T. Grust et al. (Hrsg.): Datenbanksysteme für Business, Technologie und Web (BTW 2019),
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 215

The Best of Both Worlds: Combining Hand-Tuned and Word-Embedding-Based Similarity Measures for Entity Resolution

Xiao Chen[1,2,3], Gabriel Campero Durand[1,2,3], Roman Zoun[1,2], David Broneske[1,2], Yang Li[1,2], Gunter Saake[1,2]

Abstract:

Recently word embedding has become a beneficial technique for diverse natural language processing tasks, especially after the successful introduction of several popular neural word embedding models, such as word2vec, GloVe, and FastText. Also entity resolution, i.e., the task of identifying digital records that refer to the same real-world entity, has been shown to benefit from word embedding. However, the use of word embeddings does not lead to a one-size-fits-all solution, because it cannot provide an accurate result for those values without any semantic meaning, such as numerical values. In this paper, we propose to use the combination of general word embedding with traditional hand-picked similarity measures for solving ER tasks, which aims to select the most suitable similarity measure for each attribute based on its property. We provide some guidelines on how to choose suitable similarity measures for different types of attributes and evaluate our proposed hybrid method on both synthetic and real datasets. Experiments show that a hybrid method reliant on correctly selecting required similarity measures can outperform the method of purely adopting traditional or word-embedding-based similarity measures.

Keywords: Entity resolution; Word embedding; Similarity measures; Learning-based entity resolution

1 Introduction

Entity resolution (ER) is the problem to identify digital records within one or among different data sources that refer to the same real-world entity. A typical solution is first to generate all candidate pairs, then calculate similarities between each pair [Ch12]. The high quality of ER results essentially relies on the correct selection of similarity measures, which is determined by domain experts based on an elaborate analysis of record attributes and their experiences. Except for the human efforts, traditional methods to calculate similarity between two records are mostly unaware of semantics, e.g., they determine how similar two

[1] Otto-von-Guericke-University of Magdeburg, Institute of Technical and Business Information Systems, Building 29, Universitätsplatz 2, 39106 Magdeburg, Germany
[2] E-Mail: {xiao.chen, campero.gabriel, roman.zoun, david.broneske, yang.li, saake}@ovgu.de
[3] Acknowledgements: This work was partially funded by the DFG [grant no.: SA 465/50-1] and China Scholarship Council [No. 201408080093]. Authors would also like to thank the PC and reviewers for the valuable feedback.

doi:10.18420/btw2019-14

strings look like in terms of edit distance. As a result, the accuracy may be limited for cases where words with similar semantics are expressed in different ways.

Recently, there have been several research papers that use word embedding for entity resolution, aiming to overcome the aforementioned limitations of traditional solutions [Eb18] [KAP18] [Mu18]. Word embedding maps words or phrases from the vocabulary to vectors of real numbers [Yo03] and makes it possible to extract semantic information conveniently and efficiently. It has become quite popular in various *Natural Language Processing (NLP)* tasks, after several neural word embedding models are introduced, such as word2vec [Mi13a], GloVe [PRS14] and FastText [Bo16]. Generally speaking, by using word embedding, attributes of records in ER are mapped to a vector space, no matter which kind of data it is, while capturing semantic similarities between attribute values of a record pair. Therefore, word embedding can be used to reduce the human efforts. In addition, the accuracy might also be further improved, since the approach captures semantic similarities between records.

However, adding an extra word embedding step to the similarity calculation phase will probably have negative effects on efficiency. For instance for attributes without semantic meanings, especially attributes with numerical values, it makes little sense to map them to a vector space using word embedding, because distances between them cannot be correctly managed due to the missing semantics. Still, numeric data play an important role for expressing records in financial business data.

Therefore, in this paper, we propose to combine traditional hand-tuned and popular word-embedding-based similarity calculations for ER, always choosing the most suitable similarity measures for each attribute in order to achieve the best accuracy for a given input dataset. Specifically speaking, with regard to attributes with a high appropriateness of word embedding techniques, word embedding is firstly applied to map the data to vectors using FastText pre-trained model, then cosine similarity is used to calculate similarities between record pairs. For other attributes, for which word embedding is not suitable, particularly numerical values, a tailored similarity function is used.

We summarize our contributions of this paper as follows:

- We propose a hybrid similarity calculation method for ER: the attribute-based selection between traditional hand-tuned similarity functions and word-embedding cosine similarity function to achieve a higher accuracy;
- We identify attributes that suit word embedding more than traditional similarity measures, which provides a guidance of choosing the most suitable similarity measure for ER;
- We design different combinations of adopted similarity functions and evaluate them on both real and synthetic datasets. Our results show that the hybrid solution can outperform other solutions which purely use traditional or word-embedding-based

similarity calculations, when the attributes of a dataset contain both semantic and non-semantic attributes.

The rest of this paper is organized as follows: We present related work in Section 2. Then we introduce our hybrid method in Section 3 and show the evaluation result in Section 4. At last, we conclude our work in Section 5.

2 Related Work

Entity resolution: Most of the ER research is pair-based and shares a common ER process, as surveyed in [Ch12] and [EPIV07]. In recent years, along with the increase of input data, solutions for ER are also asked to be scalable, which facilitates using parallel computing for ER, Chen et al. give an overview and classification on the research of parallel ER [CSG18].
Word embedding for entity resolution: As aforementioned, recent research has considered applying word embedding for ER. There are two main research questions of employing word embedding for ER. One is which embedding granularity to use for ER; the other one is how to get the vector of each attribute of a tuple after each word or sub-word has been embedded. Ebraheem et al. adopt word-level embedding using the GloVe pre-trained dictionary [PRS14] and propose two methods to get the vector representation of an attribute: an averaging method and an RNN-based method with LSTMs. Then a representation of a tuple is obtained by concatenating the vectors of all its attributes [Eb18]. Kooli et al. employ N-gram-level embedding using the the Fasttext library and then concatenate all vectors of all subwords [KAP18]. N-gram-level embedding should provide more accurate result when there is a large proportion of infrequent words in the input dataset [Mu18]. For ER tasks, data is often dirty and contains many infrequent words, therefore, in our work, we also use N-gram-level embedding. Mudgal et al. study several possible embedding choices from both the granularity of the embedding and adopted model, and sketch a design space for deep learning solutions [Mu18].

3 A Hybrid Approach for Entity Resolution

In this section, we introduce our ER process using hybrid similarity calculation methods. As we explained above, the work in this paper focuses on accuracy. Hence we do not take blocking (indexing) techniques into consideration to our process. With this, we get the most exact results since by blocking, we might block away matches in the worst case.

Figure 1 shows the entire process. The process first asks the user whether there is an available property file, which means the user knows the data quite well and knows the best choices of similarity functions to compare each attribute (excluding identifiers). If this is the case, we directly load the property file and divide all attributes into different groups based on the similarity measures defined in the property file. Otherwise, a property file will be

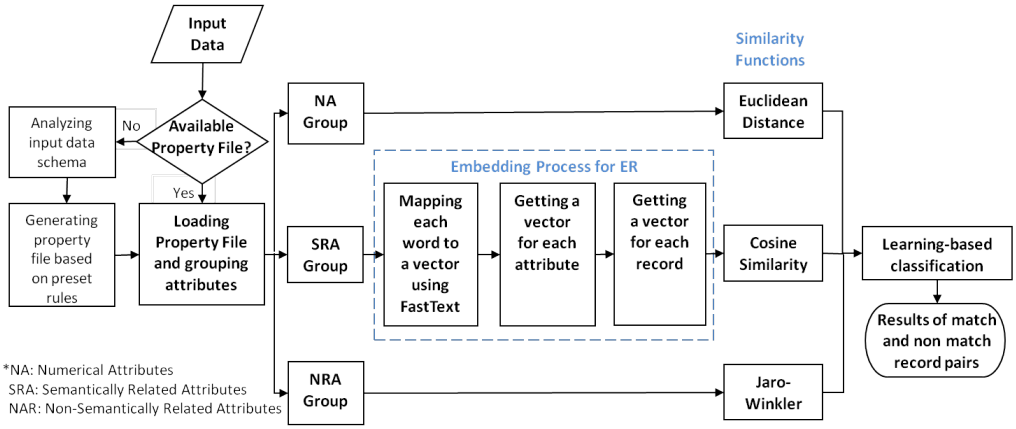

Fig. 1: Flowchart of our hybrid method

generated based on the input data schema and our preset rules. In order to achieve satisfied accuracy and provide guidance to choose suitable similarity measures, we classify common attributes into three groups, propose suitable similarity measures for each group and then the corresponding property file can be generated:

Numerical attributes (NA): Numerical attributes refer to those attributes, whose values can be compared, for example, the age. The Euclidean distance is used to provide a satisfactory accuracy for numerical attributes. However, not all numbers belong to this group. In contrast to numerical attributes, the other type of numbers are named as numerical strings, which act like strings and their values cannot be compared to be bigger or smaller. For example, telephone numbers or postcodes are considered as numerical strings. All numerical strings belong to the next group: non-semantically related attributes.

Non-semantically related attributes (NRA): These kinds of attributes are those whose values are without semantic meaning (e.g., people names or numerical strings) and usually with a short length. In ER tasks, the different values of this type of attributes of a match pair are usually caused by typos and formats. Using word embedding for this case may consider those values with a big distance, which can lead to lower accuracy. Therefore, for this group, we propose to use Jaro-Winkler similarity functions instead of word embedding. Moreover, Jaro-Winkler has a higher speed than the other traditional functions, such as Levenshtein similarity.

Semantically related attributes (SRA): The last group is semantically related attributes, whose values are strings with various meanings (often long), multiple strings or even whole sentences. For this group, word-embedding plus cosine similarity approach is chosen to calculate similarity. Traditional syntactical-based similarity measures are not used for semantically related attributes, because they only check the edit distances between values of attributes, and are not able to realize the meaning or the distributional semantic behind, so that they would probably provide unacceptable results. In terms of the predefined rules, we can calculate the similarity of each corresponding attribute pairs (cf. Section 3.1). Afterwards, the classification step is performed to divide all pairs into match and non-match groups (cf. Section 3.2).

3.1 Attribute Similarity

In this subsection, we specify how the similarity of each pair of attributes is calculated by the preset rules.

NA and NRA Similarity. According to the two rules regarding NA and NRA, the similarity between two attribute values of records calculated straightforwardly with Jaro-Winkler similarity and Euclidean metric, because the granularity for them is the entire attribute, which is calculated with the following formula:

$$attrSim(r_1.attr, r_2.attr) = \begin{cases} Euclidean(r_1.attr, r_2.attr), & attr \in NA; \\ Jaro\_Winkler(r_1.attr, r_2.attr), & attr \in NRA. \end{cases} \quad (1)$$

SRA Similarity. Inspired by the work of [Bo16], we use FastText, an extension of the continuous skip-gram model [Mi13b] that is used to produce word embeddings (i.e., vectors), to obtain the similarity of semantically related attributes. The FastText model is chosen instead of word2vec and GloVe because use cases will range across diverse domains. Hence, we cannot guarantee coverage with only models on word-level.

Vector representation of words. The input of FastText is a text corpus. Given enough text data and contexts, FastText can achieve highly accurate meanings of the words appearing in the corpus and establishes a word's association with other words. The output is a set of vectors, that is, words in the corpus are transformed a vector representation in a semantic vector space. Moreover, FastText is capable of predicting the vector representation for words not occurring in the corpus, since it decomposes each word into a set of characters of n-grams. Hence, every word appearing in the attribute value can be converted into vector representation $\vec{w}$.

Vector representation of attributes. Considering that there may be two or more words in one attribute, we achieve the vector representations of attributes by computing the mean value of all the word vectors in the attribute:

$$\vec{attr} = \frac{\sum_{i=1}^{n} \vec{w_i}}{n} \quad (2)$$

Therein, n is the number of words in attributes. We compute the cosine similarity between two attribute vectors to evaluate their similarity. More formally, the similarity of SRA is represented by:

$$attrSim(r_1.attr, r_2.attr) = cosine(r_1.\vec{attr}, r_2.\vec{attr}), \quad attr \in SRA. \quad (3)$$

3.2 Learning-Based Classification

After the similarity for each attribute pair is calculated with their respective methods, a classification step is used to classify pairs to matches or non-matches. The simplest

approach is a threshold-based method, which compares the average similarity score with the preset threshold value. Those pairs whose average score is higher than the threshold are considered as matches, and vice versa. However, in order to get the average score, similarity scores for each attribute are firstly summed up, which loses detailed information contained in the separated attribute similarities [Ch18]. Therefore, in our ER process, we adopt a learning-based classification step to overcome this drawback. A learning-based classification method firstly trains a classifier on a training dataset with available match or non-match labels, then the trained classifier is used to classify pairs with match or non-match status. So far, there have been different classifiers proposed, how to select a suitable one is based on the input data properties. In Section 4, we will describe the three classifiers we employ for our experiments.

4 Evaluation

In this section, we show our designed similarity calculation methods and the evaluation of their F-measure on different real-world datasets and synthetic datasets. Next, we show the datasets used for our experiments, and represent designed combinations of similarity calculation methods, at last we describe the three classifiers we use for the evaluation.

Datasets: Table 1 shows the three datasets we used for our experiments.

The first dataset "DBLP-ACM Citation" include two parts, which are from the DBLP citation database and the ACM citation database, respectively. All of them have four attributes, including title, authors, venue and publication year. Therein, "title","venue" are considered as SRAs, while "authors" is NRA and "publication year" is NA.

The second dataset "Amazon-Google Products" includes two parts as well, which are from Amazon and Google. Both of them have four attributes, including id, name, description, manufacturer and price. Based on their properties, "name","description" and "manufacturer" are considered as SRAs, while "price" is NA. All above datasets are downloaded from [Le17], which are benchmark datasets often used for entity resolution.

The last dataset is synthetic and generated by a data generator called GECO [TVC13]. GECO consists of a GEnerator and a COrruptor, which is specifically designed for generating ER datasets. The generated dataset contains personal information with the following 13 attributes: gender, given-name, surname, postcode, city, sex, telephone-number, credit-card-number, income-normal, age-uniform, income, age, and blood-pressure. Therein, the last five attributes are considered as NAs, while "sex" and "gender" are SRAs and the other attributes are NRAs.

The first two real datasets are commonly-used benchmark datasets for ER. However, both of them only contain one numerical value, which we think the most useful attribute type to show differences of using diffferent combinations of similarity calculations. Therefore, we involve the generated dataset as well, which contains five attributes with numerical values.

Tab. 1: Datasets used in experiments

Datasets		#Pairs (#DS1&#DS2)	#Matches	#SRA	#NRA	#NA
Real Datasets	DBLP-ACM	6001104 (2616&2294)	2224	2	1	1
	Amazon-Google	4400264 (1364&3226)	1300	3	0	1
Synthetic Dataset	Persons	551250 (1050&1050)	96	2	6	5

Combinations of similarity calculation methods:

Traditional hand-crafted similarity functions only: For this scenario, we use Jaro Winkler similarity function to calculate the similarity of all string attributes, i.e., NRAs plus SRAs, and use Euclidean distance to calculate the similarity of all NAs.
Word embedding and cosine similarity based method only: For this scenario, we use word-embedding-based method for all attributes.
Our proposed hybrid method: We propose to use word embedding for SRAs, Jaro Winkler for NRAs and the Euclidean distance function for NAs.

Classifiers: We employ three straight-forward classifiers which are intended to represent general solutions for classification in an ER context.

Tree Boosting (XGBoost): XGBoost consists of an ensemble of regression trees that uses additive optimization [CG16]. Such approach starts with a low variance, high biased solution, and gradually reduces the bias by decreasing the sizes of neighborhoods. We employ the specific XGBoost model, which introduces fine-grained improvements penalizing individual trees, leading to a competitive classification performance.
Random forest classifier (RF): This is an approach based on multiple decision tree algorithms for achieving a higher accuracy. It is intended to reduce overfitting risks.
K-nearest neighbor classifier (KNN): This approach considers the k (in our case k=5) closest training examples for each record to decide on the class membership of the record.

4.1 Results and Discussion

Tab. 2: Evaluation Results with Different Classifiers (F-measure)

Combinations		XGBoost	RF	KNN
Generated Dataset	Traditional	100	100	88.46
	WordEmbedding	100	100	100
	Hybrid	100	100	58.54
DBLP-ACM	Traditional	97.04	97.70	95.17
	WordEmbedding	92.56	94.82	93.94
	Hybrid	93.69	94.28	89.31
Amazon-Google	Traditional	20.19	25.35	21.11
	WordEmbedding	19.10	31.09	24.10
	Hybrid	29.72	38.32	19.78

Table 2 shows the results of our evaluation (results are reported for a random split of 66/34% training/test data, each including respectively 66/34% of the existing match and non-matches). It shows the F-measure values by using the three aforementioned classifiers XGBoost, KNN, RF under our three designed combinations of similarity measures on the three datasets. Next, we will discuss the result of each dataset in detail.

Generated dataset: As can be seen, the F-measure reaches its optimal for the generated dataset, with word embedding solutions performing consistently well across classifiers. This is a slightly surprising finding, as most attributes can be labeled as non-semantic. We deem the goodness of embeddings to be partly dependent on the coverage of the FastText learned representations. The purpose of using the generated dataset is it contains several NAs, which is promising to show the accuracy difference between different approaches. However, the generated dataset still lacks unpredictability and complexity compared to a real dataset, so that all F-measures are very high and we cannot get valuable conclusions from it. We also observe that the results of KNN are affected by the existence of irrelevant features, whereby the similarity of items on one dimension leads to misclassifications for close neighbors. For its less effective configurations, the model outputs 6 and 21 false positives out of 30, reducing the F-measure on this dataset.

DBLP-ACM dataset: For the relatively clean and easy citation dataset, the F-measure is observed to be consistently higher than 95% with traditional approaches outperforming others. The word-embedding-based and the hybrid approach show lower results. By careful consideration of its attributes properties, we found that the "title" attribute is, although having long strings, actually not semantic related. A paper title normally only has one version and its name usually only differs due to possible typos. Under this situation, word-embedding-based methods may fail. Therefore, for those attributes with long strings, careful consideration is needed about whether an edit-distance-based or semantic-based similarity measure is more suitable for them.

Amazon-Google dataset: For the more complex product dataset (various descriptions may express the same semantic meaning) we observe that hybrid solutions outperform the other two pure solutions, and achieve the best results with XGBoost and RF. For KNN we also observe some problems with the existence of irrelevant dimensions, which increase the number of false negatives. Overall, embedding-based approaches outperform traditional solutions (for classifiers RF and KNN), or provide comparable result with XGBoost, which indicates the necessity to use word embedding for datasets with real semantic attributes. The hybrid approach performs the best with XGBoost and RF, which indicates that by carefully choosing word-embedding-based or traditional similarity measure according to attribute properties (mainly basing on non-semantic and semantic) a better accuracy can be achieved than using only one of them.

We should also note that the F-measure values we report for this challenging dataset are much lower than many published results. We attribute this to the fact that almost all previous research adopts either blocking or thresholding techniques to reduce the amount of

non-matching pairs, so that the training data set is much more balanced, helping to achieve a higher F-measure. For this paper, our purpose is solely to test the benefit of our proposed hybrid similarity calculation approach. Therefore, we avoided blocking and thresholding, in order to highlight the specific impact of the approaches.

To conclude our discussion, a word-embedding-based approach works predominately better for semantic attributes. For non-semantic attributes it may also be possible to achieve a F-measure which is comparable (i.e., as seen for the generated dataset) or worse (i.e., as seen for the DBLP-ACM dataset) than traditional similarity measures. However, for numerical values, word embedding is not recommended since a hybrid approach shows obviously better F-measure than only using word embedding (based on the result of Amazon-Google dataset with XGBoost and RF classifiers). Therefore, the safest way for similarity calculations of ER problems is to use word embedding for semantic attributes and to use traditional similarity measures for non-semantic attributes. Besides, interestingly, in our findings we note that the choice of classifier seems to be secondary to the choice of similarity measures, with simple classifiers being able to outperform more complex ones, provided they are given adequate (i.e., descriptive, distinctive) similarity measures as input. We expect this to be partly explainable from the simplicity of the datasets (for the first two cases) and from the improvements brought by semantics (in the third case).

5 Conclusion

In this short paper we propose to use a hybrid similarity calculation solution for ER tasks and provide a practical evaluation of three different combinations of similarity measures with general machine learning classifiers. We find that embeddings are generally useful, though they are not a silver bullet, and both hybrid and traditional approaches can achieve superior results. We find that similarity measures can have a greater impact than the choice of classifiers in the resulting goodness of an ER process.

In summary, by using a prototypical workflow without blocking or thresholding and with general classifiers, we show that the current use of word embeddings alongside traditional measures for entity resolution opens-up a bundle of promising choices for practitioners, without lending itself easily to a one-size-fits-all solution. We envision two core challenges: On the one hand, previous work [KR08] has shown that thresholding, with a given blocking solution, improves the learning process; similarly work in entity resolution with embeddings [Eb18] shows good results without quantifying the precise impact of blocking and thresholding. In future work we seek to extend our current study by considering this factor for the case of hybrid solutions. On the other hand, the search for the optimal balance between the similarity measures used becomes essential, as we show in our current study. Though some guidelines can be adopted in this task for specific cases, as we propose, future work is required to achieve a general method for combining the approaches.

References

[Bo16] Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T.: Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606, 2016.

[CG16] Chen, T.; Guestrin, C.: Xgboost: A scalable tree boosting system. In: SIGKDD. ACM, pp. 785–794, 2016.

[Ch12] Christen, P.: Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection. Springer Science & Business Media, 2012.

[Ch18] Chen, X.; Rapuru, K.; Durand, G.; Schallehn, E.; Gunter, S.: Performance Comparison of Three Spark-Based Implementations of Parallel Entity Resolution. In: DEXA-BDMICS. volume 903. Springer, pp. 76–87, 2018.

[CSG18] Chen, X.; Schallehn, E.; Gunter, S.: Cloud-Scale Entity Resolution: Current State and Open Challenges. OJBD, 4(1):30–51, 2018.

[Eb18] Ebraheem, M.; Thirumuruganathan, S.; Joty, S. R.; Ouzzani, M.; Tang, N.: Distributed Representations of Tuples for Entity Resolution. PVLDB, 11(11):1454–1467, 2018.

[EPIV07] Elmagarmid, A.; P. Ipeirotis, Panagiotis; Verykios, V.: Duplicate record detection: A survey. TKDE, 19(1):1–16, 2007.

[KAP18] Kooli, N.; Allesiardo, R.; Pigneul, E.: Deep Learning Based Approach for Entity Resolution in Databases. In: ACIIDS. Springer, pp. 3–12, 2018.

[KR08] Köpcke, H.; Rahm, E.: Training selection for tuning entity matching. In: QDB/MUD. pp. 3–12, 2008.

[Le17] Leipzig, Database Group: , Benchmark datasets for entity resolution, 2017. Downloaded on 27.11.2017.

[Mi13a] Mikolov, T.; Chen, K.; Corrado, G.; Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.

[Mi13b] Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. Curran Associates, 2013.

[Mu18] 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. In: SIGMOD. ACM, pp. 19–34, 2018.

[PRS14] Pennington, J.; R. Socher, Riand Manning, Christopher: Glove: Global vectors for word representation. In: EMNLP. pp. 1532–1543, 2014.

[TVC13] Tran, K.; Vatsalan, D.; Christen, P.: GeCo: An Online Personal Data Generator and Corruptor. In: CIKM. ACM, pp. 2473–2476, 2013.

[Yo03] Yoshua, B.; Réjean, D.; Pascal, V.; Christian, J.: A neural probabilistic language model. Journal of machine learning research, 3:1137–1155, 2003.