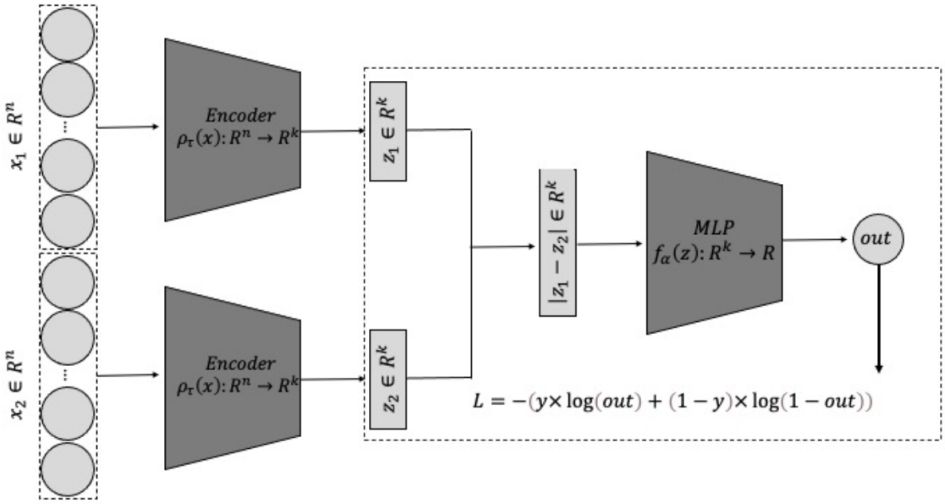

Fig. 4. Siamese Autoencoder with Multilayer Perceptron applied for the RL task.

Loss function. In the first phase the loss function is the same as the architecture shown in Figure 3.
In the second phase, with the MLP introduction, a *cross-entropy loss function* of the following form is used:

$$L(z_1, z_2) = -\left(y \log(out) + (1-y) \log(1-out)\right)$$

where p represents the predicted probability that the original records r_1 and r_2 refer to the same entity.

4 Experimental Evaluation

In this section we present the experimental setup of dataset and the four models architectures presented in the previous section.

4.1 Dataset

The dataset consists of a set of products specifications in JSON format, automatically extracted from multiple e-commerce websites, each of which refers to a real-world product.
Here is an example of a specification:

```
{   "<page title>": "ASUS VT229H & Full Specifications at ebay.
    com",
    "screen size": "21.5''",
    "brand": "Asus",
    "display type": "LED",
    "dimension": "Dimensions: 19.40 x 8.00 x 11.80 inches",
    "refresh rate": "75hz",
    "general features": "Black" }
```

The dataset exhibits high degree of heterogeneity both across and within sources. Attribute names are sparse (only the page title is always present), there are several homonyms (i.e., attributes with the same name but different semantics) and several synonyms (i.e., attributes with the same semantics but different names).
The ground truth for this task is provided in CSV format containing three columns: $left\_spec\_id$, $right\_spec\_id$ and $entity\_id$.

- $left\_spec\_id$ and $right\_spec\_id$ are the identifiers of the real-world products (progressive identifiers without any semantics);
- *label* is a binary value expressing if those products are in match.

An example of the ground truth is shown i Table 1.

left_spec_id	**right_spec_id**	**label**
www.ebay.com//123	www.ebay.com//2	1
catalog.com//254	catalog.com//1	1
catalog.com//254	ca.pcpartpicker.com//1	0

Table 1. Dataset specifications.

4.2 Models configurations

The following configurations exhibit the dimensions of each layer for each one of the models used during the experiments. All configurations refer to the size of the $BERT_{BASE}$ model embeddings.

4.2.1 SNN1 configuration

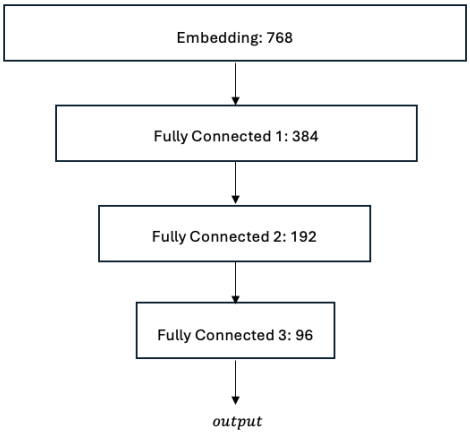

Fig. 5. Layers sizes of the SNN1 model.

4.2.2 SNN2 configuration

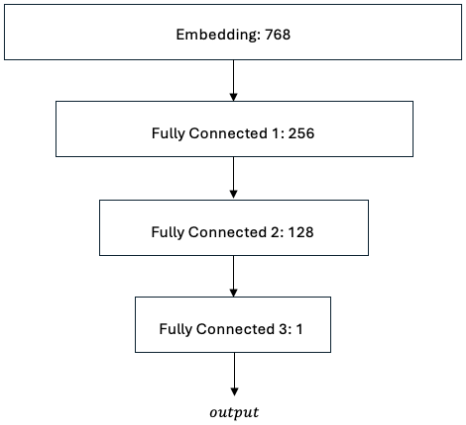

Fig. 6. Layers sizes of the SNN2 model.

4.2.3 Siamese Autoencoder configuration

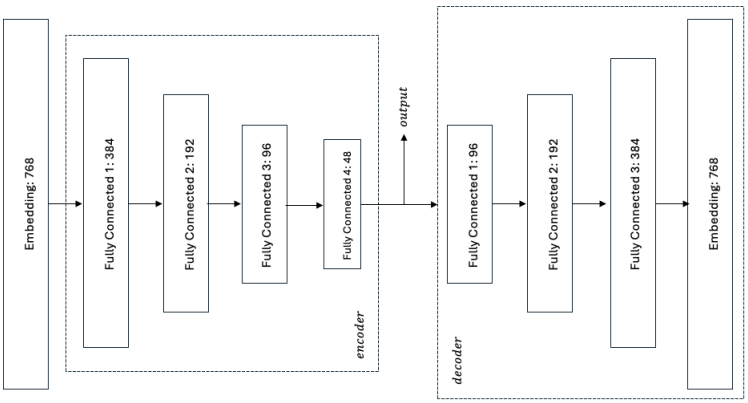

Fig. 7. Layers sizes of the Siamese Autoencoder model.

4.2.4 Hybrid solution configuration

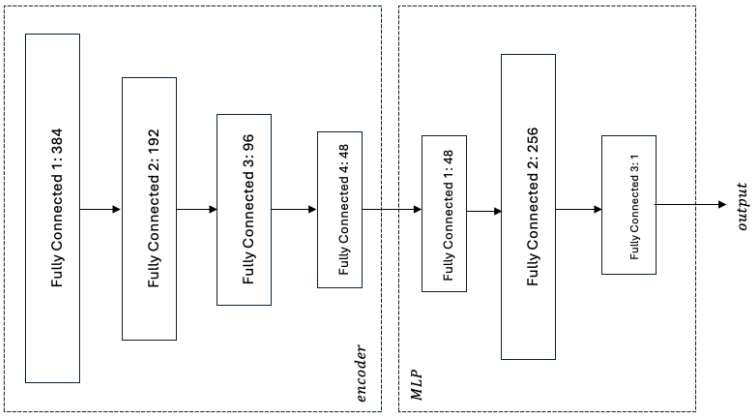

Fig. 8. Layers sizes of the Hybrid model.

4.3 Experimental Setup

All subsequent results refer to the following: the ground-truth was initially composed of negative and positive examples with a 100:1 ratio respectively, so it got filtered bringing the ratio down to 2:1, in order to work with a more balanced dataset, resulting in a better training; subsequently, 75% of this subset was used for the training process, while the remaining 25% was used as the test set to evaluate the model performances.

4.4 Experimental Results

As we can see in Figure 9, all configurations reach an F1-measure above 0.8 after a training process of 100 epochs, with a possible margin for improvement with a longer training and a more accurate tuning of the hyperparameters. These promising results fully justify the choice of improving the originally proposed architectures using BERT instead of Word2Vec to obtain the embeddings of the unstructured entries of the dataset.
As expected, the hybrid model is the one that performs best, achieving an F1-measure of 0.97, demonstrating great generalization capabilities in the pairwise matching task.
A more in-depth overview of each model's performance can be found in Table 2, highlighting in each section which one of the investigated models performs best. Even tho the last architecture is the best performing overall, considering it has the highest F1-measure value, it gets outperformed in terms of recall by other models.

Model	F1	Precision	Recall	TP	FP	TN	FN
SNN1	0.89	0.80	**0.99**	275	69	454	**2**
SNN2	0.84	0.78	0.91	244	69	463	24
AutoEncoder	0.91	0.84	**0.99**	**296**	56	446	**2**
AutoEncoder+MLP	**0.97**	**0.99**	0.95	282	**2**	**500**	16

Table 2. Performance metrics of various models

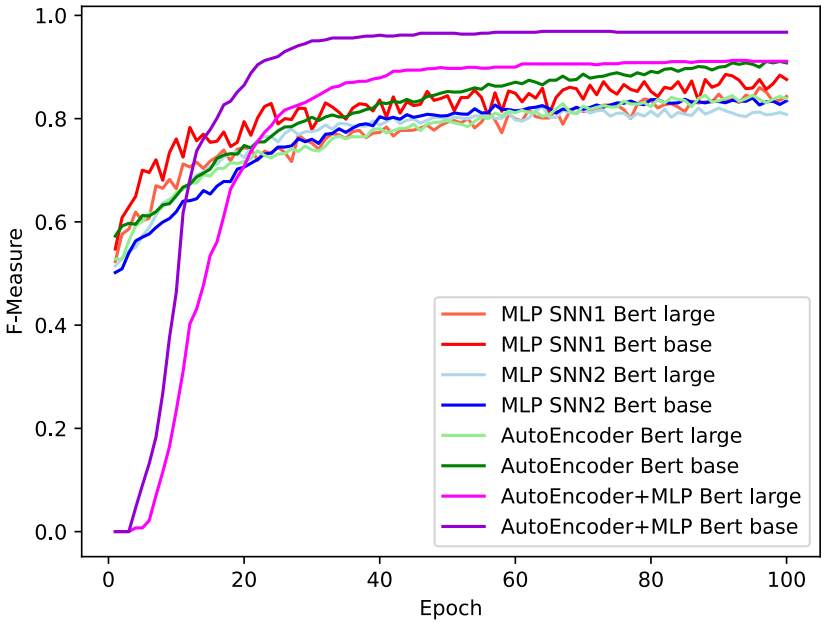

Fig. 9. F-measure over the epochs of each model.

4.5 Links

All the source code of the work can be found at the following GitHub repository: pairwise-matching.
The dataset used to test the approach comes from the DI2KG 2020 challenge, available at: http://di2kg.inf.uniroma3.it/2020/#challenge.

References

1. Gu, Lifang & Baxter, Rohan & Vickers, Deanne & Rainsford, Chris. (2003). Record Linkage: Current Practice and Future Directions. CSIRO Mathematical and Information Sciences Technical Report. 3.
2. Winkler, William. (2014). Matching And Record Linkage. Wiley Interdisciplinary Reviews: Computational Statistics. 6. 10.1002/wics.1317.
3. Jurek-Loughrey, A. (2021). Deep learning based approach to unstructured record linkage. International Journal of Web Information Systems, 17(6), 607-621.
4. Foster, I., Ghani, R., Jarmin, R.S., Kreuter, F., Lane, J. (Eds.). (2020). Big Data and Social Science: Data Science Methods and Tools for Research and Practice (2nd ed.). Chapman and Hall/CRC. https://doi.org/10.1201/9780429324383
5. Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition." ICML deep learning workshop. Vol. 2. No. 1. 2015.
6. Anna Jurek-Loughrey. 2021. Siamese Neural Network for Unstructured Data Linkage. In Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services (iiWAS '20). Association for Computing Machinery, New York, NY, USA, 417–425. https://doi.org/10.1145/3428757.3429106
7. Word Embeddings: https://paperswithcode.com/task/word-embeddings
8. Barman, Dibyendu & Hasnat, Abul & Nag, Rupam. (2022). AN INTRODUCTION TO AUTOENCODERS.
9. Devlin, Jacob et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." North American Chapter of the Association for Computational Linguistics (2019).