Learning Representations of Web Entities for Entity Resolution

Luciano Barbosa
Centro de Informatica
Universidade Federal de Pernambuco
Av. Jornalista Anbal Fernandes, s/n
Recife, Pernambuco, 50740-560, Brazil
luciano@cin.ufpe.br

Abstract
Purpose - Matching instances of the same entity, a task known as entity resolution, is a key step in the process of data integration. This paper proposes a deep learning network that learns different representations of Web entities for entity resolution.
Design/Methodology/Approach - To match Web entities, the proposed network learns the following representations of entities: (1) embeddings, which are vector representations of the words in the entities in a low-dimensional space; (2) convolutional vectors from a convolutional layer, which captures short-distance patterns in word sequences in the entities; and (3) bag-of-word vectors, created by a bow layer that learns weights for words in the vocabulary based on the task at hand. Given a pair of entities, the similarity between their learned representations are used as features to a binary classifier that identifies a possible match. In addition to those features, the classifier also uses a modification of idf (inverse document frequency) for pairs, which identifies discriminative words in pairs of entities.
Findings - The proposed approach was evaluated in 2 commercial and 2 academic entity-resolution benchmarking datasets. The results have shown the proposed strategy outperforms previous approaches in the commercial datasets, which are more challenging, and have similar results to its competitors in the academic datasets.
Originality/value - No previous work has used a single deep-learning framework to learn different representations of Web entities for entity resolution.

Keywords entity resolution, representation learning, Web entity

Paper type Research paper

1. Introduction

The Web contains information about many real-world entities. For instance, information about products can be found on retail websites (e.g. amazon.com), descriptions about movies are present on movie sites (e.g., imdb.com), and information about entities in general is available on Wikipedia. Entity-centric data is currently used in a wide range of applications. Search engines use this data to augment their search results as a response to a queried entity (e.g., movies and actors). Entities and relations extracted from Web structured data are also used to build knowledge bases (Cafarella et al., 2011). Moreover, the entity-centric data itself has been made available on data-as-a-service market sites[a]. To take advantage of such useful content, many approaches have been proposed to collect (Qiu et al., 2015; Meusel et al., 2014; He et al., 2013), extract (Gupta and Sarawagi, 2009; Cafarella et al., 2008) and integrate (Madhavan et al., 2007) Web entity-centric data.

Entity resolution, also known as entity matching or record linkage, is a key step in the process of entity-centric data integration. Its goal is to identify entities that represent the same real-world object. To perform this task effectively, an entity-resolution approach must be able to deal with cases in which a same entity is represented in different ways, or distinct entities are presented in similar ways. Figures 1 and 2 illustrate those cases. In Figure 1, a same entity (a product in this example) has distinct representations on the websites abt.com and buy.com: on buy.com (Figure 1(a)), the product description is much concise than on abt.com (Figure 1(b)). In Figure 2, different entities, which are related products, are represented with similar content on two distinct websites (amazon.com and Google Shopping).

The problem of entity resolution has been extensively studied (Christen, 2008; Bhattacharya and Getoor, 2006; Firmani et al., 2016). A typical workflow of entity matching has two main steps (Christophides et al., 2015): blocking (Efthymiou et al., 2015) and matching (Köpcke et al., 2010). The blocking step is responsible to reduce the number of comparisons, and the matching step decides whether a given pair of entities represents the same entity.

For the matching phase in particular, there are two main classes of solutions according to (Köpcke et al., 2010): the non-learning-based and the learning-based matching approaches. Usually, the non-learning-based approaches rely on off-the-shelf similarity measures to compose the matching function. Entities with a similarity match higher than a given threshold are considered as equivalent. Learning-based techniques for entity resolution, on the other hand, provide different types of similarity measures between the input entities as features to a machine learning classifier that checks the existence of matches.

This paper presents a learning-based method that learns not only the most suitable weights for the features to detect entity matches, as previous approaches,

[a]Examples of data market sites: https://www.factual.com/ and http://www.xignite.com/

```
Linksys EtherFast 4124 24-Port Ethernet Switch-EF4124
/ 24 Autosensing 10/100 Full Duplex, Auto MDI/MDI-X
Ports / Up To 200Mbps / Address Learning, Aging And
Data Flow Control / Compact Size $119.00
```

(a) Entity from Abt

```
Linksys EtherFast EF4124 Ethernet Switch 24 x
10/100Base-TX LAN LINKSYS $64.99
```

(b) Entity from Buy

Fig. 1. Representations of the same entity in two distinct sites.

```
windows vista business spanish full version -LSB- dvd
-RSB- microsoft, 329.95
```

(a) Entity from Amazon

```
microsoft windows vista business - upgrade license -
1 pc - english the windows vista business operating
system is designed to meet the needs of business
organizations of all sizes. 168.99
```

(b) Entity from Google Shopping

Fig. 2. Two *different* entities with similar content.

but also the best weights for the representations of those entities that minimize the training error in a *single deep learning network*. To perform entity matching, the network uses specific layers, namely embedding, convolutional and a modification of bag of words, to learn new representations of the original entity's representation. The similarity between the learned representations of entities are then used as features to a classifier. In addition to those features, a modification of *idf* weighting scheme for pairs is also added to the set of the classifier's features.

The strategy is evaluated on commercial and academic datasets for entity resolution. The results have shown that, on the more challenging datasets (the commercial ones), the approach outperformed previous solutions to this problem and, on the academic ones, it obtained similar results to its competitors.

The remaining of this paper is organized as follows. Section 2 presents the proposed solution for entity resolution. The experimental evaluation is presented in Section 3. Finally, The related work is discussed in Section 4 and Section 5 concludes and gives some possible future directions for this work.

2. Proposed Network

Figure 3 presents an overview of the deep-learning network for entity resolution proposed in this work. Given a pair of entities e_1 and e_2, the network infers whether e_1 and e_2 represent the same entity based on their similarity on different rep-

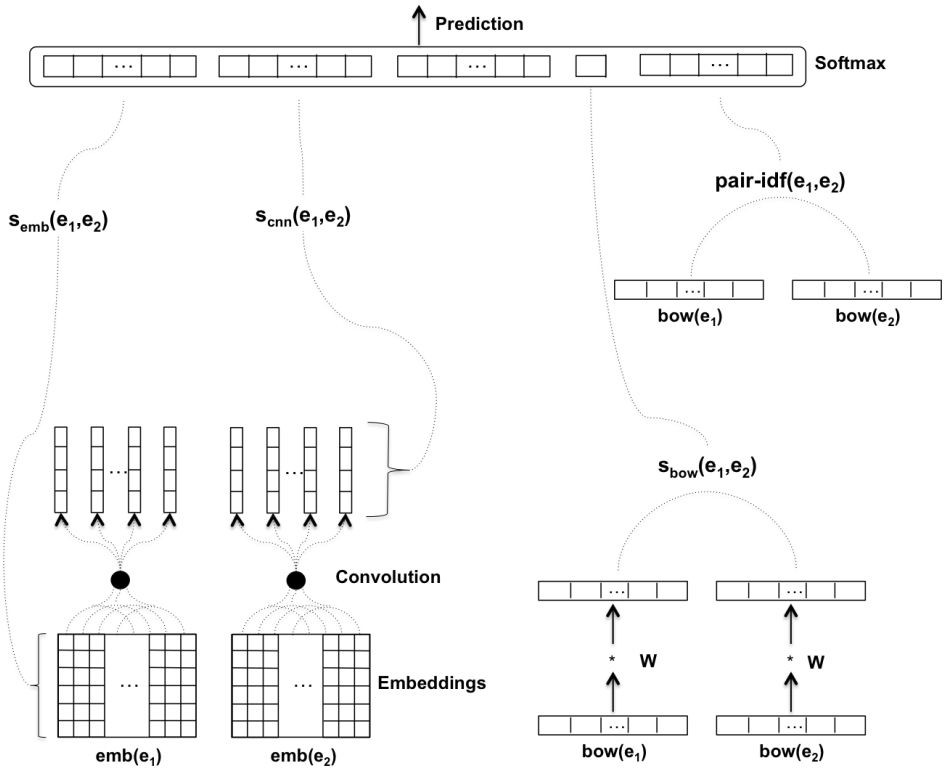

Fig. 3. The deep-learning network for entity resolution.

resentations: embeddings $s_{emb}(e_1, e_2)$; convolution $s_{cnn}(e_1, e_2)$; and bag of words $s_{bow}(e_1, e_2)$, which are provided as features to a softmax classifier. The network also uses, as additional features, a representation of the pair based on the frequency of their words: *pair-idf*(e_1, e_2). This section describes in details the components that compose the network.

2.1. *Embeddings*

Word embeddings (Mikolov et al., 2013) is a dimensionality reduction technique that captures syntactic and semantic regularities of a word into a geometric space (embedding space), represented in a low-dimensional vector (e.g., 100 or 200 dimensions). The weight of each dimension can be learned during training and initialized from vectors pre-trained on a corpus.

Given a pair of entities e_1 and e_2, the network applies a pair of embedding layers to produce the embedding vectors of the words for each entity: $emb(e_1)$ and $emb(e_2)$. The network then computes the similarity of each vector in $emb(e_1)$ with each vector in $emb(e_2)$ and provides those values as features ($s_{emb}(e_1, e_2)$ Figure 3)

to the softmax classifier.

2.2. *CNN*

Convolutional neural network has been successfully applied in different text-related tasks such as name entity recognition (Collobert et al., 2011) and sentiment analysis (Kim, 2014). The convolution operation learns local patterns in the text by taking into account the proximity (or context) of the words.

Given a sequence of vectors E, representing the embedding vectors of the words in an entity, the convolutional layer applies a filter $f \in \mathbb{R}^{k \times d}$ to each sequence of k vectors in E, producing a set of $n - (k - 1)$ convolutional vectors. For each k-sequence of E, the convolution operation calculates the dot product between f and a matrix X, where X is the concatenation of k embedding vectors from the words in the k-sequence of E, plus a bias term b. The result of the dot product is then applied to a non-linear function h (e.g. hyperbolic tangent), creating a convolution vector v of size $|d|$:

$$v = h(f \cdot X + b) \tag{1}$$

The network proposed in this work applies the convolutional layer on the embedding representation of the pair of input entities, $emb(e_1)$ and $emb(e_2)$, producing two set of convolutional vectors: $cnn(e_1)$ and $cnn(e_2)$. The network then computes the similarity of each vector in $cnn(e_1)$ with each vetor in $cnn(e_2)$, and those similarities are used as features ($s_{cnn}(e_1, e_2)$ in Figure 3) by the softmax classifier. In the network implementation, two convolutional layers are used with different values of k: 3 and 5[b].

2.3. *BOW Layer*

The bow layer (dos Santos et al., 2015) is inspired on the $tfidf$ weighting scheme (Baeza-Yates et al., 1999), widely used in Information Retrieval. In the original $tfidf$, the idf of a term t is the inverse (or log of the inverse) of the document frequency of t in a given collection. This scheme gives higher weights to discriminative words in a collection. In classification tasks, however, one might not necessarily be interested in discriminative words in general but mainly for the task at hand. Thus, in the bow layer, instead of having a fixed weight associated with a word, the weight of a word is learned in the training process.

In the context of entity resolution, an entity e is represented by a bow vector: $bow(e) \in \mathbb{R}^{|V|}$, whose size is the collection vocabulary $|V|$. The bow layer calculates the weighted bow vector of e by performing an element-wise multiplication between $bow(e)$ and the weight vector $W \in \mathbb{R}^{|V|}$, which contains the learned weight of each word in the vocabulary. The vector W can be randomly initialized or the idf of the

[b]Different convolutional strategies were tried in the experiments, for instance, adding a convolutional layer on top of other, but they did not produce better results.

words in the collection can be used as initial weights. After computing the weighted bow vectors of the pairs of the input entities, the bow layer calculates the similarity between these vectors and add it to the feature set ($s_{bow}(e_1, e_2)$ in Figure 3) of the classifier.

2.4. *Pair-idf*

Having a high overlap of words between two entities not necessarily indicates they correspond to the same entity. Figure 2 illustrates an example in which two products with similar descriptions represent different real-world objects.

Pair-idf is a weighting scheme that identifies discriminative words in pairs of entities. More specifically, the pair-idf of a term t is the inverse of its frequency in pairs of entities: $1/pf$, where pf is the frequency of t in the pairs of the collection. Terms that do not appear in pairs of the collection has pair-idf equals to 0.

In this solution, given a pair of entities e_1 and e_2, represented by their bow vectors: $bow(e_1)$ and $bow(e_2)$, the network computes the pair-idf vector ($pair\text{-}idf \in \mathbb{R}^{|V|}$, where $|V|$ is the vocabulary size) from the words in the intersection of $bow(e_1)$ and $bow(e_2)$, and provides it as features to the classifier (pair-idf(e_1, e_2) in Figure 3).

2.5. *Training*

The output of the network is the class-membership probabilities provided by the softmax of a given input x:

$$P(Y = i|x, W, b) = \frac{e^{W_i x+b}}{\sum_j e^{W_j x+b}} \tag{2}$$

where i is the index of a class, W is a matrix of weights and b is a bias term. The optimal parameters of the network are learned by minimizing a loss function. In the network, it is used the negative log-likelihood loss function:

$$\mathcal{L}(X) = -\sum_{x \in X} log(P(Y = y^*|x)) \tag{3}$$

where X is the set of training examples x and $P(Y = y^*|x)$ is the probability of x belonging to the true class y^* returned by the softmax function. The network uses the stochastic optimization method Adam (Kingma and Adam, 2015) to minimize the loss function with a learning rate of 10^{-6}.

3. Experimental Evaluation

In this section, the effectiveness of the proposed solution is evaluated over real datasets comparing it with previous approaches.

Dataset	Attributes	# Pairs	Avg. Similarity Between Matches
DBLP-ACM	title, authors, venue, year	4949	0.98
DBLP-Scholar	title, authors, venue, year	19140	0.89
Abt-Buy	name, description, price	3382	0.59
Amazon-Google	name, description, manufacturer, price	4017	0.62

Table 1. Description of the datasets used in the evaluation.

3.1. *Experimental Setup*

Datasets. The proposed strategy is assessed on 4 different datasets widely used to evaluate entity-resolution approaches[c]:

- DBLP-ACM: entries of article references from DBLP[d] and ACM[e].
- DBLP-Scholar: entries of article references from DBLP and Google Scholar[f].
- Abt-Buy: product descriptions from the sites Abt[g] and Buy.com[h].
- Amazon-Google: product descriptions from the sites Amazon.com[i] and Google Products[j].

Table 1 presents more details about the datasets. The dataset with the biggest number of pairs was DBLP-Scholar with 19,140 pairs and the one with the smallest number was Abt-Buy with 3,382 pairs. Table 1 also shows the average similarity between matches for each dataset to have a sense how similar the matches (positive pairs) are in these datasets. As one can see in Table 1, the academic datasets (DBLP-ACM and DBLP-Scholar) have high average similarities (0.98 and 0.89 respectively) whereas the commercial ones (Abt-Buy and Amazon-Google) have much smaller average similarity (0.59 and 0.62 respectively). This might indicate that the commercial datasets are more challenging for entity resolution.

To prepare the datasets for the experiments, the following steps were executed. First, each entity was tokenized using the Stanford Tokenizer (Klein and Manning, 2003). Second, for each dataset, 70% of the data was randomly selected for training and 30% for test. For the negative examples, the blocking strategy was to consider only pairs whose cosine similarity was higher than a given threshold (0.4 in this evaluation) because: (1) this strategy removes trivial non-matching entity pairs; (2) and it is very costly to perform the evaluation over the Cartesian product of all

[c]The datasets were downloaded from Database Group Leipzig web site: https://dbs.uni-leipzig.de/en/
[d]http://dblp.uni-trier.de/
[e]http://dl.acm.org/
[f]https://scholar.google.com/
[g]http://www.abt.com/
[h]http://www.rakuten.com/
[i]https://www.amazon.com/
[j]https://www.google.com/shopping/

entity pairs of each dataset. All attributes of the entries were considered in the experiments.

Entity-Resolution Strategies. The following strategies were executed for evaluation:

- Cosine: the cosine similarity (Baeza-Yates et al., 1999) between pair of entities is one of the baselines. To consider a match, the similarity threshold was varied from 0.4 to 0.9.
- Kenter: Kenter et al. (Kenter and de Rijke, 2015) propose a strategy to calculate the similarity between texts using so-called semantic features. These features are extracted from word embeddings. Examples of features are: a modification of the BM25 algorithm (Robertson and Walker, 1994) that uses the cosine similarity between the word embeddings of the sentences to compose the BM25 formula; and the euclidean distance and cosine similarity of the mean embedding vector that represents E_1 and E_2. These features are then used by an SVM classifier to predict whether a pair of text has the same meaning (paraphrase), in this context, whether a pair refers to the same entity.
- Febrl (Freely extensible biomedical record linkage)[k]: Febrl is a tool that implements different steps of the record linkage process. For entity matching, it uses machine learning algorithms that learn a model from a set of features computed using similarity measures. For this evaluation, Support Vector Machine (SVM) was used as learning algorithm and 3 features: q_gram, tokenset and winkler. The q_gram similarity measures the normalized intersection of q_grams between two strings (q_grams are the substrings of size q in a string). The tokenset similarity calculates the normalized intersection of the tokens between two strings. The winkler (or Jaro-Winkler) (Winkler, 1990) similarity is a variant of string edit similarity suitable for entity resolution. For more details, see (Christen, 2008).
- Dedupe[l]: similar to Febrl, Dedupe (Bilenko and Mooney, 2003) (also known as Marlin) is an approach that uses machine learning for entity resolution. Dedupe performs 3 steps: blocking, matching and grouping. Since the dataset has already been blocked in the experimental setup mentioned previously, this step was not executed on Dedupe. The features used in the matching step are the affine gap penalty string distance, also known as the Smith-Waterman algorithm (Smith and Waterman, 1981), between the words of each entity pair. The logistic regression algorithm is used to build a model from these features. The output from the logistic model is the matching probability of a given pair. To make the decision of matches

[k] https://github.com/fgregg/febrl
[l] https://github.com/dedupeio/dedupe

based on this probability is necessary to define a decision threshold. In this evaluation, this threshold was varied from 0.1 and 0.9, and only reported the best result for each dataset[m]. The final step of Dedupe is grouping. Its goal is to cluster groups composed by the same entities. Dedupe does that by using a hierarchical clustering with centroid linkage.

- Deep learning network: The deep learning approach proposed in this work was implemented using Keras (Chollet, 2015). For training the network, it was used a minibatch of 10 in 3 epochs. To assess the contribution of each component of the solution, the following network configurations were executed:
  - Emb: in this configuration, the feature set used is the vector containing the similarities of the Cartesian product between the embeddings of all terms in the pair of entities provided as input. The number of dimensions of the embeddings was set to 100, and used pre-trained embeddings trained on the English Wikipedia using word2vec (Mikolov et al., 2013). The dot product was used as similarity measure.
  - CNN: this configuration uses the convolutional layer on the top of the embedding layer to produce convolutional vectors. The feature set is the catersian product of the similarities of convolutional vectors of the pairs of entities. The similarity measure used is the dot product. In this configuration, two feature sets are provided to the sofmax layer: the first one with a convolutional layer with window size equals to 3 and the second one with window size equals to 5.
  - CNN+Emb: this configuration is the combination of CNN and Emb mentioned before.
  - CNN+Emb+Bow: in this configuration, the cosine similarity of the bag-of-word representations of the pair of input entities provided by the Bow layer is added to the feature set of $CNN + Emb$.
  - CNN+Emb+Bow+Pair-Idf: this network adds to $CNN + Emb + Bow$ the pair-idf feature set.

3.2. *Evaluation*

Table 2 and 3 show the F-measure (or F-1 score) on the positive matches and the overall accuracy for each approach for the 4 datasets. Overall, the deep learning approach (CNN+Emb+Bow+ Pair-Idf) outperforms the baselines on the commercial datasets (Abt-Buy and Amazon-Google) and obtains similar results to same baselines on the academic ones (DBLP-ACM and DBLP-Scholar).

Regarding the ABT-Buy dataset, the network obtained an F-measure of 0.59 and accuracy of 0.76, whereas the baseline with the highest F-measure was Dedupe with

[m]The decision threshold for DBLP-ACM and DBLP-Scholar datasets was 0.5, and 0.3 for Abt-Buy and Amazon-Google datasets

	DBLP-ACM		**DBLP-Scholar**	
	Accuracy	F1	Accuracy	F1
Cosine-0.4	0.47	0.62	0.33	0.44
Cosine-0.5	0.8	0.82	0.87	0.81
Cosine-0.6	0.92	0.92	0.92	0.86
Cosine 0.7	0.94	0.94	0.89	0.77
Cosine-0.8	0.87	0.85	0.82	0.53
Cosine-0.9	0.64	0.35	0.76	0.27
Kenter	**0.95**	**0.95**	0.94	0.89
Febrl	**0.95**	**0.95**	**0.96**	**0.93**
Dedupe	0.89	0.88	**0.96**	0.92
Emb	0.94	0.94	0.92	0.87
CNN	0.94	0.93	0.94	0.9
CNN+Emb	**0.95**	**0.95**	0.95	0.92
CNN+Emb+Bow	**0.95**	**0.95**	**0.96**	0.92
CNN+Emb+Bow+Pair-Idf	**0.95**	**0.95**	**0.96**	**0.93**

Table 2. Accuracy and F-1 score of all approaches for the academic datasets: DBLP-ACM and DBLB-Scholar.

	Abt-Buy		**Amazon-Google**	
	Accuracy	F1	Accuracy	F1
Cosine-0.4	0.28	0.41	0.22	0.32
Cosine-0.5	0.67	0.51	0.54	0.34
Cosine-0.6	0.73	0.39	0.66	0.26
Cosine 0.7	0.69	0.16	0.68	0.16
Cosine-0.8	0.68	0.02	0.68	0.08
Cosine-0.9	0.67	0	0.67	0.01
Kenter	0.71	0.32	0.73	0.48
Febrl	0.69	0.12	0.49	0.54
Dedupe	0.6	0.53	0.48	0.48
Emb	0.69	0.4	0.75	0.57
CNN	0.71	0.51	0.77	0.59
CNN+Emb	0.71	0.51	0.77	0.64
CNN+Emb+Bow	0.74	0.57	0.78	0.66
CNN+Emb+Bow+Pair-Idf	**0.76**	**0.59**	**0.79**	**0.68**

Table 3. Accuracy and F-1 score of all approaches for the commercial datasets: Abt-Buy and Amazon-Google.

0.53, and with the highest accuracy was Kenter with 0.71. On the Amazon-Google dataset, the approach's performance was also far superior than the baselines with F-measure equals to 0.68 and accuracy equals to 0.79, whereas the best baselines

obtained F-measure=0.54 and accuracy=0.49 for Febrl, and F-measure=0.48 and accuracy=0.73 for Kenter. If one considers the average similarity between matches on a dataset as a measure of how challenging is a dataset for entity resolution, the network obtains better results than the baselines in the more challenging datasets for this task, the commercial ones, as Table 1 suggests.

As mentioned before, some baselines obtained similar results to the approach for the academic datasets. For instance, both approaches, Febrl and Kenter, on the DBLP-ACM dataset obtained accuracy equals to 0.95 and F1 equals to 0.95, and on the DBLP-Scholar dataset, Febrl had an accuracy of 0.96 and F-measure of 0.93. These are the same results the approach obtained on these two datasets. However, as Table 1 indicates, finding the matches in those datasets does not seem to be hard since the average cosine similarity of matches in the DBLP-ACM dataset is 0.98 and 0.89 for the DBLP-Scholar dataset, meaning that there is a great overlap of words between pair matches, which are bibliographic citations in the case of these two datasets.

Table 2 and Table 3 also show the results of the different combinations of the network. In all 4 datasets, the results improved as more features were added to the model. The improvements, however, were greater in the most challenging datasets: the commercial ones. For instance, if one compares the F-measure values of Emb versus CNN+Emb+Bow+Pair-Idf on the Amazon-Google dataset, it improves 19.2% and on the Abt-Buy dataset 47.5%, whereas for the DBLP-ACM dataset, the gain in F-measure was 1% and the DBLP-Scholar 6%.

4. Related Work

In this section, it is discussed previous entity resolution approaches and deep learning strategies of metric learning.

4.1. *Entity Resolution*

There are two main classes of solutions for entity resolution (Köpcke et al., 2010): the non-learning (Thor and Rahm, 2007; Benjelloun et al., 2009; Xiao et al., 2011) and the learning-based match approaches (Bilenko and Mooney, 2003; Christen, 2008).

Usually, the non-learning based approaches rely on pre-defined similarity measures to compose the match function. Entities with a similarity match higher than a given threshold are considered as equivalent. PPJoin+ (Xiao et al., 2011) proposes a strategy to efficiently compute matches of pairs of records. Matches are computed based on a similarity function passed as a parameter. Only pairs with similarity higher than a given threshold (also a parameter) are considered matches. In the same direction, SERF (Benjelloun et al., 2009) creates an efficient strategy for entity resolution. For that, it considers previously matched pairs to avoid redundant comparisons. The matchers (similarity functions) are considered as black

boxes. As opposed to those approaches, this work does not focus on efficiency of entity resolution but how to compute the correct matches.

Another work (Efthymiou et al., 2017) proposes three non-learning based approaches to match entities present in Web tables to entities in knowledge bases. The first approach uses the entity context in the Web tables to find the correspondent entities in the knowledge base. The second one, the semantic embedding method, represents the entities as distributed representations, created from the knowledge base graph. A disambiguation graph is built from all candidates of a given entity in the knowledge base and the weight of the edges is the cosine similarity between the distributed representations of the entities. The entities with the highest values of PageRank in this graph are selected. Finally, the third strategy takes into consideration not only the instance information of the instances in the knowledge base and web tables, but also their schemata. In our work, we have a different problem setting: we are performing matching between entities in flat tables, not between tables and knowledge bases. For this reason, as opposed to them, we can not rely on the structure of knowledge bases to perform this task.

The learning-based approaches use pre-defined functions as features to a machine learning classifier that combines them, assigning weights that minimize the error on a given training data. FEBRL (Christen, 2008) is a framework that provides a set of string similarity measures to be used as features by a Support Vector Machine (SVM) classifier. Another learning-based strategy, Dedupe (Bilenko and Mooney, 2003), uses a logistic regression algorithm to perform entity resolution using features such as the affinite gap penalty between strings. The experimental evaluation (Section 3) shows the proposed strategy outperforms FEBRL and Dedupe in the commercial datasets and have similar results in academic ones.

4.2. *Deep Learning for Entity Matching*

In their seminal work (Bromley et al., 1993), Bromley et al. proposed a neural network architecture (so-called siamese network) for signature verification: from an input with two signatures – the real one and the one to be verified – the network outputs a distance measure. Signatures with distance lower than a threshold are considered as authentic. The network is composed of two branches with shared weights of stacks of two convolutional and averaging layers that transform the inputs into signature feature vectors. The output of the network is the cosine distance between the feature vectors. Similar to them, Chopra et al. (Chopra et al., 2005) use a siamese network to the problem of face verification. Similar to them, the proposed approach uses a siamese network but instead of using a threshold on a distance measure to detect a match, distance measures from different representations of the input are provided as features to a binary classifier.

Deep learning approaches have also been used in natural language processing tasks (Yin and Schütze, 2015; dos Santos et al., 2015). For instance, Yin and Schutze propose a neural network for paraphrase detection. Their strategy extracts features

to a binary classifier from different representations of the input pair namely a modification of the convolutional layer for sentences (Kalchbrenner et al., 2014) and from pooling layers (averaging and max). The proposed approach in contrast uses a standard convolutional layer and does not obtain features from pooling layers, which makes it easier to implement. In addition, the proposed approach adds more features to the binary classifier such as the bow layer and the *pair-idf*.

5. Conclusions

This paper presents a deep learning strategy for entity resolution. Given a pair of entities, the proposed approach learns different representations of those, namely embeddings, convolutional and a modification of bag-of-words, computes the similarity between the entities in the pair using those representations, and use them as features on a binary classifier. Moreover, a feature set resulting from a modification of *idf* for pairs is added to set of classifier's features. The experimental evaluation over real datasets has shown the proposed approach outperforms previous approaches in the commercial datasets, which seem to be more challenging, and has comparable results with previous approaches in the academic datasets.

As possible future work, different deep learning architectures can be tried for this problem and, instead of a fixed weight *pair-idf*, to learn the weights of the *pair-idf* based on the training data.

References

Baeza-Yates, R., Ribeiro-Neto, B. et al. (1999). *Modern information retrieval*, Vol. 463, ACM press New York.

Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S. E. and Widom, J. (2009). Swoosh: a generic approach to entity resolution, *The VLDB JournalThe International Journal on Very Large Data Bases* **18**(1): 255–276.

Bhattacharya, I. and Getoor, L. (2006). A latent dirichlet model for unsupervised entity resolution., *Proceedings of the 2006 SIAM International Conference on Data Mining*, pp. 47–58.

Bilenko, M. and Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures, *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 39–48.

Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E. and Shah, R. (1993). Signature verification using a "siamese" time delay neural network, *IJPRAI* **7**(4): 669–688.

Cafarella, M. J., Halevy, A. and Madhavan, J. (2011). Structured data on the web, *Communications of the ACM* **54**(2): 72–79.

Cafarella, M. J., Halevy, A., Wang, D. Z., Wu, E. and Zhang, Y. (2008). Webtables: exploring the power of tables on the web, *Proceedings of the VLDB Endowment* **1**(1): 538–549.

Chollet, F. (2015). Keras: Theano-based deep learning library, *Code: https://github.com/fchollet. Documentation: http://keras. io* .

Chopra, S., Hadsell, R. and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 539–546.

Christen, P. (2008). Febrl: a freely available record linkage system with a graphical user interface, *Proceedings of the second Australasian workshop on Health data and knowledge management*, pp. 17–25.

Christophides, V., Efthymiou, V. and Stefanidis, K. (2015). Entity resolution in the web of data, *Synthesis Lectures on the Semantic Web* **5**(3): 1–122.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. and Kuksa, P. (2011). Natural language processing (almost) from scratch, *The Journal of Machine Learning Research* **12**: 2493–2537.

dos Santos, C., Barbosa, L., Bogdanova, D. and Zadrozny, B. (2015). Learning hybrid representations to retrieve semantically equivalent questions, *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pp. 694–699.

Efthymiou, V., Hassanzadeh, O., Rodriguez-Muro, M. and Christophides, V. (2017). Matching web tables with knowledge base entities: from entity lookups to entity embeddings, *International Semantic Web Conference*, Springer, pp. 260–277.

Efthymiou, V., Stefanidis, K. and Christophides, V. (2015). Big data entity resolution: From highly to somehow similar entity descriptions in the web, *Big Data (Big Data), 2015 IEEE International Conference on*, IEEE, pp. 401–410.

Firmani, D., Saha, B. and Srivastava, D. (2016). Online entity resolution using an oracle, *Proceedings of the VLDB Endowment* **9**(5): 384–395.

Gupta, R. and Sarawagi, S. (2009). Answering table augmentation queries from unstructured lists on the web, *Proceedings of the VLDB Endowment* **2**(1): 289–300.

He, Y., Xin, D., Ganti, V., Rajaraman, S. and Shah, N. (2013). Crawling deep web entity pages, *Proceedings of the sixth ACM international conference on Web search and data mining*, pp. 355–364.

Kalchbrenner, N., Grefenstette, E. and Blunsom, P. (2014). A convolutional neural network for modelling sentences, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pp. 655–665.

Kenter, T. and de Rijke, M. (2015). Short text similarity with word embeddings, *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 1411–1420.

Kim, Y. (2014). Convolutional neural networks for sentence classification, *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pp. 1746–1751.

Kingma, D. P. and Adam, J. B. (2015). A method for stochastic optimization, *International Conference on Learning Representation*.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing, *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pp. 423–430.

Köpcke, H., Thor, A. and Rahm, E. (2010). Evaluation of entity resolution approaches on real-world match problems, *Proceedings of the VLDB Endowment* **3**(1-2): 484–493.

Madhavan, J., Jeffery, S., Cohen, S., Dong, X., Ko, D., Yu, C. and Halevy, A. (2007). Web-scale data integration: You can only afford to pay as you go, *Proceedings of Conference on Innovative Data Systems Research*, pp. 342–350.

Meusel, R., Mika, P. and Blanco, R. (2014). Focused crawling for structured data, *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pp. 1039–1048.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. and Dean, J. (2013). Distributed representations of words and phrases and their compositionality, *Advances in neural information processing systems*, pp. 3111–3119.

Qiu, D., Barbosa, L., Dong, X. L., Shen, Y. and Srivastava, D. (2015). Dexter: Large-scale discovery and extraction of product specifications on the web, *Proceedings of the VLDB Endowment* **8**(13): 2194–2205.

Robertson, S. E. and Walker, S. (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval, *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 232–241.

Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences, *Journal of molecular biology* **147**(1): 195–197.

Thor, A. and Rahm, E. (2007). Moma-a mapping-based object matching system., *Proceedings of Conference on Innovative Data Systems Research*, pp. 247–258.

Winkler, W. E. (1990). String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage., *Proceedings of the Section on Survey Research Methods*, pp. 354–359.

Xiao, C., Wang, W., Lin, X., Yu, J. X. and Wang, G. (2011). Efficient similarity joins for near-duplicate detection, *ACM Transactions on Database Systems* **36**(3): 1–41.

Yin, W. and Schütze, H. (2015). Convolutional neural network for paraphrase identification., *HLT-NAACL*, pp. 901–911.