Poster Paper

Entity Matching on Web Tables: a Table Embeddings Approach for Blocking

Anna Lisa Gentile
University of Mannheim
annalisa@informatik.uni-mannheim.de

Petar Ristoski
University of Mannheim
petar.ristoski@informatik.uni-mannheim.de

Steffen Eckel
University of Mannheim
steffen.eckel@students.uni-mannheim.de

Dominique Ritze
University of Mannheim
dominique@informatik.uni-mannheim.de

Heiko Paulheim
University of Mannheim
heiko@informatik.uni-mannheim.de

ABSTRACT

Entity matching, or record linkage, is the task of identifying records that refer to the same entity. Naive entity matching techniques (i.e., brute-force pairwise comparisons) have quadratic complexity. A typical shortcut to the problem is to employ blocking techniques to reduce the number of comparisons, i.e. to partition the data in several blocks and only compare records within the same block.

While classic blocking methods are designed for data from relational databases with clearly defined schemas, they are not applicable to data from Web tables, which are more prone to noise and do not come with an explicit schema. At the same time, Web tables are an interesting data source for many knowledge intensive tasks, which makes record linkage on Web Tables an important challenge. In this work, we propose an unsupervised approach to partition the data, that does not exploit any external knowledge, but only relies on heuristics to select the blocking attributes. We compare different partitioning methods: we use (i) clustering on bag-of-words, (ii) binning via Locality-Sensitive Hashing and (iii) clustering using word embeddings. In particular, the clustering methods show good results on a standard dataset of Web Tables, and, when combined with word embeddings, are a robust solution which allows for computing the clusters in a dense, low-dimensional space.

Keywords

Instance matching; Web tables; blocking methods; word embeddings

1. INTRODUCTION

Entity matching aims at identifying different descriptions in unstructured or (semi-)structured textual content that refer to the same real-world entity. Similarly, the deduplication task (or record linkage) looks for nearly duplicate records in relational data, usually amongst data points that refer the same entity type. In both cases, if addressed in a brute force fashion, the matching task has quadratic complexity, as it requires pairwise comparisons of all the records. The most widely adopted solution for this problem is to group records in blocks before comparing them, a pre-processing step usually referred to as *blocking*. Blocking offers a compromise between the number of comparisons to perform and the number of missed entity matches [3].

Traditionally, a plethora of blocking techniques have been proposed to reduce number of comparisons [5, 6, 12, 13], especially for the deduplication task. These exploit specific clues from the data schema, ad hoc similarity functions and mapping rules, as well as background knowledge of the domain and of the type of the entities.

In this paper, we focus on Web tables (i.e., tables on HTML pages), which stand in the middle ground between unstructured Web content and relational data. From a structural point of view records in Web tables resemble records in database tables. Nevertheless, they come with no schema attached, making the direct usage of traditional blocking techniques not applicable, since we cannot rely on any type of assumption for the data nor have domain knowledge [3]. Thus, in this scenario, the high number of entity types and the representational heterogeneity, even for entities of the same semantic type, make traditional blocking techiques hard to apply. Therefore, we argue for the need of a domain agnostic representation of Web tables, which has the property of being succinct and can be used in a similar fashion as the signatures in the traditional blocking techniques. We explore different heuristics based on bag of words and word embeddings, combined with different clustering methods.

By using a publicly available gold standard (Section 4.1), we measure the pair comparison reduction ratio and the pair completeness of the instance matching task, when performed after the blocking step. We show that the blocking based on table embeddings leads to the best tradeoff between (i) reduction ratio on the number of pair comparisons to perform and (ii) pair completeness (recall on entity matches). Moreover, it is robust with respect to the type of table preprocessing (Section 3.1), while also offering a succinct representation for the tables.

The main contribution of this work is twofold. First, we propose a novel solution to represent tables in a latent space using Neural Language Models (NLM). NLM have been proved successful in replacing the classical bag of word representation of text (binary feature vector, where each vector index represents one word) with a latent representation with a lower vector dimensionality. Second, we exploit the latent table representations to perform blocking in the context of entity matching in Web tables.

2. STATE OF THE ART

Entity matching is a task with quadratic complexity, therefore, when applied to larger data collections, it is necessary to reduce the number of comparisons to be performed. A common way to reduce complexity is the use of so-called *blocking strategies* to reduce the search space [13] and achieve a reasonable compromise between the number of comparisons and the number of missed entity matches [3]. Traditionally, blocking techniques exploit specific criteria in the data schema to split the data before performing entity comparisons, only utilizing the values of some key attributes. A typical example is the Sorted Neighborhood Method (SNM) [5], which performs sorting of the records according to a specifically chosen Blocking Key.

Content based blocking strategies usually look for common tokens between two entities. The search for common tokens can be performed in entities descriptions or it can be restricted to the values of attributes that overall have similar values [12]. In a naive scenario, each token defines a new block and all entities that share the token end up in that block (here, entities may belong to multiple blocks); more sophisticated methods index and rank tokens, so that the search is restricted to the n most frequent ones [8]. Another commonly used technique is *Locality-Sensitive Hashing (LSH)*, which produces effective signatures of records to perform fast comparisons amongst entities. Specifically, Duan et al. [4] used LSH to perform instance based ontology matching. The underlying assumption of their work is that discovering relationships in data from different sources can only be achieved after correctly typing the data. Their objective is to perform comparisons amongst *entity types*. The representation of each type is the sum of all its entities, which can be seen as a big document, where they can exploit the tf-idf weighting schema to select relevant tokens.

Multi-Block is a blocking strategy that uses a multidimensional index in which similar objects are located near each other. In each dimension, the entities are indexed by a different property to achieve effect retrieval [6]. To boost recall, data is enriched by interlinking to DBpedia [9].

Differently from available state of the art we propose a strategy which is (i) completely agnostic w.r.t. the data schema, (ii) does not rely on external knowledge and (iii) performs only minimal preprocessing of the data.

3. A BLOCKING APPROACH USING TABLE EMBEDDINGS

The blocking step is performed at table label. We propose an adaptation of neural language models (NLM) to represent table embeddings, which provide a succinct latent representation of tables without relying on any domain knowledge. In the following, we describe the preliminary table preprocessing step, followed by the proposed embedding strategy.

3.1 Table Preprocessing

As Web tables are different from relational tables, i.e., they are schema-free and prone to noise, preprocessing and normalization are required. We reuse components from the "Mannheim Search Join Engine" [10] to: (i) identify pseudo key attributes (the subject column); (ii) recognize table header structures; and (iii) identify data types.

To identify the subject column we apply the heuristic proposed by [14] of choosing the column of type string with the highest number of unique values. In case of a tie, the leftmost column is used. The subject column basically contains entity names which act as pseudo-keys for the table [2, 15, 16]. In the example depicted in Table 1, the first column is used as a subject column.

For detecting the headers, we assume that the header row is the first non-empty row. In the example depicted in Table 1, the first row is used as headers.

For data type detection, we use about 100 manually defined regular expressions to detect numeric values, dates, and links. Based on the data type, the values of each column are normalized, e.g., the string values are lower-cased and special characters are removed. Furthermore, for this work we replace numbers, timestamps, and geo-coordinates with a static value, as our approach currently does not handle numeric values. For example, given Table 1, the first two columns will be recognized as strings, and the last two columns as numerical. The first row will be recognized as the header row, and the first column as the subject attribute. After preprocessing, the table will look like Table 2.

Table 1: Example table of countries

Country	Capital	Population	GDP (USD)
Germany	Berlin	80M	46,268.64
France	Paris	60M	42,503.30
United Kingdom	London	64.1M	41,787.47

Table 2: Preprocessed table

h:country	h:capital	h:population	h:gdp_(usd)
v:germany	v:berlin	NUM	NUM
v:france	v:paris	NUM	NUM
v:united_kingdom	v:london	NUM	NUM

3.2 Table Embeddings

NLMs are explicitly built to take into account the order of words in text documents and to encode a stronger statistical dependence amongst words which are closer in the sequence. As the model is originally designed for raw text, where the sequence of words is naturally derivable from the sentences, we first need to transform tables to word sequences, which can be used to train a NLM. We consider table values and table attributes instead of word sequences. Thus, in order to apply such approaches on table data, we first have to transform the tables into sequences of values and attributes, which can be considered as sentences. Using those sentences, we can train the same neural language models to represent each value and attribute in the table as a vector of numerical values in a latent feature space.

We propose three general approaches to transform tables to sentences: (i) *attributes model*: the header row is

converted into a sequence of attributes, e.g., for Table 2 the produced sequence is: *"h:country h:capital h':population h:gdp_(usd)"*; (ii) *entities model*: the subject column is converted into a sequence of entities, e.g., for Table 2 the produced sequence is: *"v:germany v:france v:united_kingdom"*. (iii) *attributes and entities model*: we convert the table into a sequence of triples of the form <entity, header, value>, where the entity is the subject, the header is the predicate, and the corresponding value is the object; e.g. for Table 2 the produced sequence is: *"v:germany h:capital v:berlin; v:germany h:population NUM; ... ; v:united_kingdom h:gdp_(usd) NUM"*.

As NLM, we use *word2vec* [11] , a two-layer neural net model that learns word embeddings from raw text. Specifically, we employ the skip-gram model, which tries to predict the context words given a target word. Given a sequence of words $w_1, w_2, w_3, ..., w_T$ and a context window c, the objective of the skip-gram model is to maximize the following average log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_{t+j}|w_t), \quad (1)$$

The probability $p(w_{t+j}|w_t)$ is calculated using the softmax function:

$$p(w_o|w_i) = \frac{exp(v'^T_{wo} v_{wi})}{\sum_{w=1}^{V} exp(v'^T_w v_{wi})}, \quad (2)$$

where v_w and v'_w are the input and the output vector of the word w, and V is the entire vocabulary.

Directly calculating the softmax function is inefficient, as it is proportional to the size of V, therefore we used *negative sampling* as an optimization technique, following the approach discussed in [11].

Once the training is finished, all words (i.e., table values and attributes in our case) are projected into a lower-dimensional feature space, and semantically similar words are positioned close to each other. We can then use the produced latent representation of tables to calculate the similarity between two tables using the standard cosine similarity measure.

4. EXPERIMENTS

The aim of the experiments is to verify how the proposed table embeddings perform in reducing the complexity of entity matching, specifically in reducing the number of performed pair comparisons, without loosing recall. As entity matching per se is not the focus of this paper, we only consider the *pair completeness*, i.e., the number of entity pairs that are contained in the same bucket identified by the blocking mechanism. That way, we can directly evaluate the tradeoff between (i) the reduction ratio achieved with the blocking and (ii) the pair completeness, i.e. the ratio of matches that can be potentially identified.

4.1 The T2D Dataset

As a gold standard, we use the publicly available T2D dataset[1]. T2D contains a subset of the Web Data Commons Web Tables Corpus[2] for which schema-level and instance-level correspondences to DBpedia 2014 are provided. We use the 233 tables of T2D for which rows are mapped to entities in the DBpedia knowledge base (for a total 26,124 entity-level correspondences). As we are not interested in matching to DBpedia as an external knowledge base, but rather finding correspondences *within* a dataset, we transform the matches to internal matches within the tables, i.e., for every pair of table rows mapped to the same DBpedia entity, we create a correspondence amongst the two table rows. This process produces 50,072 instance correspondences.

To build the table embedding models, we use the T2D dataset and the WikiTables dataset[3]. The WikiTables dataset has been extracted from Wikipedia pages in the course of the WikiTables project [1]. The corpus consists of 1.35 million Wikipedia tables. Additionally, we extracted 365,194 tables from the Wikipedia 2015 dumps. We build Skip-Gram models with the following parameters: window size = 10; number of iterations = 15; negative sampling for optimization; negative samples = 25; 500 latent dimensions. All the models, as well as the code, are publicly available[4].

4.2 Results

Overall, we compared two different representations for tabular data. We use the classical bag of word model (bow) and the table embeddings. We build all the representations using four different input features, as identified in the table pre-processing step (Section 3.1): (i) the full content of the table, (ii) the table header, (iii) the table subject column and (iv) the combination of the table subject column and the table header. For (i) and (iii), we use the attributes and entities model, for (ii), we use the attributes model, and for (iii), we use the entities model, as described in section 3.2. Once the representations are built, we use either unsupervised clustering (k-means) or binning via Locality-Sensitive Hashing (LSH) to partition tables into bins.

Table (3) shows the results of applying k-means to the representations (according to the four different input features) generated via *table embeddings* (shortened in the table as *emb*) and *bag of words* (shortened in the table as *bow*). As we are looking for a good tradeoff between the pair comparison reduction ratio and pair completeness of matched instances, we also report the harmonic mean (*hm*) of those two values for each run. The first observation concerns the input features. In all scenarios the table headers alone are not enough to perform effective binning. The subject column (also combined with the table header) instead leads to good performance, regardless of the representation, with comparable performance (>90%) for 10 and 20 bins. Furthermore, we can observe that the word embeddings are also competitive when using the full table as input, again with performance >90% for 10 and 20 bins. This means that the potentially expensive and error-prone detection of header and key columns can be omitted when using that approach.

We also perform LSH using a java-LSH implementation[5] both with the standard bow model as well as using the table embeddings as input (using the Super-Bit algorithm [7]). While the table embeddings representation outperformed the standard bow, the general performance were quite low (hm around 60%). This is a direct consequence of the size of the gold standard. As we perform the binning at table level and not at instance level, the dataset only contains 233

[1]http://webdatacommons.org/webtables/goldstandard.html

[2]http://webdatacommons.org/webtables/

[3]http://downey-n1.cs.northwestern.edu/public/

[4]http://data.dws.informatik.uni-mannheim.de/table2vec/

[5]https://github.com/tdebatty/java-LSH

Table 3: Pair comparison reduction ratio (rr) and pair completeness (pc) of the instance matching task and their harmonic mean (hm), when performed after blocking using either table embeddings (emb) or simple bag of word model (bow), when run on either the full content of the table (all), the table header (h), the table subject column (k) or the table subject column and the table header (k+h).

input	#k	rr-emb	pc-emb	hm-emb	rr-bow	pc-bow	hm-bow
all	5	0.75	0.98	0.85	0.74	0.89	0.81
all	10	0.85	0.98	**0.91**	0.83	0.5	0.62
all	20	0.91	0.97	**0.94***	0.92	0.5	0.65
all	30	0.93	0.62	0.74	0.91	0.29	0.44
all	40	0.92	0.85	0.88	0.92	0.4	0.56
all	50	0.95	0.65	0.77	0.95	0.32	0.48
h	5	0.79	0.8	0.79	0.73	0.47	0.57
h	10	0.87	0.66	0.75	0.87	0.42	0.57
h	20	0.94	0.38	0.54	0.91	0.41	0.57
h	30	0.96	0.33	0.49	0.93	0.26	0.41
h	40	0.96	0.32	0.48	0	1	
h	50	0.19	0.97	0.32	0	1	
k+h	5	0.66	0.98	0.79	0.79	0.99	0.88
k+h	10	0.85	0.98	**0.91**	0.85	0.99	**0.91**
k+h	20	0.9	0.97	**0.93**	0.91	0.96	**0.93***
k+h	30	0.92	0.96	**0.94***	0.93	0.68	0.79
k+h	40	0.93	0.79	0.85	0.94	0.69	0.8
k+h	50	0.95	0.63	0.76	0.96	0.41	0.57
k	5	0.79	0.99	0.88	0.77	0.92	0.84
k	10	0.87	0.98	**0.92**	0.88	0.98	**0.93***
k	20	0.92	0.97	**0.94***	0.92	0.88	**0.9**
k	30	0.94	0.89	**0.91**	0.93	0.78	0.85
k	40	0.94	0.89	**0.91**	0.93	0.7	0.8
k	50	0.96	0.52	0.67	0.96	0.51	0.67

tables. In constrast, LSH is know to be effective for large datasets, where the number of points in each bin is at least 100.

5. CONCLUSIONS AND FUTURE WORK

The paper presents a method to perform entity blocking for the task of entity matching in Web tables, by representing tables in a latent space using Neural Language Models. When compared to state of the art blocking methods table embeddings prove to be a promising solution. The current study has been performed on a set of 233 Web tables, manually annotated with entities matches (for a total of 50,072 instance correspondences). In future work, we plan to repeat the experiment on a larger dataset of tables from the Web Data Commons project by semi-automatically generating the gold standard, and to explore ways of meaningfully exploiting other value types, such as numbers and dates.

Acknowledgments

This work is part of the project DS4DM (Data Search for Data Mining) http://ds4dm.de, funded by the German Federal Ministry of Education and Research (BMBF).

6. REFERENCES

[1] C. S. Bhagavatula, T. Noraset, and D. Downey. Methods for exploring and mining tables on wikipedia. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, IDEA '13, pages 18–26. ACM, 2013.

[2] M. J. Cafarella, A. Halevy, and N. Khoussainova. Data Integration for the Relational Web. *Proc. VLDB Endow.*, 2:1090–1101, 2009.

[3] V. Christophides, V. Efthymiou, and K. Stefanidis. *Entity Resolution in the Web of Data*. MORGAN & CLAYPOOL PUBLISHERS, 2015.

[4] S. Duan, A. Fokoue, and O. Hassanzadeh. Instance-Based Matching of Large Ontologies Using Locality-Sensitive Hashing. pages 49–64, 2012.

[5] M. A. Hernandez and S. J. Stolfo. The merge/purge problem for large databases. In *CM SIGMOD international conference on Management of data, SIGMOD '95*. ACM, 1995.

[6] R. Isele, A. Jentzsch, and C. Bizer. Efficient Multidimensional Blocking for Link Discovery without losing Recall. (WebDB), 2011.

[7] J. Ji, J. Li, S. Yan, B. Zhang, and Q. Tian. Super-Bit Locality-Sensitive Hashing. *Advances in Neural Information Processing Systems2012*, pages 1–9.

[8] B. Kenig and A. Gal. MFIBlocks : An effective blocking algorithm for entity resolution. 38:908–910, 2013.

[9] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 2013.

[10] O. Lehmberg, D. Ritze, P. Ristoski, R. Meusel, H. Paulheim, and C. Bizer. The mannheim search join engine. *Web Semant.*, 35(P3):159–166, Dec. 2015.

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

[12] G. Papadakis, E. Ioannou, T. Palpanas, C. Niedere, and W. Nejdl. A Blocking Framework for Entity Resolution in Highly Heterogeneous Information Spaces. 25(12):2665–2682, 2013.

[13] V. Rastogi, N. Dalvi, and M. Garofalakis. Large-scale collective entity matching. *Proceedings of the VLDB Endowment*, 4(4):208–218, 2011.

[14] D. Ritze, O. Lehmberg, and C. Bizer. Matching html tables to dbpedia. In *Proc. of the 5th International Conference on Web Intelligence, Mining and Semantics*, WIMS '15, pages 10:1–10:6, 2015.

[15] P. Venetis, A. Halevy, J. Madhavan, M. Paşca, W. Shen, F. Wu, G. Miao, and C. Wu. Recovering Semantics of Tables on the Web. *Proc. of VLDB Endow.*, 4(9):528–538, 2011.

[16] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. InfoGather: Entity Augmentation and Attribute Discovery by Holistic Matching with Web Tables. In *Proc. of the 2012 ACM SIGMOD Int. Conf. on Management of Data*, pages 97–108, 2012.