Exploiting Transitivity for Entity Matching

Jurian Baas(✉), Mehdi M. Dastani, and Ad J. Feelders

Utrecht University, Heidelberglaan 8, 3584 CS Utrecht, Netherlands
{j.baas,m.m.dastani,a.j.feelders}@uu.nl

Abstract. The goal of entity matching in knowledge graphs is to identify sets of entities that refer to the same real-world object. Methods for entity matching in knowledge graphs, however, produce a collection of pairs of entities claimed to be duplicates. This collection that represents the sameAs relation may fail to satisfy some of its structural properties such as transitivity. We show that an ad-hoc enforcement of transitivity on the set of identified entity pairs may decrease precision. We therefore propose a methodology that starts with a given similarity measure, generates a set of entity pairs, and applies cluster editing to enforce transitivity, leading to overall improved performance.

1 Introduction

Many datasets use different identifies to refer to the same real life entities, or may contain duplicates themselves. Automated methods for identifying and linking duplicate entities, also known as entity matching, in the knowledge graphs are necessary. A considerable difficulty with entity matching is that the total number of possible entity pairs is much larger than the number of actual (duplicate) entity pairs, also known as the problem of skewness [1,10]. This extreme skewness can cause false positive results to overwhelm the true positives, even for highly accurate classifiers. This has caused many other works to use ranking techniques, and their associated metrics, to sort the possible entity pairs with some similarity measure, where duplicate entity pairs are expected to appear on top of the ranking [4,8,9,11]. Other works, such as Saeedi et al. [7], perform blocking in the first stages to reduce the number of pairs that are evaluated. Furthermore, Raad et al. [5] start with a set of sameAs relations, and use a community detection algorithm to associate an error degree for each sameAs relation. They show that when only taking sameAs relations with an error degree ≤ 0.4, they achieve 100% accuracy within their random sample.

The identified set of pairs are generally required to satisfy some structural properties, in particular transitivity. However, taking the transitive closure of the entity pairs identified by entity matching techniques may not work as this may possibly conclude many spurious entity pairs.

We propose the application of cluster editing for entity matching and set up a number of experiments to evaluate our proposal. We show that compared

R. Verborgh et al. (Eds.): ESWC 2021 Satellite Events, LNCS 12739, pp. 109–114, 2021.
https://doi.org/10.1007/978-3-030-80418-3_20

to an ad-hoc enforcement of transitive closure on identified pairs, our approach always results fewer distinct entity pairs (i.e. they have a higher precision) while retaining duplicate entity pairs (i.e. recall is not lowered).

The experiments are performed on semi-synthetic datasets that are generated by introducing duplicates in an existing dataset in a controlled manner. This results in a range of different cluster distributions, where we measure the effects of the number of clusters and different cluster sizes.

2 Applying Cluster Editing on Matched Entities

An overview of our overall method is given in Fig. 1. We start with an embedding of a set of entities E, some of which may be duplicates, and use Euclidean distance to measure their proximity (panel A of Fig. 1). For each entity $e_i \in E$, we make k candidate pairs (e_i, e_j), where e_j is the k-nearest neighbor of e_i, thereby addressing skewness by ruling out the vast majority of pairs.

The dotted lines in panel B illustrate the candidate pairs for $k = 1$. Moreover, we assume that a (small) subset of these candidate pairs is labeled by a domain expert (blue lines in panel B). The labeled pairs are used to train a probabilistic classifier. This classifier is used to determine, for each candidate pair (e_i, e_j), the fitted probability p_{ij} that e_i and e_j are duplicates. Depending on the features used by the classifier, and its complexity, the fitted probabilities need not be proportional to the distance between entities. We do however assume that the features used by the classifier are symmetric so that $p_{ij} = p_{ji}$, and therefore we can indeed regard a pair of entities as unordered. We then use a cut-off value θ so that if $p_{ij} > \theta$, then e_i and e_j are predicted to be duplicates (panel C). This "raw outcome" of the pairwise classifier, which represents te sameAs relation, may however violate the expected transitivity constraint. Obviously, an ad-hoc application of transitive closure to the links predicted by the classifier never removes any links, but can only add new links (panel G). This may result many spurious entity pairs. A more principled method to restore transitivity is to use the cluster editing technique [3]. Here, we compute a weight $w(i, j) = \log(\frac{p_{ij}}{1-p_{ij}}) - \log(\frac{\theta}{1-\theta})$ for each pair of entities (e_i, e_j) within the same connected component (regardless of whether it is a candidate pair or not), such that $w(i, j)$ is positive if $p_{ij} > \theta$, and negative otherwise (panel D). If $w(i, j)$ is positive (negative), a link between i and j is provisionally assumed to be present (absent). The resulting set of links may however again violate the transitivity constraint. Cluster editing is used to restore transitivity by adding and/or removing links in such a way that the total score $\sum_{(i,j)} w(i, j)x_{ij}$ is maximized, where $x_{ij} = 1$ if a link between i and j is present in the solution, and $x_{ij} = 0$ otherwise (panels E and F). Finally, more information about our method is available at [2].

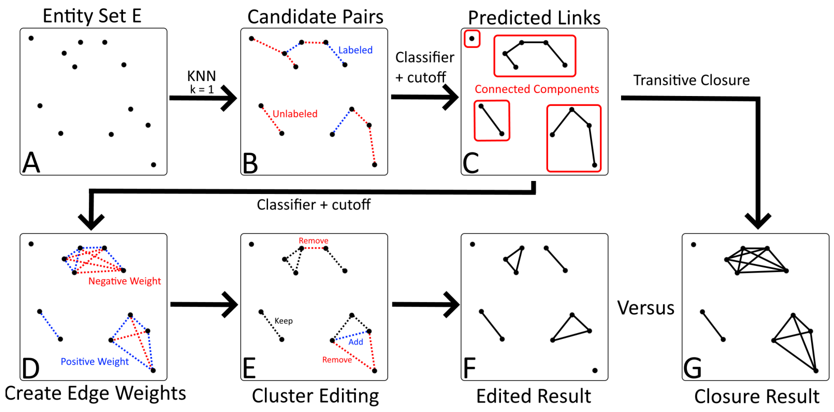


Fig. 1. An overview of the entity matching process.

3 Results

The data source we use is an RDF version of Ecartico[1], a comprehensive collection of biographical data about, among others, painters, engravers and book sellers. These people worked in the Low Countries at the time of the 16^{th} and 17^{th} century. This data source is actively curated so we can be sure there are no duplicate records of persons. From the Ecartico graph, we have constructed a new graph containing all `schema:Person` entities that have values for the properties – `schema:name`, – `schema:workLocation`, and – `schema:hasOccupation`. For each of these entities, we also copy the property-value pairs for – `schema:birthDate`, – `schema:deathDate`, – `schema:birthPlace`, and – `schema:deathPlace` when they are present. This resulted in a graph with, including `rdf:type` for the class `schema:Person`, eight properties, all centered around that one RDF class.

Then we introduce duplicates by uniformly sampling a percentage of entities and altering their respective URI's. For example, when sampling entities, an entity with URI `www.data.uu.nl/1234` will be modified to `www.data.uu.nl/1234/3`. We create three cluster distributions $\mathcal{D}_{10}$, $\mathcal{D}_{25}$ and $\mathcal{D}_{50}$, which are determined by the percentage of entities that is sampled, which can be either 10, 25 or 50%. Figure 2 shows the resulting distributions of clusters, where, for instance with 10%, we expect most entities to be in a cluster of size 1, meaning they were not duplicated. When modifying 50%, of entities, however, this results in most entities being duplicated at least once, and the largest proportion being duplicated twice. In this case non-duplicate entities are in the minority, making the entity matching problem considerably harder.

Table 1 shows the results of our experiments. We denote the application of transitive closure with the subscript TC and the application of cluster editing with the subscript CE. For every value of $\theta \in (0, 1)$ (in steps of 0.01) we generate an associated $F_{\frac{1}{2}}$-score, as it is our experience that a low precision has a larger

[1] http://www.vondel.humanities.uva.nl/ecartico.

negative impact (than low recall) on the performance of downstream systems such as SPARQL engines. The $F\frac{1}{2}$-score weights precision twice as heavy as recall. We average the $F\frac{1}{2}$-score for all values of θ (100 values between 0 and 1) to denote the performance of a given combination of cluster distribution, classifier and features. We experimented with a logistic regression (LR) and support vector machine (SVM) classifier. All were trained using cosine similarity as the sole feature. Furthermore, we used just 100 pairs to train each classifier, limiting the burden on the domain expert as much as possible.

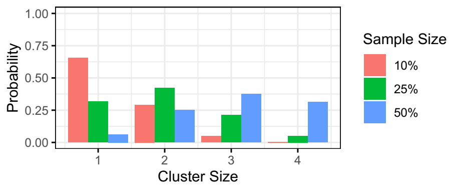
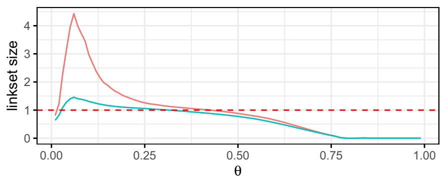
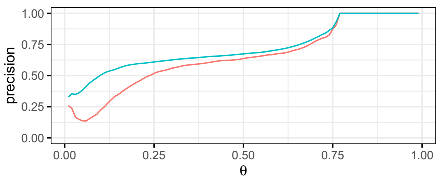


Fig. 2. Generated probability distributions of entity clusters of size 1 to 4 in the synthetic data. The values of a color sum to one.



(a) $\mathcal{D}_{50}$, LR - Cosine Similarity



(b) $\mathcal{D}_{50}$, LR - Cosine Similarity

Fig. 3. Left: relative number of pairs predicted vs actual number (red dotted line). Right: precision for transitive closure (red lines) and edited closure (blue lines). (Color figure online)

In all cases we observe that the application of cluster editing improves the resulting set of duplicates over the application of transitive closure. Furthermore, the optimal value for θ is in most cases reduced when cluster editing is applied, suggesting that a more lenient cutoff can be used, while at the same time improving performance. Furthermore, Fig. 3a shows how closure of the entity pair set tend to overestimate the number of duplicate pairs for low values of θ, where the number of predicted pairs is more than 4 times the number of actual pairs. Finally, Fig. 3b shows that the cluster editing method consistently outperforms transitive closure in precision.

Table 1. A comparison of the mean and maximum $F_{\frac{1}{2}}$-scores, and associated value for θ for transitive closure (TC) and cluster editing (CE), per classifier and cluster distributions ($\mathcal{D}_{10}, \mathcal{D}_{25}, \mathcal{D}_{50}$)

Classifier	Dataset	θ_{TC}	θ_{CE}	Mean$_{TC}$	Mean$_{CE}$	Max$_{TC}$	Max$_{CE}$
LR	$\mathcal{D}_{10}$	0.43	0.51	0.46	0.50	0.69	0.70
	$\mathcal{D}_{25}$	0.40	0.35	0.37	0.41	0.62	0.64
	$\mathcal{D}_{50}$	0.51	0.41	0.38	0.44	0.62	0.64
SVM	$\mathcal{D}_{10}$	0.72	0.83	0.61	0.64	0.68	0.70
	$\mathcal{D}_{25}$	0.70	0.66	0.43	0.51	0.62	0.64
	$\mathcal{D}_{50}$	0.78	0.67	0.45	0.54	0.62	0.64

4 Conclusion and Future Work

In practice, entity matching methods are applied and the resulting entity pairs are used by, e.g., a reasoner in a SPARQL engine, which applies the transitive closure. This may introduce many spurious links, potentially creating large clusters of unrelated entities. We propose to apply cluster editing to create a set of links that is closed under transitivity and show that the application of cluster editing, compared to the transitive closure, always results in a set of duplicates that contains fewer distinct entity pairs (i.e. they have a higher precision) while retaining duplicate entity pairs (i.e. recall is not lowered). The NP-Hardness of cluster editing limits us to solving only relatively small connected components. There are, however, heuristic methods which enables larger components to be solved. These heuristic methods can effectively reduce the instance size of the problem and are fast in case a small number of edits is allowed [6]. Additionally, the Louvain community detection technique proposed by Raad et al. [5] for detecting erroneous links can be applied to the connected components formed by the candidate pairs. It would be interesting to see how it compares to other (heuristic) clustering techniques.

References

1. Al Hasan, M., Zaki, M.J.: A survey of link prediction in social networks. In: Aggarwal, C. (ed.) Social Network Data Analytics, pp. 243–275. Springer, Boston (2011). https://doi.org/10.1007/978-1-4419-8462-3_9
2. Baas, J., Dastani, M., Feelders, A.: Exploiting transitivity constraints for entity matching in knowledge graphs. arXiv:2104.12589 (2021)
3. Böcker, S., Baumbach, J.: Cluster editing. In: Bonizzoni, P., Brattka, V., Löwe, B. (eds.) CiE 2013. LNCS, vol. 7921, pp. 33–44. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39053-1_5
4. Chen, M., Tian, Y., Chang, K.W., Skiena, S., Zaniolo, C.: Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. arXiv preprint arXiv:1806.06478 (2018)

5. Raad, J., Beek, W., van Harmelen, F., Pernelle, N., Saïs, F.: Detecting erroneous identity links on the web using network metrics. In: Vrandečić, D. (ed.) ISWC 2018. LNCS, vol. 11136, pp. 391–407. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00671-6_23
6. Rahmann, S., Wittkop, T., Baumbach, J., Martin, M., Truss, A., Böcker, S.: Exact and heuristic algorithms for weighted cluster editing. In: Computational Systems Bioinformatics, vol. 6, pp. 391–401. World Scientific (2007)
7. Saeedi, A., Nentwig, M., Peukert, E., Rahm, E.: Scalable matching and clustering of entities with famer. Complex Syst. Inform. Model. Q. **16**, 61–83 (2018)
8. Sun, Z., Hu, W., Li, C.: Cross-lingual entity alignment via joint attribute-preserving embedding. In: d'Amato, C., Fernandez, M., Tamma, V., Lecue, F., Cudré-Mauroux, P., Sequeda, J., Lange, C., Heflin, J. (eds.) ISWC 2017. LNCS, vol. 10587, pp. 628–644. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68288-4_37
9. Trisedya, B.D., Qi, J., Zhang, R.: Entity alignment between knowledge graphs using attribute embeddings. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 297–304 (2019)
10. Weiss, G.M.: Mining with rarity: a unifying framework. ACM Sigkdd Explorations Newsletter **6**(1), 7–19 (2004)
11. Zhu, H., Xie, R., Liu, Z., Sun, M.: Iterative entity alignment via joint knowledge embeddings. IJCAI **17**, 4258–4264 (2017)