The Properties of Property Alignment

Michelle Cheatham and Pascal Hitzler

Data Semantics (DaSe) Laboratory, Wright State University, Dayton OH 45435, USA

Abstract. The performance of alignment systems on property matching lags significantly behind that on class and instance matching. This work seeks to understand the reasons for this and consider possible avenues for improvement. The paper contains an in-depth exploration of the performance of current alignment systems on the only commonly accepted alignment benchmark that involves matches between properties. A second benchmark involving properties is also proposed. Finally, an entirely string-based approach targeted towards aligning properties is presented and evaluated using both benchmarks.

1 Introduction

Previously, we conducted an analysis of the performance of string similarity metrics and preprocessing strategies on ontology alignment tasks [1]. One of the findings of that work was that string metrics perform much worse on properties than on classes. Furthermore, preprocessing strategies such as stopword removal and word stemming were ineffective at improving performance.

Others have noted the challenge of aligning properties as well. For example, this is stated without additional detail by Maedche and Staab in [4], while Pernelle et al. note that human experts had a more difficult time agreeing on when properties match than on when classes do [8]. In this paper we build on previous work by considering the difference in performance of full-featured alignment systems on properties versus classes (Section 2). In additional to overall performance, we look at the false positives and false negatives commonly made by current systems when aligning properties within the OAEI Conference track. Then, because the Conference track is the *only* commonly used non-synthetic alignment benchmark that involves properties, we introduce a potential new benchmark to allow for verification of results in Section 3. In Section 4 we continue our exploration of the limits of string-centric approaches for ontology alignment by introducing an entirely string-based property alignment system and evaluating its results on both the Conference track and our newly-proposed secondary benchmark. The results compare favorably to the best-performing string similarity metric and PARIS, a full-featured alignment system.

2 OAEI Conference Track

The OAEI Conference track is the only established non-synthetic test set for alignment systems that has reference alignments containing matches between

System	Class Prec	Class Rec	Class Fms	Prop Prec	Prop Rec	Prop Fms
AML	0.86	0.62	0.72	1.00	0.20	0.33
AMLback	0.86	0.64	0.73	1.00	0.24	0.39
CIDER_CL	0.46	0.59	0.52	0.07	0.22	0.11
HerTUDA	0.84	0.56	0.67	0.26	0.20	0.23
HotMatch	0.81	0.57	0.67	0.24	0.20	0.22
IAMA	0.87	0.55	0.67	0.14	0.07	0.09
LogMap	0.82	0.65	0.73	0.62	0.28	0.39
MapSSS	0.74	0.59	0.66	0.00	0.00	0.00
ODGOMS	0.87	0.55	0.67	0.32	0.26	0.29
ODGOMS1_2	0.81	0.66	0.73	0.32	0.26	0.29
ServOMap_v104	0.74	0.65	0.69	0.00	0.00	0.00
StringsAuto	0.71	0.63	0.67	0.00	0.00	0.00
WeSeEMatch	0.85	0.54	0.66	0.50	0.02	0.04
WikiMatch	0.84	0.54	0.66	0.26	0.22	0.24
YAM++	0.82	0.71	0.76	0.68	0.57	0.62
Average	0.79	0.60	0.68	0.36	0.18	0.21

Table 1. Performance of the top 2013 OAEI competitors on classes versus properties

properties as well as classes. Table 1 shows the results of the top 2013 OAEI competitors on the Conference track, broken down into classes and properties[1]. The average f-measure for classes is more than three times that for properties.

Table 2 presents the most common correct and incorrect property matches identified by the participants in the 2013 OAEI, along with the valid property matches that were most frequently omitted by those systems. The frequency column in the table indicates the number of alignment systems out of the 15 qualifying[2] systems that produced (or failed to produce, in the case of false negatives) each match. The first section of the table shows that the equivalent properties that were most frequently correctly identified all have very high string similarity. Unfortunately, the second section shows that high string similarity is also the defining characteristic of the most common false positives. It may seem surprising that some of the matches in this section of the table are not valid. In some cases the domain or range of the matched properties indicate that they are not being used in the same way. For instance, the domain of cmt:name is the union of Person and Conference whereas the domain of sigkdd:Name is only Person and a separate property, Name_of_conference, is used to represent a conference's name. In other cases the match may make sense in isolation but would lead to logical inconsistency of the merged ontology. Finally, we see in the last section of the table that the properties involved in the most common false negatives generally have a much lower string similarity, such as cmt:hasBeenAssigned and ekaw:ReviewerOfPaper. In many of these cases, the domain and range of the properties *do* have strong syntactic similarity however, e.g. Reviewer and Paper for hasBeenAssigned and Possible_Reviewer and Paper for reviewerOfPaper. Further, there were some quite frequently missed equivalent properties that have

[1] MapSSS and StringsAuto do not attempt to align properties

[2] Those performing better than the basic edit-distance string metric

strong clues in the labels themselves, such as cmt:writePaper and confOf:writes. Of the 31 common false negatives, 13 have noticeable string similarity.

3 YAGO-DBPedia

In addition to the OAEI Conference test set, we would also like to analyze the performance of alignment systems on properties from another real-world alignment task. For this we have chosen DBPedia[3] and YAGO.[4] DBPedia is a linked data version of the information in Wikipedia. The YAGO knowledge base has been automatically extracted from Wikipedia, WordNet, and GeoNames by researchers at the Max Planck Institute for Computer Science. Both DBPedia and YAGO contain millions of instances and thousands of schema-level entities. This scale is too large for many current alignment systems. We are specifically interested in aligning the properties of these two datasets, so we have extracted a cohesive subset of each one that will allow us to do this without requiring an inordinately long runtime. This was done using the following procedure:

1. For each property in YAGO, randomly choose five facts that involve the property. For properties with less than five facts, use all that are available.
2. Add the classes (type) of every instance mentioned in the facts from step 1.
3. Randomly add up to five other facts related to the instances from step 1.
4. Repeat step 2 for any additional instances added during step 3.
5. Compute the "closure" of this set of entities by recursively retrieving all schema-related axioms related to any entity within our sample.

The procedure for creating the DBPedia sample was analogous, except that instead of randomly choosing the facts in step 1, we selected facts with the same instances as our YAGO sample when available. This is possible because, since DBPedia and YAGO both represent information from Wikipedia, there is error-free mapping of instances that point to the same Wikipedia page. When there were no matching YAGO instances for the facts related to a particular DBPedia property, we reverted to randomly choosing facts. The characteristics of these dataset samples are shown in Table 3.

This dataset sample may be of use to other researchers, so we have made it publicly available at http://www.michellecheatham.com/files/dbpedia-yago.zip. It should be noted that DBPedia and YAGO have some idiosyncrasies. For instance, many properties defined in the ontologies are never used or are incompletely defined (e.g. missing domain or range definitions). Also, the definitions of some properties are spread across a datatype property, which specifies the range, and an annotation property, which specifies the domain. Furthermore, some of the properties appear to be used inconsistently, or at least more broadly than they are defined. For instance, in DBPedia we see that the instance HAL 9000

[3] http://wiki.dbpedia.org/Downloads39

[4] http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/

Type	Property 1	Property 2	Freq.
Correct	cmt:email	confOf:hasEmail	11
	confOf:hasFirstName	edas:hasFirstName	11
	conference:has_an_email	confOf:hasEmail	9
	cmt:email	conference:has_an_email	9
	conference:has_the_last_name	edas:hasLastName	9
	conference:has_a_review	ekaw:hasReview	9
	conference:has_the_first_name	edas:hasFirstName	9
	conference:has_the_first_name	confOf:hasFirstName	9
False Positive	iasted:pay	sigkdd:pay	9
	confOf:hasEmail	edas:hasEmail	9
	cmt:email	edas:hasEmail	8
	cmt:name	sigkdd:Name	8
	confOf:hasPhone	edas:hasPhone	8
	confOf:hasStreet	edas:hasStreet	7
	confOf:hasPostalCode	edas:hasPostalCode	7
	iasted:obtain	sigkdd:obtain	7
	confOf:hasTopic	edas:hasTopic	7
	conference:has_an_email	edas:hasEmail	7
	cmt:writtenBy	confOf:writtenBy	7
False Negative	cmt:hasBeenAssigned	ekaw:reviewerOfPaper	15
	cmt:assignExternalReviewer	conference:invites_co-reviewers	15
	cmt:assignedByReviewer	conference:invited_by	15
	edas:endDate	sigkdd:End_of_conference	15
	conference:is_given_by	sigkdd:presentationed_by	15
	conference:has_a...tutorial_topic	confOf:hasTopic	15
	conference:contributes	iasted:write	15
	cmt:hasBeenAssigned	confOf:reviewes	15
	conference:gives_presentations	sigkdd:presentation	15
	conference:has_the_last_name	confOf:hasSurname	15
	cmt:assignedTo	ekaw:hasReviewer	15
	confOf:reviewes	edas:isReviewing	15
	confOf:hasSurname	edas:hasLastName	15
	conference:has_a_review_expertise	edas:hasRating	15
	cmt:writtenBy	ekaw:reviewWrittenBy	15
	cmt:hasSubjectArea	confOf:dealsWith	14
	cmt:writePaper	confOf:writes	14
	edas:isReviewedBy	ekaw:hasReviewer	14
	cmt:hasAuthor	confOf:writtenBy	14
	confOf:writes	edas:hasRelatedPaper	14
	edas:hasCostAmount	sigkdd:Price	14
	cmt:assignedTo	edas:isReviewedBy	14
	edas:startDate	sigkdd:Start_of_conference	14
	cmt:hasConferenceMember	edas:hasMember	14
	cmt:hasBeenAssigned	edas:isReviewing	14
	edas:hasLocation	ekaw:heldIn	14
	edas:hasName	sigkdd:Name_of_conference	14
	edas:isReviewing	ekaw:reviewerOfPaper	14
	confOf:hasEmail	sigkdd:E-mail	13
	conference:has_an_email	sigkdd:E-mail	13
	conference:contributes	ekaw:authorOf	13

Table 2. Most common correct, false positive, and false negative property matches identified by alignment systems in the 2013 OAEI

Dataset	DBPedia	YAGO
Classes	617	10962
Object Properties	1046	85
Data Properties	1407	37
Named Individuals	8685	1680
Datatypes	23	23
Annotations	77	125
Total Entities	11855	12912

Table 3. Characteristics of the DBPedia and YAGO samples

has a gender property with a value of male and that Eaglet (Alice's Adventures in Wonderland) has a gender value female. In some cases the gender property is used differently, however: the instance Alexander has a gender property value of Alexandra, and the value for Maine North High School is mixed-sex education. While these issues can be a pain to work with, they are realistic concerns that ontology alignment systems will need to face for many application scenarios.

There is currently no curated alignment of the properties in the DBPedia and YAGO datasets. We would like to use the crowdsourcing approach described in Section 4 based on Amazon's Mechanical Turk system to create a complete reference alignment for the properties in these two datasets. It is not realistic to crowdsource opinion on all possible pairs of properties, however. A set of potential mappings is needed to bootstrap the crowdsourcing effort. Unfortunately, not many alignment systems have made results available for this pair of ontologies. The developers of the PARIS alignment system are the exception – they have produced and made public a set of subsumption relationships between properties [10]. We can consider the cases where subsumption relations between two properties exist in both directions as indicative of an equivalence relation. We will use these matches, together with those produced by a basic string similarity metric and by our string-based property matcher described in the next section to begin the process of crowdsourcing a viable reference alignment. Due to the limited number of alignment approaches providing the potential matches to verify, this method will allow us to assess precision reasonably well but recall values are likely to be less accurate. While less than ideal, this is a common method of evaluation in the absence of an established reference alignment [3,10,9]. Mechanical Turk has previously been successfully used by other researchers for a similar purpose – verifying relationships within biomedical ontologies [6].

4 String-based Property Alignment

In this section we present an entirely string-based approach to property alignment, which we will call PropString.[5]

Four strings are extracted for each property: the label, the core concept, the domain, and the range. The label is simply the entity's label. The core

[5] http://michellecheatham.com/files/PropString.zip

concept is either the first verb in the label that is greater than four characters long or, if there is no such verb, the first noun in the label, together with any adjectives that modify that noun. For example, the label "wrote paper" has the core concept "wrote" and the label "has corresponding author" has the core concept "corresponding author." We arrived at this technique through an analysis of common naming patterns for properties. We used the Standford log-linear part of speech tagger to compute the core concept [15]. The domain (resp. range) string is a concatenation of the labels of any classes in the domain (resp. range) of a property. The similarity of each of these four pairs of strings is then computed using the Soft TF-IDF metric, which was the string metric shown in [1] to have the best performance on properties.

While the vast majority of alignment systems use a string similarity metric, they use them in different ways. One approach is to find highly precise "anchor" matches which serve as the seed that the rest of the alignment grows out from. Another approach is to use a string metric to filter out any obviously incorrect matches in order to reduce computational complexity. This requires a string metric with high recall. To address both of these use cases, the PropString approach can be run in two configurations: precision-oriented and recall-oriented. In the precision-oriented mode, a pair of entities is considered a match if the similarity values for their core concepts, domains, **and** ranges are all greater than the threshold. In the recall-oriented mode, the pair is considered a match if the similarity values for their core concepts **or** their domains and ranges are greater than the threshold.

Allowing matches based solely on high similarity of domain and range in the recall-oriented configuration results in very low precision unless further steps are taken. We use a combination of two approaches to reduce the number of false positives. The first is the calculation of the confidence value: this is done by averaging the similarity values for the exact labels, their domains, and their ranges. The second is that we keep a list of each entity that is considered a match so far, along with the entity it maps to and the confidence value. Every time a new potential match between properties is identified, its confidence value is checked against any existing current matches involving those properties. If the new match has a greater confidence value, the old match is removed in favor of the new one, otherwise the new match is ignored. Using the exact label similarity when computing the confidence values rather than the core concept eliminates the loss of precision associated with extracting the core concept, effectively breaking any ties in favor of the closer lexical matches. The effect of this approach is that any properties with the same domain and range act as a filter, with the specific match from that set chosen based on the actual property label.

4.1 Evaluation: Conference track

Table 4 shows the results of PropString on the OAEI Conference track. The system was configured with a threshold of 0.9 and to only include matches in which both entities were in the namespace of the ontologies to be matched (in accordance with the OAEI guidelines). The results are compared with those of Soft

System	Precision	Recall	F-measure
PropString (prec)	1.0	0.26	0.41
PropString (rec)	0.34	0.5	0.4
Soft TF-IDF	0.2	0.24	0.22

Table 4. Results on the OAEI Conference track

TF-IDF with a threshold of 0.8. This was shown in [1] to be the best-performing string metric for property alignment. It is evident that PropString greatly outperforms Soft TF-IDF on this test set. The precision-oriented configuration of PropString quintuples the precision of Soft TF-IDF (to a perfect 1.0) while maintaining roughly the same recall. Analogously, the recall-oriented version doubles the recall of Soft TF-IDF while still achieving noticeably better precision. The f-measures for both the precision- and recall-oriented configurations are double that of Soft TF-IDF.

We also conducted a series of tests which show that there are no redundant aspects to the PropString metric: removing any element reduces performance. In particular, removing the idea of extracting the core concept from property labels has such a disastrous effect on recall that the precision-oriented configuration becomes useless. Similarly, removing either the best match filter or using simple label similarity for the confidence value rather than averaging label, domain, and range similarity cuts precision in half in the recall-oriented configuration. Consideration of domain and range in the similarity computation is shown to be the key to this approach.

4.2 Evaluation: YAGO-DBPedia

We also evaluated the performance of PropString on the YAGO-DBPedia alignment task. We compare the performance of PropString to that of the basic Soft TF-IDF similarity metric and the PARIS alignment system. PARIS is an acronym for Probabilistic Alignment of Relations, Instances, and Schema. The system approaches property alignment by considering the degree of overlap between the sets of instances involving each property [11].

There is no established reference alignment for the DBPedia and YAGO ontologies. We begin the process of creating one by collecting the equivalent property relationships generated by PropString, Soft TF-IDF, and PARIS and using Amazon's Mechanical Turk to verify their accuracy. In total, these three approaches produced 133 unique equivalence matches that involved properties. We formulated questions for each match of the form "Does property label A mean the same thing as property label B?" Respondents were instructed to choose one of four options: they mean the same thing, one is a more general or more specific term than the other, they are related in some other way, or there is no relation. We provided these more nuanced options rather than just yes or no because we would like to eventually develop a reference alignment useful for evaluating the performance of alignment systems that produce all types of matches. In order to

provide some context, we provided information about the domain and range of each property and up to five examples of instances with values for each property.

The 133 matches were grouped into 19 sets of 7 questions each, and we paid 25 cents for each set. Preliminary testing showed that the general response on these nuanced verification questions were not very reliable (others have indicated problems with scammers for these tasks as well [6]). We therefore invited only Turkers who had previously demonstrated good performance on alignment verification tasks to participate in this one. There were ten of these individuals, and we received input from 6 or 7 of them for each match.

Rather than requiring precise agreement on the type of relationship (if any) for each potential match, it might make sense for our current purposes to consider a weaker sense of agreement. One option is to consider two answers to be in agreement if they both either indicate some relationship exists or they both conclude there is no relation between the two properties. In this case, if one person indicated two entities are related in a sub/super relationship and another indicated that they are equivalent, these answers would be considered in agreement. Two answers would only be seen to disagree if one indicated there is no relation at all and the other disagreed. This way of interpreting the results might be useful for an alignment system if the results from this phase were being used to either find all types of relationships between entities or to gather all possible matches and use further processing to filter the set down to only equivalence relations. We will call this "recall-oriented." Figure 1 (top) shows the results of PARIS, Soft TF-IDF, and PropString on the YAGO-DBPedia property alignment task using this definition of correctness.

Another possible way to interpret the results is to consider two answers to be in agreement only if they both conclude either that the entities are precisely equivalent or that they are not equivalent. Using this viewpoint, if one person indicated that two entities are related in a sub/super relationship and another indicated that they are precisely equivalent, these answers would be seen as disagreeing. If instead one person considered the match to be a sub/super relationship and another considered them to have no relationship at all, these two individuals would be seen as in agreement because they both conclude that there is no equivalence relationship. This interpretation may be useful if an alignment system is attempting to find high-quality equivalence relations between entities, which it may subsequently use as a seed for further processing. We will refer to this as "precision-oriented." Figure 1 (bottom) is analogous to Figure 1 (top) but uses this precision-oriented definition of correctness.

The basic string metric Soft TF-IDF produces the highest precision, regardless of how is correctness is measured. Further, that precision is 0.79 and .96 (depending on evaluation approach), which is on par with the degree of agreement among the Turkers on these matches. So we see that a straightforward string metric can in some ways outperform more sophisticated alignment strategies. In fact, PARIS and the precision-based configuration of PropString have such low recall that they may not be of much utility for many application sce-

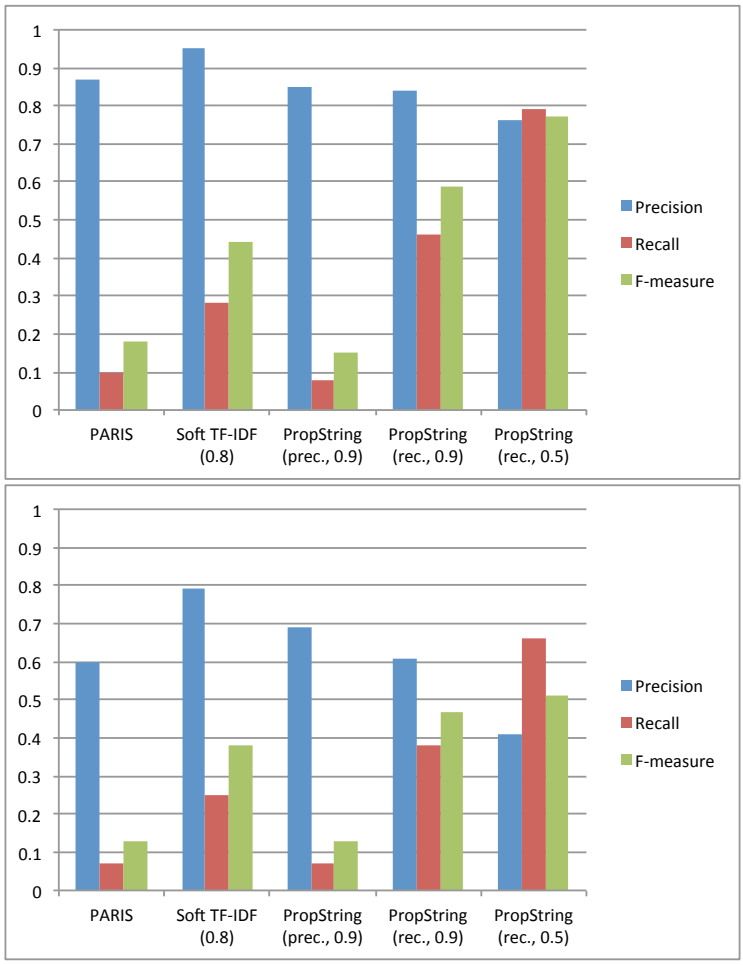

Fig. 1. Results of the YAGO-DBPedia alignment task: Recall-oriented evaluation (top), precision-oriented evaluation (bottom)

narios. This is surprising considering the strong performance of this PropString configuration on the properties within the Conference track.

Another thing to note from these results is the very strong performance of the recall-based configuration of PropString, both relative to the other approaches and in an absolute sense. When PropString is run in its recall configuration with a threshold of 0.5, both the precision and recall are in the neighborhood of that produced by much more complex alignment systems on the simpler task of class equivalence in smaller test sets, such as the Conference track. Of course, the very preliminary nature of the YAGO-DBPedia reference alignment must be kept in

mind. More work, hopefully involving results produced by many other alignment system on this pair of ontologies, is needed to confirm these results.

5 Related Work

We are not aware of any existing alignment approaches focused specifically on property matching. However, there has been some work in the alignment field that has implications for that goal.

Empirical analysis of existing ontologies has shown that different naming conventions are used for different entity types. For instance, empirical analysis of existing ontologies has shown that object properties generally begin with a verb (e.g. attends, employs) or end with a preposition (e.g. friendOf, componentFor) while datatype properties are usually nouns (e.g. value, id, etc.). Additionally, the names of inverse properties were found to commonly follow one of two patterns: (1) active and passive forms of the same verb (e.g. wrote and writtenBy) or same noun phrase packed in auxiliary terms (e.g. memberOf and hasMember) [14]. These different naming conventions may be one reason for the generally poor performance of string similarity metrics on properties.

In 2002 Melnik and his colleagues developed a strategy called "similarity flooding" to improve the performance of alignment systems. The general idea is that an initial pass is made through the datasets to establish a set of high precision anchor mappings, such as exact string matches. Then similarity values are propagated to adjacent nodes. If the similarity value of two nodes reaches a threshold, they are considered equivalent. The algorithm iterates until a fixed point is reached [5]. This technique may improve the performance on property alignment by leveraging the increased accuracy of class and instance alignment.

An ontology-centric version of the basic similarity flooding technique was first employed in RiMOM and subsequently adopted by many other alignment systems. Rather than propagating similarity values to all neighbors in a graph, this approach considered sub-concepts, siblings, and properties for classes and sub-properties, range, and domain for properties [2]. Suchanek et. al. recently applied this ontology-oriented similarity flooding approach in their PARIS alignment system, which identifies both equivalences and subsumptions for classes and properties [10]. They found that while class alignments didn't do much to facilitate alignment of properties or instances, there was significant interplay between the latter two. This was particularly true for functional or nearly functional properties, in which any domain value maps to only one range value.

There have been several attempts to modify the standard similarity flooding approach to further improve the performance on property matching. For example, comparison of instance data and datatype property range values can be improved by using different similarity metrics for strings, dates, integers, etc. [16]. Further, in deference to the difficulty of matching properties, it is possible to propagate a fraction of the normal similarity values when adjacent properties are compatible rather than definite matches. This is the approach taken in [8]

where compatibility for properties is defined as those with domains and ranges that are either the same or subtypes of one another.

Another particularly problematic aspect for property matching is the variety of design decisions made when an ontology is created. For instance, some ontologies are class-centric while others are property-centric (e.g. SeasonTicketHolder versus holdsSeasonTicket) [12]. Intuitively we would like to say that if two ontologies had these entities, there should be some sort of mapping between them. Other ontology design decisions that impact property matching are how to handle part-whole relationships and when to reify properties [7]. Additionally, taxonomies of properties are much less common than those of classes [14,7]. There has been some discussion in the literature of handling differences in design philosophy through ontology transformation, in which design patterns are recognized and translated into an analogous form [13]. Ritze and her colleagues used this pattern-centric approach to find complex mappings between classes (and value restrictions) in one ontology and properties in another [9].

6 Conclusions and Future Work

This work explored the performance of current ontology alignment systems on property alignment using the OAEI Conference track as a benchmark. In addition, a second benchmark involving property matches was suggested. The paper also introduced PropString, an entirely string-based approach to aligning properties. The performance of PropString was evaluated using both benchmarks and was shown to be better than the best-performing string metric by a wide margin. PropString also compared favorably to the PARIS alignment system on the secondary benchmark, based on a crowdsourced evaluation of matches using Mechanical Turk.

Several aspects of the work presented here require further validation. In particular, additional experimentation regarding crowdsourcing reference alignments using Mechanical Turk needs to be done to verify the potential uses of the approach. For instance, our preliminary results showed that general users can often give good input on "yes or no" alignment verification tasks but that more complex questions regarding the type of relationship between two entities (e.g. equivalence, subsumption, inverse properties) is more difficult. It would be useful to develop guidelines for when and how to qualify users for different types of alignment tasks. More work in particular remains to be done in order to generate an established high-quality reference alignment for the YAGO-DBPedia alignment task. In order to do this, we need to generate results on this ontology pair using more alignment systems. These results can then be manually verified, either through Mechanical Turk or by experts. Additionally, we would like to incorporate the PropString approach into a full-featured alignment system and evaluate the difference in performance.

Acknowledgments. This work was supported by the National Science Foundation under award 1017225 "III: Small: TROn—Tractable Reasoning with Ontologies."

References

1. Cheatham, M., Hitzler, P.: String similarity metrics for ontology alignment. In: The Semantic Web–ISWC 2013, pp. 294–309. Springer (2013)
2. Li, J., Tang, J., Li, Y., Luo, Q.: Rimom: A dynamic multistrategy ontology alignment framework. Knowledge and Data Engineering, IEEE Transactions on 21(8), 1218–1232 (2009)
3. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic schema matching with cupid. In: VLDB. vol. 1, pp. 49–58 (2001)
4. Maedche, A., Staab, S.: Measuring similarity between ontologies. In: Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web, pp. 251–263. Springer (2002)
5. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In: Proc. 18th ICDE Conf.(Best Student Paper award) (2002)
6. Mortensen, J.M., Musen, M.A., Noy, N.F.: Crowdsourcing the verification of relationships in biomedical ontologies. In: AMIA Annual Symposium (submitted, 2013) (2013)
7. Noy, N.F., Hafner, C.D.: The state of the art in ontology design: A survey and comparative review. AI magazine 18(3), 53 (1997)
8. Pernelle, N., Saïs, F., Safar, B., Koutraki, M., Ghosh, T.: N2r-part: identity link discovery using partially aligned ontologies. In: Proceedings of the 2nd International Workshop on Open Data. p. 6. ACM (2013)
9. Ritze, D., Meilicke, C., Sváb-Zamazal, O., Stuckenschmidt, H.: A pattern-based ontology matching approach for detecting complex correspondences. In: ISWC Workshop on Ontology Matching, Chantilly (VA US). pp. 25–36. Citeseer (2009)
10. Suchanek, F., Abiteboul, S., Senellart, P.: Ontology alignment at the instance and schema level. arXiv preprint arXiv:1105.5516 (2011)
11. Suchanek, F.M., Abiteboul, S., Senellart, P.: Paris: Probabilistic alignment of relations, instances, and schema. Proceedings of the VLDB Endowment 5(3), 157–168 (2011)
12. Šváb, O.: Exploiting patterns in ontology mapping. In: The Semantic Web, pp. 956–960. Springer (2007)
13. Sváb-Zamazal, O., Svátek, V., Scharffe, F., et al.: Pattern-based ontology transformation service. In: Proc. 1st IK3C international conference on knowledge engineering and ontology development (KEOD). pp. 210–223 (2009)
14. Svátek, V., Šváb-Zamazal, O., Presutti, V.: Ontology naming pattern sauce for (human and computer) gourmets. In: Workshop on Ontology Patterns. pp. 171–178 (2009)
15. Toutanova, K., Klein, D., Manning, C.D., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1. pp. 173–180. Association for Computational Linguistics (2003)
16. Zhao, L., Ichise, R.: Instance-based ontological knowledge acquisition. In: The Semantic Web: Semantics and Big Data, pp. 155–169. Springer (2013)