Knowledge Graph Completion with FAMER

Daniel Obraczka[0000−0002−0366−9872], Alieh Saeedi[0000−0002−1066−1959], and Erhard Rahm[0000−0002−2665−1114]

University of Leipzig, Germany
{obraczka,saeedi,rahm}@informatik.uni-leipzig.de

Abstract. We outline the use of the tool FAMER to address the schema and entity matching tasks for the DI2KG 2019 challenge. FAMER supports both the static and incremental matching and clustering of entities from multiple sources. To alleviate entity matching, we first identify matching properties in the provided datasets based on the similarity of property names and instance values. This approach utilizes the given training data to derive property matches from entity matches. For entity matching, we consider multiple configurations to determine entity similarities with the optional use of word embeddings.

Keywords: Entity Resolution · Knowledge Graph

1 Introduction

Knowledge graphs (KG) physically integrate numerous entities with their properties (attributes) and relationships as well as associated metadata about entity types and relationship types in a graph-like structure [8]. A product KG may thus contain a huge number of products of many types where the product types can also be organized in an ontological structure, e.g., to differentiate camera-related products into different kinds of cameras (DSLR, mirrorless, ...), camera parts (e.g. camera bodies, lenses, ...) and different kinds of camera accessories. The KG entities and relationships are typically integrated from numerous sources, such as other knowledge graphs, databases, web pages, documents etc. Integrating such sources implies a matching and fusion of equivalent entities and relations. The initial KG may be created from a single source (e.g., a pre-existing knowledge graph such as DBpedia or the product KG of a specific merchant) or a static integration of multiple sources. KG completion (or extension) refers to the incremental addition of new entities and relationships. The addition of new entities requires solving several challenging tasks:

1. *preprocessing* of new datasets for data profiling (e.g., to determine the cardinality and value ranges of properties) and data cleaning
2. determining the entity type *(classification) of new entities*

DI2KG 2019, August 5, 2019, Anchorage, Alaska.

3. *incremental schema matching* to match and group (cluster) properties of new entities with known properties in the KG
4. *incremental entity resolution* to match and cluster new entities with already known entities in the KG
5. fusion of newly matching entities
6. addition of relationships for new entities.

At the Univ. of Leipzig, we are developing a scalable framework for the end-to-end generation and maintenance of KGs building on our previous work on learning-based product matching [5] and parallel entity resolution [3]. The core of this framework is a new parallel tool called FAMER (FAst Multi-source Entity Resolution) for both static and incremental matching and clustering of entities from multiple sources [10]. FAMER first determines or updates a so-called similarity graph between entities of a certain type and then applies a clustering approach to determine or update clusters of matching entities. These clusters group matching entities from different sources and thus support both the fusion of matching entities as well the tracking of original entities (which is also helpful for a possible cluster repair).

We address both the schema and entity matching tasks of the DI2KG 2019 challenge for KG integration of product entities about cameras. We are not providing a full-blown schema (property) matching solution but focus on a simple approach to support entity matching on the most frequent properties. We also use FAMER for building a similarity graph on properties and to determine and incrementally update clusters of matching properties.

In the next section, we provide an overview about FAMER. We then describe how we address preprocessing and schema matching (Sec. 3) and entity matching (Sec. 4) for the DI2KG 2019 challenge. Obtained results are described in Sec. 5.

2 FAMER Overview

Figure 1 illustrates the main components of the FAMER framework for incremental matching. As outlined in [11][12][10], the framework consists of two major configurable phases (gray boxes) named *Linking* and *Clustering*. In the Linking phase, a similarity graph is generated so that similar entities are linked pairwise with each other. This phase starts with blocking on selected properties so that only entities of the same block need to be compared with each other. In the initial version of FAMER, pairwise matching is manually configured by specifying a combination of several property similarities that has to exceed a minimal similarity threshold. We have now also added support for learning-based linking configurations, e.g. using random forest classification, which utilizes training data of matching and non-matching entity pairs. Similar to [6], we also added support for word embeddings, e.g. using FastText, to replace the value of string (textual) properties by their embeddings for a possibly improved similarity computation. Depending on the method to determine potential matches, the edges in the similarity graph include a similarity score to indicate the match likelihood. The second part of FAMER uses the similarity graph to determine entity

clusters where a cluster groups all matching entities from the different input sources. Clustering can be based on different algorithms including the so-called CLIP approach favoring so-called strong inter-source links that connect maximally similar entity pairs from both sides [12].

FAMER is able to update the output result for new entities and new sources [13] as needed for KG completion. In this case, the input is a stream of new entities from known sources or from a new source plus the already determined entity clusters (stored in the KG) (Figure 1). Here, the *Linking* part focuses on the new entities and does not re-link among previous entities. The output of the linking is an updated similarity graph composed of existing clusters and the group of new entities and the newly created links (the light-blue colored group in Figure 1). The *Incremental Clustering/Fusion* part integrates the group of new entities into clusters. The updated clusters are fused in the *Fusion* component so that all entities are represented by a single entity called cluster representative.

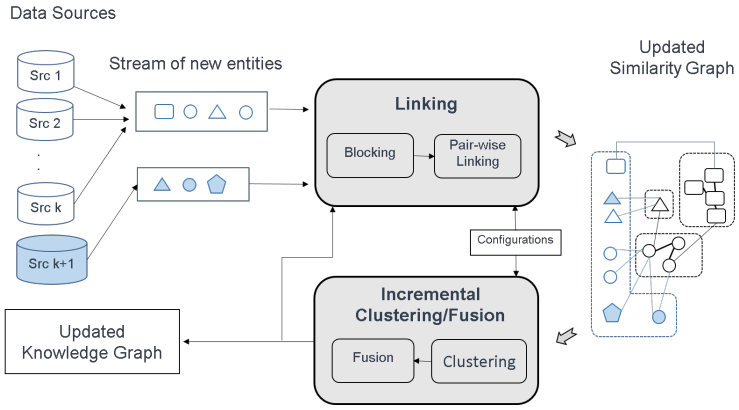

Fig. 1: Incremental entity resolution with FAMER

FAMER is implemented using Apache Flink so that the calculation of similarity graphs and the clustering approaches can be executed in parallel on clusters of variable size. For the implementation of the parallel clustering schemes we also use the Gelly library of Flink supporting a so-called vertex-centric programming of graph algorithms to iteratively execute a user-defined program in parallel over all vertices of a graph. The vertex functions are executed by a configurable number of worker nodes among which the graph data is partitioned, e.g., according to a hash partitioning on vertex ids.

3 Preprocessing and Schema Matching

To illustrate the data quality problems in the given dataset of the DI2KG challenge, we show in Table 1 two matching *Nikon* camera products from different

sources. We observe significant differences in the set of properties and property values. For example the first entity owns the property *features* while the second camera does neither contain this property nor the corresponding value (*Slimline*). This may happen even among entities of the same source. Moreover, the same property values are not represented similarly in different entities. For example in the first camera the property *camera resolution* with the value *16 Megapixels* is represented as *"approx resolution": "16MP"* for the second camera. Altogether, the challenge includes 24 sources with vastly heterogenuous schemas. For example, the source "ebay" has over 2000 properties some of which are likely duplicate properties such as "maximum shutter speed" and "max shutter speed".

Before we perform an incremental schema matching and entity matching we first perform *preprocessing* on the input dataset to derive some statistics and to perform data cleaning steps. In particular, we focus both entity and schema matching on the most frequent properties since infrequent properties are unlikely to be present for all matching pairs of entities so that their use is of limited value. For example the property *energy consumption per year* only occurs in one entity in the entire dataset and will therefore most likely not have a corresponding property in other sources and is therefore useless for entity resolution. For each source we therefore determine the k (≤ 10) most frequent properties.

We also perform data cleaning to harmonize property values to make similarity computations more meaningful. For example, we can see that different units are used for *weight* in different sources. Comparing values in ounces with values in grams would lead to a poor similarity value and we therefore transform both into the same unit. Further data cleaning procedures are performed, such as lowercasing strings and using canonical abbreviations.

Incremental schema matching Schema matching or schema alignment consists of determining which properties of different sources correspond with each other. There are a plethora of different approaches like e.g., instances-based or linguistic matchers that try to tackle this problem (see [9,7,2] for overviews). FAMER currently expects to be provided with already matched properties for entity resolution. For the DI2KG challenge, we however need to first align the properties before we can apply our entity resolution approach.

Our approach makes use of the provided training data for entity resolution task that includes a subset of the true matching entity pairs. While the provided training data contains example entities for all sources, entity matches are only available for a subset of source combinations. For example we are given entity matches for "canon-europe.com" and "price-hunt.com", but not for the combination of "canon-europe.com" and "ebay.com". However we have matches for "price-hunt.com" and "ebay.com". We can therefore first align the properties of "ebay.com" and "price-hunt.com" and then integrate "canon-europe.com" into this intermediary result using the given entity match between "canon-europe.com" and "price-hunt.com".

We therefore follow an incremental property clustering approach that starts with the pair of sources with the most matches in the training data and consider the further sources for property matching in the order of their number of matches.

Table 1: Example Raw Data

property	value
”35mm equivalent”	”25-300mm”
”<page title>”	”Nikon Coolpix S6800 Digital Camera (Black) | UK Digital Cameras”
”brand”	”Nikon”
”camera resolution”	”16 Megapixels”
”colour”	”Black”
”features”	”Slimline”
”hd video”	”Full HD (1080P)”
”lcd size”	”3.0””
”lens tele mm”	”300”
”lens wide mm”	”25”
”mpn”	”VNA520E1”
”optical zoom”	”23”
”optical zoom range”	”18x and higher”
”<page title>”	”Nikon Coolpix S6800 Price in India with Offers, Reviews & Full Specifications | PriceDekho.com”
”color”	”Black”
”amazon”	”Infibeam Ebay Homeshop18 Snapdeal Flipkart”
”digital zoom”	”4x”
”bangalore”	”Hyderabad Chennai Mumbai Delhi Pune”
”approx resolution”	”16 MP”
”external memory”	”Yes”
”face detection”	”NA”
”gps”	”NA”
”hdmi”	”NA”
”maximum shutter speed”	”1/2000 sec”
”metering”	”NA”
”minimum shutter speed”	”1 sec”
”optical zoom”	”18x”
”screen size”	”3 Inches”
”usb”	”Yes”
”video display resolution”	”NA”
”wifi”	”Yes; Wi-Fi 802.11 b/g/n”

For each source s we thus use the training data to count the number of entities that have s as provenance. We will refer to this as the *provCount* of s. Assuming that the source with the highest *provCount* has already been integrated into the KG, we start property matching with the source that has the second highest *provCount* and continue with the further sources in descending order of their *provCount* until all sources are processed.

Each incremental step consists of the following procedure:

1. Categorize properties by value range
2. Calculate property similarities by computing the weighted arithmetic mean of property name similarity and aggregated property value similarity
3. Update similarity graph
4. Cluster properties.

To avoid comparing apples with oranges all properties are first categorized by looking at the property value range. Possible categories are for example "string", "number" or "boolean". Looking at Table 1 for example the properties *optical zoom* and *color* clearly belong to different categories since the former mainly has number values and the latter consists of strings.

In the next step we calculate a combined similarity between properties of a new source and already considered properties of previous sources of the same category. The similarity between two properties is based on the similarity of property names and the aggregated similarity of all property values. The property values are derived from all relevant matches for the considered sources from the training data.

The calculated similarites are used to build and update a similarity graph consisting of the properties as vertices and the similarities as edges. This graph is given to FAMER's *clustering* module to determine new property clusters. This is iteratively done until no more sources are left to integrate. The resulting property clusters can now be used in the entity resolution step by fusing all members of a cluster to a new property.

4 Entity Resolution

FAMER assumes the knowledge of matching properties for both blocking and pair-wise linking. We therefore use the schema matching result and data cleaning for the most frequent properties to harmonize the entities before entity resolution. Table 2 indicates the improved data of Table 1 after preprocessing and property alignment. As illustrated we consider only a subset of the properties and both the property names and some property values have been harmonized.

FAMER provides many options to perform entity resolution for the prepared dataset and we aim at a comparative evaluation of several configurations. In particular, we can apply a batch-like (static) matching and clustering for all (24) sources at once or we can apply an incremental approach that iteratively adds and matches one source after the other. We decided to compare a batch-like approach, which we will denote as *1step*, and an alternative approach dubbed

Table 2: Example Data after Preprocessing and Property Alignment

property	value
page title	nikon coolpix s6800 digital black
manufacturer	nikon
resolution	16 mp
color	black
optical zoom	18x
screen size	3.0 inch
page title	nikon coolpix s6800
resolution	16 mp
color	black
optical zoom	18x
digital zoom	4x
screen size	3.0 inch

2step, in which we first deduplicate each source independently, fuse duplicate entities and then perform matching and clustering on the deduplicated sources.

In both cases blocking is done on the *manufacturer* property that is needed for a sufficiently low runtime. The camera products lacking the value of *manufacturer* form a special block and are matched with all other entities.

The most promising linking configuration used the following weighted similarity:

$$sim(e_1, e_2) = \omega_1 * productSim(e_1, e_2) + \omega_2 * JaroWinkler(e_1, e_2),$$

where ω_i are weights. The similarity $productSim$ is 0 or 1 depending on whether the product codes of the entities e_1 and e_2 match. The product codes are extracted from the *page title* attribute. Finally, $JaroWinkler$ is the jaro-winkler similarity performed on the concatenation of all respective properties of the entities except the page title.

The third approach we submitted utilized machine learning. We used the provided training data as input to Magellan's [4] XGBoost [1] implementation. As before we used the first 2 letters of manufacturers. Negative training examples were created by taking the most dissimilar entities in a block. Since Magellan is only able to perform pairwise matching we ran this approach for all possible data source pairs, where training data was available. The trained classifiers were then used to classify unseen entity pairs and the resulting classifier probabilities were used to create a similarity graph of all sources. Finally, FAMERs clustering module was used on this similarity graph.

Measure	Pairs	Target
Fmeasure	0.73	0.32
Precision	0.96	0.60
Recall	0.59	0.22

Table 3: Performance of schema matching for correctly pairing properties and clustering to the right target attribute

5 Results

In this section we will describe the performance of our approaches on the tasks *schema matching* and *entity matching* of the DI2KG challenge 2019. We present the evaluation of our results at the time of our submission, as well as the results obtained from the workshop organizers. Unfortunately, we could not directly use the golden truth for a comprehensive evaluation but had to rely on the results determined by the workshop organizers for one schema matching approach and three entity resolution approaches.

5.1 Schema Matching

For the schema matching task the challenge organizers provided us with the results for our approach shown in Table 3.

Since we were only concerned with clustering the most frequent properties, the evaluation was done with regards to two different aspects. On the one hand only the correct matching of attribute pairs is considered. In this regard our approach achieves a high precision of 0.96. The schema matching challenge consisted of matching source attributes to target attributes, which can be seen as attributes of the integrated schema. In this regard our approach performed worse. The main reason for this discrepancy lies in the fact that we were more concerned with correctly clustering source attributes, than finding the corresponding target attributes, because this seemed more relevant for the following entity resolution task. Each source attribute belonging to a source attribute cluster was therefore assumed to claim the union of all target attributes claimed by the other cluster members. This obviously disregards some intricacies of this task.

In general we can see that even for the most frequent properties schema matching is a very difficult task, due to the heterogeneity of these datasets. Attributes might have the same attribute name, but contain different information and are not to be matched. E.g. the attribute *resolution* in "www.priceme.co.nz" contains a technical description about the resolution, while in all other datasets this attribute contains the number of megapixel of the camera. Another problem lies in distinguishing attributes with similar value ranges. For example properties that have a value range that simply consists of numbers are very similar to each other.

The obtained results are not yet of sufficient quality indicating that inferring property matches from given entity matches is not as effective for the given

dataset as we had hoped for. To obtain better and more complete results for all properties we therefore need a more general solution for property matching, e.g. with a more comprehensive use of instance data.

5.2 Entity Matching

As described in Section 4, we submitted results of three different entity resolution approaches. While we initially also wished to employ word embeddings in these methods, in initial tests this technique did not prove as promising for the given dataset. The first two approaches consist of manually created configurations of our system, while the third utilized machine learning. For the weighted similarity, used in the first two approaches, the best weights were determined to be $\omega_1 = 0.6$ and $\omega_2 = 0.4$. The results are presented in Table 4. Before submission we created

Measure	1step	2step	ML(train)	ML(test)
Fmeasure	0.91	0.88	0.59	0.60
Precision	0.99	0.98	0.77	0.77
Recall	0.84	0.79	0.48	0.50

(a) Training Data

Measure	1step	2step	ML
Fmeasure	0.64	0.56	0.002
Precision	0.78	0.59	0.06
Recall	0.54	0.54	0.001

(b) Golden Truth

Table 4: Performance of ER approaches on training data and golden truth

a test dataset to avoid only evaluating the machine learning approach on the data we trained on. To obtain this test data, the entities with the most similar page titles in a block where regarded as true matches, and the most dissimilar entities where regarded as non-matches. At the time of submission we already observed that our manually created configurations were superior to the machine learning approach. We attribute this to the low number of training examples (especially per source-pair). The conference organizers informed us that our first two approaches enabled them to augment their golden truth with roughly 800 new entities that were previously not identified as matching indicating that the considered golden truth is not yet in a perfect state. We can see a huge difference between the performance on the training data set and the larger golden truth. This might indicate that the training data generally contains simpler examples, or our methods overfit to the training data. The bad performance of the machine learning approach on the whole golden truth is not explainable at this point and might be due to some error. Unfortunately, a more detailed analysis of this issue was not yet possible due to the unavailability of the golden truth for us.

Our *1step* method outperformed the *2step* method. We assume, deduplicating each source and fusing detected duplicate entities in one entity, may create false links in the 2nd step between the wrongly fused entities and other entities from other sources explaining the relatively low precision for the golden truth (Table 4b). The more detailed comparison of 1-step vs. 2-step approaches is another topic for future study.

6 Conclusion

We have shown that the FAMER tool could reasonably well solve the entity resolution task of the challenging 2019 DI2KG dataset. While there is still room for improvement, our approach determined matches that helped the conference organizers to enhance the golden truth (which thus may be more a silver truth). We could also provide a reasonable solution for (simplified) property matching, but more effort is necessary to achieve a full-fledged solution. Future work will also investigate how to improve accuracy on sources containing duplicates, and the integration and optimal use of of machine learning approaches in our system.

Acknowledgments

This work is partially funded by the German Federal Ministry of Education and Research under grant `BMBF 01IS18026B`. Some computations have been done with resources of Leipzig University Computing Centre.

References

1. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proc. ACM SIGKDD Conf. pp. 785–794 (2016). https://doi.org/10.1145/2939672.2939785
2. Euzenat, J., Shvaiko, P.: Ontology matching. Springer, Heidelberg (DE), 2nd edn. (2013)
3. Kolb, L., Thor, A., Rahm, E.: Dedoop: Efficient deduplication with Hadoop. PVLDB **5**(12) (2012)
4. Konda, P., Das, S., C., P.S.G., Doan, A., Ardalan, A., Ballard, J.R., Li, H., Panahi, F., Zhang, H., Naughton, J., Prasad, S., Krishnan, G., Deep, R., Raghavendra, V.: Magellan: Toward building entity matching management systems over data science stacks. PVLDB **9**(13), 1581–1584 (Sep 2016). https://doi.org/10.14778/3007263.3007314
5. Köpcke, H., Thor, A., Thomas, S., Rahm, E.: Tailoring entity resolution for matching product offers. In: Proc. EDBT (2012)
6. 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: Proc. ACM SIGMOD conf. pp. 19–34 (2018)
7. Rahm, E.: Towards large-scale schema and ontology matching. In: Schema matching and mapping, pp. 3–27. Springer (2011)
8. Rahm, E.: The case for holistic data integration. In: Proc. ADBIS. pp. 11–27. Springer (2016)
9. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. The VLDB Journal **10**(4), 334–350 (Dec 2001). https://doi.org/10.1007/s007780100057
10. Saeedi, A., Nentwig, M., Peukert, E., Rahm, E.: Scalable matching and clustering of entities with famer. Complex Systems Informatics and Modeling Quarterly (16), 61–83 (2018)
11. Saeedi, A., Peukert, E., Rahm, E.: Comparative evaluation of distributed clustering schemes for multi-source entity resolution. In: Proc. ADBIS. pp. 278–293. Springer (2017)

12. Saeedi, A., Peukert, E., Rahm, E.: Using link features for entity clustering in knowledge graphs. In: Proc. ESWC. pp. 576–592. Springer (2018)
13. Saeedi, A., Peukert, E., Rahm, E.: Incremental multi-source entity resolution with famer. p. submitted for publication (2019)