Ontology Matching Paper: Exact and Fuzzy Matching of Movies and Actors

Andrei Sili, Saranots Tzortzis

[1] andrei.sili@student.uva.nl
[2] sarantos_tzortzis@icloud.com
https://github.com/sarantinio/Semantic-Web-RDFs.git
May 2th, 2018

Abstract. This paper proposes a fuzzy property instance matcher that can be used when it is not clear which instances should be treated as identical. Our example uses the LinkedMovieDatabase and DBPedia. We try to infer identical instances from the Film class. We evaluate our measure using the provided and valid owl:sameAs links from LMDB, which we consider as ground truth. We find that by allowing for very small differences in the values of the inverse functional properties set of instances from different classes, one can improve the the recall of identical instances by roughly 27% while losing only 1% precision on this particular case.

1 Introduction

The aim of this paper is to explore ways of equate concepts from different name spaces using the similarity of their instances in the face of highly noisy data. A traditional approach to concept-matching using *owl:sameAs* properties will does not provide satisfactory results when applied to the Web of Data. This happens for a host of reasons relating to the noisiness of the recorded data. Links can be outdated, poorly encoded, simply wrong or missing altogether. Because of this, a naive approach will results a dramatically underestimating the similarity between two concepts. A possible solution to this problem is to first infer identity between instances using one or several properties that constitute an inverse functional set for that class of instances. If these can be matched to the set of inverse functional properties of an instance from another class, than identity can be inferred, and then similarity between 2 classes can be based upon this new identity relationship. The final ontology matching algorithm for the LMDB and DBPedia Film concepts can be visualized in Figure 1.

Throughout the paper, we use the running example of the *Film* class from LMDB and DBPedia. More specifically we want to find out how similar are the entities *LinkedMovieDataBase:Film* with the *DBPedia:Film*.

In this paper, we explore exact and fuzzy instance matching techniques, based on hand-engineered features between pairs of resources. As a disclaimer, obviously, this approach is not practical since these features would have to be engineered for every pair of concepts manually. A more generalizable approach

would have been to learn these features using some form of machine learning techniques, but this falls out of the scope of this project. However, given that this project has been plagued by the high number of deficiencies found in the LMDB data set, roughly half the time of this research project was spent on data engineering tasks.

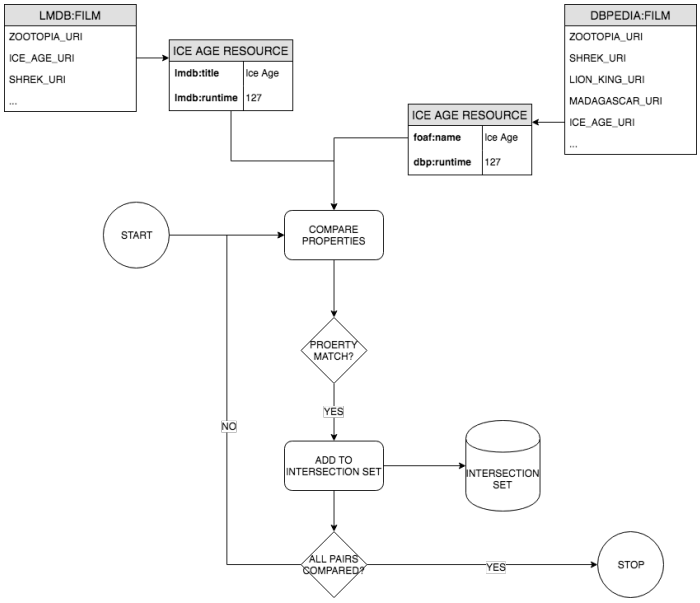


Fig. 1: Concept Matching Pipeline

2 Related Work

The problem of instance-based ontology matching has gained much interest recently due to the need to analyze the Web of Data, which is inherently heterogeneous. A notable first issue is defining what similarity is between concepts. We take a semantic view of this issue as described in [2], where similarity is defined by the number of overlapping instances. In this paper, the authors also define what evaluates to be an improved Jaccard measure, called CorrectedJaccard. It would have been an interesting to implement such a measure for our similarity scores, but the mechanics of this measure are not entirely clear from the reading so we decide to stick with the more general simpler Jaccard similarity.

A second issue, as hinted upon in the introduction, is related to matching instances across different datasets. This problem is described in detail in [1]. The authors design a system able to clean, validate and deduplicate relationships across datasets by analyzing the sub-graphs formed by entities, the main

contribution being that their work is generalizable to any pair of concepts, since no domain-knowledge is needed. Specifically, matching is done based on a score computed as a linear combination of different graph properties, with the weights being learned via a machine learning approach. We would like to implement a similar approach, but based on the properties of entities themselves, rather the properties of their graph representation. Our approach is much more simplistic and requires domain knowledge to map properties from different entities to each other. We also hand-tune the linear combination weights and leave automated learning approaches for future work.

3 Ontology Matcher: Method and System Description

An ontology matching algorithm compares the similarity between two entities based on some metric(s) of instance similarity. Both the similarity measure and the metric are subject to research with no unified consensus on what are the best approaches. As mentioned earlier, the similarity measure we use is the simple Jaccard similarity for convenience. Thus, we take a semantic view of similarity, where this is defined in terms of the number of instances that belong to both classes. Cross-dataset identical instances can be explicitly stated through *owl:sameAs* property or can be inferred from a set of matching instance properties.

The Jaccard similarity is a value between 0 and 1, from completely dissimilar to completely identical. It is not surprising that, given the heterogeneity of the Web of Data, entities are never completely identical as measured by the Jaccard similarity. However, approximate similarity can be inferred if the value is high enough. Unfortunately, there is no consensus on what a threshold should be, leaving an empirical choice as the only option.

To compute the Jaccard similarity, we need to determine identical instances. We achieve this in 3 ways. First, we rely on the explicit annotations of LMDB data, using the *owl:sameAs* property. This is taken as a ground truth measure. That is, any of the film instance in LMDB for which the corresponding DBPedia resource can be retrieved using the *owl:sameAs* link are treated as being identical. As we will see later, this leads to a host of issues, so we try more nuanced approaches. The second attempt we make is property-based instance matching. By choosing a set that can reasonably be though of as inverse functional from 2 entities, we can infer identity.

We attempt property matching in 2 ways. First, we opt for an exact match, meaning that each pair of properties needs to be completely identical. Obviously, this will produce a substantial amount of false negatives, so our final matcher gives each property pair a score of similarity (not to be confused with the similarity of the entity) and the linearly combines these scores into one. We then set a threshold on this value above which we consider instances identical. It should be noted that different types of properties require different similarity measures. Most notably, for numeric properties we use the most basic absolute difference

metric

$$AD(a, b) = |a - b|$$

where a and b are the values of the numeric properties. For string properties, we use the *Levenshtein distance*, which in plain English computes the number of string insertions, removals, and substitutions required to convert one string to the other

$$LD(s, t) = removals(s, t) + deletions(s, t) + substitutions(s, t)$$

where s is a source string and t is a target string. Note that this is a proper distance measure as it is symmetric, non-negative with 0 meaning equality of inputs, and satisfies the triangle inequality.

Before computing a final score, all values are normalized between 0 and 1, with the final score being calculated as a linear interpolation between the two values. Because everything is normalized, the linear interpolation also ranges between 0 and 1 and because small distances are what we are looking for, we will subtract the result from 1. So higher values of the $fuzzy_score$ means more similar instances.

$$fuzzy_score = 1 - (\alpha \cdot LD(s, t) + \beta \cdot AD(a, b))$$

4 Experimental Setup

4.1 Deriving a Gold Standard

To evaluate our property-based ontology matcher, we need to first derive some sort of ground truth about the film instances from LMDB and DBPedia. This is done using the *owl:sameAs* property. If and only if an LMDB film links to a valid DBPedia URI, do we consider the pair of instances to be identical. In this way, after the property-based matcher will predict an identity between 2 instances, we will benchmark this against LMDB links to calculate quality measures (e.g. accuracy, precision, recall).

4.2 Relevant Property Extraction

Both DBPedia and LMDB provide many properties for a film or an actor. Among those that might constitute an inverse functional set we enumerate the name of the film, the release date, and the duration of the movie. Our assumption is that having sufficient similarity on this set indicates an identical instance. However, data constraints force us to make concession it is area as well. For example, a majority of film resources do not have a release date. For this reason, we choose to match on movie title and run time.

4.3 Preprocessing the Data

Unfortunately, not all *LMDB:film* instances provide us with a *owl:sameAs* property for dbpedia, or some that they do the link is malformed in some way or another, or it is outdated. There are potential fixes around this using the redirect links from DBPedia to retrieve the right resource, but we do not treat this in our report and leave it for future initiatives to tackle. So we need first to gather only the intersection of the *dbpedia:film* instances and the *LMDB:film owl:sameAs* links in order to create a set that after it is matched, it can be evaluated as described.

We retrieve all the instances of *dbpedia:Film* and from the *LMDB:Film* we selected the *owl:sameAs* property to retrieve their dbpedia link. Out of the roughly $20,000$ movies in LMDB, the conjunction of these two sets now provide us with the dataset for which we have ground truth data, and measures around $9,000$ instances. For each movie in our final set, we retrieve the title and run time from both DBPedia and LMDB. Because again, some run time values are missing from LMDB, we drop these instances, resulting in a much reduced data set of $2,204$ instances. The Figure 2 illustrates the procedure to generate the dataset we need.

The last processing step is to create a Cartesian product of the $2,204$ instances from LMDB and DBPedia such that we now have a set of $2,204$ positive examples and $2,204 \cdot 2,204 - 2,204$ negative examples. With this at hand, we can now start matching properties using exact or fuzzy logic in order to infer instance identities.

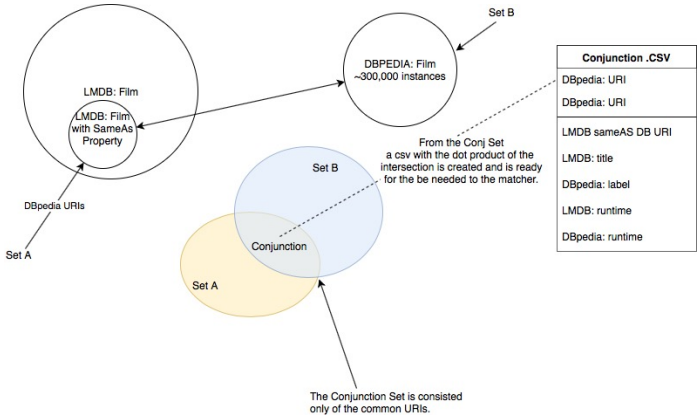


Fig. 2: Conjunction Set

4.4 Instance Identity

In order to decide weather two instances are the same their corresponding attributes should match. This is done either by exact or fuzzy matching as described in the previous sections. Not requiring an exact match will provide an algorithm that is more robust to noise, such as malformed strings, a disambiguating addendum to the title, or slightly different run times.

4.5 Evaluation

In order to evaluate the instance matching algorithm, standard quality metrics can be computer such as accuracy, or precision. All evaluations are based upon the ground truth in terms of the sameAs links as previously discussed.

5 Results and Discussion

Let us first motivate our approach by observing the Jaccard similarity measure between *LMDB:film* and *DBPedia:Film* using only the *owl:sameAs* links to identify identical instances. The results are presented in Table 1. Out of approximately $20,000$ movies, only about $11,000$ have links to DBPedia, and out of those only about $9,000$ are valid. Combined with the fact that DBPedia has many more movies than LMDB, this gives a infinitesimal score for similarity. While it seems reasonable that these classes are not identical, it would seem reasonable to assume that the LMDB class is a subclass of DBPedia. However, this is also not evidently clear as the number of instances in LMDB that do not have a valid correspondent in DBPedia, even only within the set of LMDB instance that have a specified link to begin with, is rather high. By these results, a more robust matcher is warranted.

	FILM	ACTOR
$LMDB \cap DBP$	9,282	1,354
$LMDB \cup DBP$	301,137	168,586
$LMDB \setminus DBP$	1243	560
$DBP \setminus LMDB$	290,612	166,672
$SIMILARITY$	0.03	0.01

Table 1: Similarity Scores. *Both similarity score are very low, indicating the counter intuitive result that the classes are distinct. Class subsumption cannot be inferred either given the relatively big difference between LMDB and DBPedia*

Our property-based fuzzy matcher outputs a score for each pair of resources from LMDB and DBPedia. First, we need to set the interpolation weights. Because run time on it's own is not a great indicator of uniqueness since many

movies can have the same run time, we put much less emphasis on it.By toying around with different values, we reach an empirical estimate of $\alpha = 0.8$ and $\beta = 0.2$. Naturally, we also need to define a threshold as a cut-off point. We use point estimator information about our score for the positive and negative examples to arrive at a threshold value of 0.99.

Statistical information can be viewed in Table 2. Most notably, the mean and median of the positive examples are most telling. While the mean is a bit lower, this is only because there are a few great outliers with score in the 0.3 0.4 range that drag the average down. This can be seen by looking at the median, which sits at a perfect 1. Inspecting some of the outliers is also interesting. They seem to be cause by 2 factors. On the one hand, there are some movies for which the title is recorded in different languages, for example the LMDB movie 'Love Can Seriously Damage Your Health' is recorded in DBPedia with the title 'El amor perjudica seriamente la salud'. On the other, there are movies that are recorded with a much longer title, which is still similar to the human eye.

	POSITIVE EXAMPLES	NEGATIVE EXAMPLES
MEAN	0.99	0.88
MEDIAN	1.00	0.89

Table 2: Statistics of fuzzy score. *Mean and median values of the fuzzy score grouped by positive and negative examples. In this context, positive examples are pairs of resources that match based on the owl:sameAs links.*

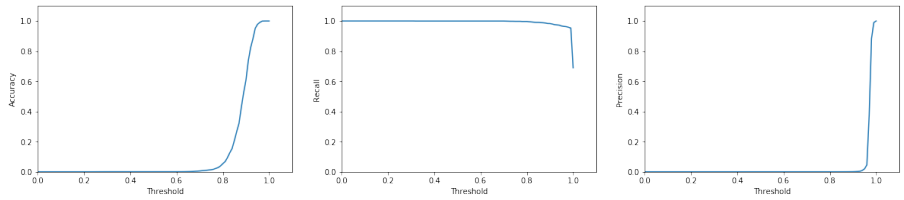
At this point, we can compute quality measures such as accuracy, precision and recall. At 0.99 threshold value, we observe quite satisfactory results. These and other results can be seen in Table 3 and are visually represented in Figure 3. It seems that recall remains virtually unchanged over the range $0 - 0.8$, which makes sense since almost the entire dataset falls in the range $0.8 - 1.0$. However, it falls drastically to around 0.7 once the threshold reaches the $0.95 - 1.0$ range. On the other hand, precision and accuracy increase drastically when the threshold reaches above 0.9 and keep a steady and steep increase until it plateaus close to the perfect value of 1.0.

	1.0000	0.9900	0.9000	0.7500
ACCURACY	0.9990	0.9999	0.0006	0.0138
PRECISION	1.0000	0.9898	0.0004	0.0000
RECALL	0.6800	0.9531	0.9995	0.0997

Table 3: Quality Measures. *The most important observation is that by allowing for small discrepancies in inverse functional properties, we can improve recall dramatically while preserving precision almost entirely.*

The main take-away that should be observed from this experiment is the following. By allowing for small discrepancies in the inverse functional property set, we can dramatically improve recall, roughly by 28% while only losing only 1% precision. As far as we can tell, this is a significant result that is worth mentioning.

Of course, we did not use an evaluation or a test set for this experiment because of prohibiting time costs. However, a natural next step would be do deploy this algorithm on the entire *LMDB:Film* graph and obtain identical instances, and concept similarity scores. This would be a monumental task, because all pairs need to be evaluated by the algorithm first and then curated by human beings that can attest to the validity of the identity relationship. One way of speeding up the human evaluation process would be to eliminate all other potential identities from the to-be-curated list once one relationship has been identified between the LMDB resource and the DBPedia one, assuming that the DBPedia resource does not point to identical resource with itself.



(a) Accuracy (b) Recall (c) Precision

Fig. 3: Quality Measure for Property-Based Fuzzy Instance Matcher. *The threshold ranges from 0 to 1. Values for accuracy, precision, and recall remain roughly unchanged for the better part but show dramatic changes in the upper ranges of the threshold.*

6 Summary and Conclusion

In this project we were called to implement an ontology matcher. The datasets included for this project were these of the previous assignment of the course. From the one hand we have the *DBpedia:Film* and from the other the LMDB: film. We restricted and hence limited the set of the instances from the LMDB: Film so we only included the ones with an *owl:sameAs* property linking to the dbpedia: Film URI. The combined set that our custom matcher will try to match on is the intersection of the two sets. The property *owl:sameAs* worked as ground truth for the matcher. The matcher compared the LMDB:title with the *DBPedia:title* and so did for their perspective run times. The matcher used a fuzzy match method, the initial idea behind this is to tackle with different encodings and malformed strings translated in different languages, or simply just

competing conventions and noisy data. As expected once that two instances share a very similar title and approximately same run time, it is enough to assume that there is match and the ground truth helped us to calibrate the threshold of this confidence. The main discovery of this paper is that by allowing a very small amount of leniency, one can improve recall drastically while keeping precision roughly the same. This however, only applies in the context of this experiment, and should be validated on other datasets as well.

Future work further on this seems promising. There are plenty of movie datasets through the Internet and many in different formats and datasets. Until now there are not big robust applications that entails many datasets for movies sufficiently because exactly the data exist in many different forms. Machine Learning approach is possibly the next step to solve the open problem of the datasets matching. Neural networks can learn to empirically to calibrate correct thresholds for the matcher when there is no explicit properties to instant link the datasets. Moreover, instead of engineering features, a deep net could learn the features it needs to extract similarity information.

References

1. Michal Holub, Ondrej Proksa, and Mária Bieliková. Detecting identical entities in the semantic web data. In *Proceedings of the 41st International Conference on SOFSEM 2015: Theory and Practice of Computer Science - Volume 8939*, pages 519–530, New York, NY, USA, 2015. Springer-Verlag New York, Inc.
2. Antoine Isaac, Lourens van der Meij, Stefan Schlobach, and Shenghui Wang. An empirical study of instance-based ontology matching. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web*, pages 253–266, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.