Domain-Independent Entity Coreference in RDF Graphs

Dezhao Song
Lehigh University
19 Memorial Drive West
Bethlehem, PA 18015
des308@lehigh.edu

Jeff Heflin
Lehigh University
19 Memorial Drive West
Bethlehem, PA 18015
heflin@cse.lehigh.edu

ABSTRACT

In this paper, we present a novel entity coreference algorithm for Semantic Web instances. The key issues include how to locate context information and how to utilize the context appropriately. To collect context information, we select a neighborhood (consisting of triples) of each instance from the RDF graph. To determine the similarity between two instances, our algorithm computes the similarity between comparable property values in the neighborhood graphs. The similarity of distinct URIs and blank nodes is computed by comparing their outgoing links. To provide the best possible domain-independent matches, we examine an appropriate way to compute the discriminability of triples. To reduce the impact of distant nodes, we explore a distance-based discounting approach. We evaluated our algorithm using different instance categories in two datasets. Our experiments show that the best results are achieved by including both our triple discrimination and discounting approaches.

Categories and Subject Descriptors

I.2.6 [**Learning**]: Knowledge acquisition

General Terms

Algorithms, Experimentation, Theory

Keywords

Discriminability, Entity Coreference, Semantic Web

1. INTRODUCTION

The purpose of entity coreference is to decide if different mentions (person names, place names, ontology instances, etc) in documents (free text, web pages, etc) refer to the same real world entity. The entity coreference task is challenging primarily due to two general aspects: how to locate context information for each mention and how to utilize the context in an appropriate way. To collect context information, we can utilize the documents where the mentions occur. On the other hand, it is really significant to use the context appropriately. Name variations, the use of abbreviations, and misspellings can all affect the final results. Also, context information from heterogeneous sources may not be complete. For instance, two documents may describe different aspects of a person. The name and affiliation can appear in one document while the other one can have name, date of birth, email address, etc. In addition, there may be noise in the data provided.

In this paper, we present a novel entity coreference algorithm for ontology instances. Given a pair of instances of comparable classes, our algorithm tells if they are coreferent, i.e., they refer to the same real world entity, such as the same person or publication. As a preprocessing step, we learn the discriminability of each triple, taking into account its predicate. Then we use an expansion process to extract a neighborhood graph to serve as the context for each instance from the dataset (an RDF graph). We compare the triples in the graphs of different instances, taking into account the discriminability of their predicates, as well as applying a discount according to each triple's distance to the root node (the ontology instance). We show that using a combination of discriminability and discounting results in an F1-score ranging from 86.7% to 96.7% for two types of ontology instances and two distinct datasets. This performance is better than using neither feature, as well as better than using either feature alone.

2. RELATED WORK

Entity coreference, also known as entity resolution or entity matching, has been the subject of much research. Bagga and Baldwin [2] employ a vector space model to do cross-document entity coreference on mentions in free text. Gooi and Allan [3] employ three different statistical models for entity coreference on person mentions. They use a window size of 55 words centered on a mention to collect context. Mann and Yarowsky [6] utilize an unsupervised clustering technique over a feature space for coreference. They extract more representative information from web pages, such as biographical information.

In the Semantic Web, Hassel et al. [4] propose an ontology driven disambiguation algorithm to match instances from an ontology created from the DBLP bibliography [5] to mentions in DBWorld documents[1], using several features, such as the co-occurrence relationship. Differently, Aswani et al. [1] propose an algorithm for matching two ontology

[1]http://www.cs.wisc.edu/dbworld/

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.

instances. One of their focuses is to exploit the web to find information to support the coreference process.

3. ENTITY COREFERENCE ALGORITHM

In this section, we formally present our entity coreference algorithm, including how to locate and appropriately utilize context information and the core algorithm.

3.1 Selecting a Neighborhood Graph

In our current approach, we use the RDF graph as the sole source of context information. Starting from the root node, i.e., an ontology instance, we *expand* it by searching for triples whose subject (object) equals to its URI. For such triples, if their objects (subjects) are still URIs or blank nodes, then we repeat this expansion process to get more triples until we reach a predefined depth limit or a literal value, whichever comes first. Upon completing our search, we have a neighborhood/context consisting of a set of paths for each ontology instance, starting from that instance and ending with a URI or a literal value. A path is defined as follows: $path = (r, p_1, n_1, ..., p_n, n_n)$, where r is the root node, $n_i (i > 0)$ is any expanded RDF node and p_i is a predicate in the path.

We use the function $N(G, i)$ to represent the paths for instance i given a RDF graph G. We denote the RDF node at the rear of a path as $E(path)$, where $path \in N(G, i)$. We do not record paths that end in blank nodes because such paths provide little information about an instance. However, we rely on paths that go through blank nodes to get further literals and URIs. Based on experimentation, we found that a depth limit of 2 provides sufficient context information to perform coreference in the RDF graph.

3.2 Calculating Path Weights

Generally, each triple has its own importance, reflecting its possible level of discrimination to the ontology instance from which it originally comes from. Our approach, given a dataset, takes the entire dataset (triples) as input and automatically learns the discriminabilities regardless of the domain. Thinking broadly, we can measure the discriminability of a triple by looking at what its predicate is. As for a predicate, the more diverse value set it has, the more discriminative it will be. Equation 1 computes the predicate discriminability.

$$Per_{p_i} = \frac{number\ of\ distinct\ objects\ of\ p_i}{number\ of\ occurrences\ of\ p_i} \quad (1)$$

where Per_{p_i} represents a percentage value for predicate p_i. We record the max percentage value as Per_{max} and normalize all percentage values so that the most discriminative predicate has a discriminability of 1. Recall from section 3.1, that during the expansion process we try a URI or a blank node both as a subject and object (i.e., a predicate may be used in the subject-to-object direction or the object-to-subject one). Thus a predicate could have two discriminabilities in different directions. For a given predicate p_i, we denote a predicate's discriminabilities in the two directions as P_{p_i} and $P^-_{p_i}$, respectively.

When counting the size of the distinct object or subject value set, we assume that if any two objects/subjects have distinct syntactic forms, then they truly represent different things. However, because RDF/OWL does not make a unique name assumption, they could actually represent the same real world entity, resulting in an overestimation of the discriminability. But if we assume that for every predicate such unknown coreferent relationships occur uniformly throughout the dataset, we actually overestimate all predicates by the same proportion. Thus our current approach still gives reasonable discriminability.

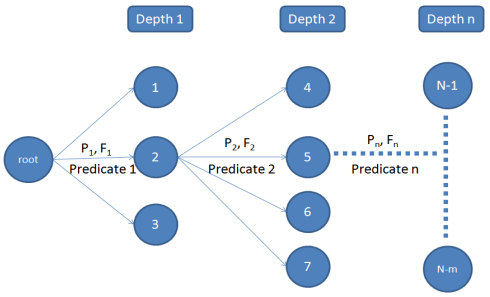


Figure 1: Assign Weight to Neighborhood Graph

With the learned discriminability, we assign each path in the neighborhood graph a weight, indicating its importance to the root node. The weight combines two elements, the learned discriminability and a discount value. Figure 1 displays how we assign the weight to each path. In Figure 1, when we expand the root, we get nodes 1, 2 and 3 and we reach nodes 4, 5, 6, and 7 by further expanding node 2, so on and so forth. P_1 and P_2 represent the discriminabilities for the two triples ending with node 2 and 5 respectively. We also add another parameter to each node called its factor, indicating how important a node is to its parent node. The factor evenly distributes a parent node's weight among all of its children, thus the value of factor F_1 in the figure is one-third. Using factors and the learned discriminability, we take a distance-based discounting approach to assign a weight to a path as shown in Equation 2:

$$W_{path} = \prod_{i=1}^{length(path)} P_i * F_i \quad (2)$$

where the length function counts the number of predicates in a path; P_i and F_i represent the ith triple's discriminability in the path and the factor of its subject/object node respectively. This distance-based discounting approach minimizes the effect of potentially noisy nodes that result from deep expansion of the neighborhood graph.

3.3 Algorithm Design

Once we have identified the neighborhood graph for each instance and computed the weight for each path, we are ready to perform entity coreference. Algorithm 1 presents the pseudo code for our entity coreference algorithm for ontology instances. In this description, the function *Comp* tells if two paths are comparable; *JW* computes a Jaro-Winkler TFIDF score between two literals.

The essential idea is that we adopt the bag-of-paths approach to compare paths between ontology instances, such as instanceA and B. For each path (pathA) of instanceA, we compare its rear node to the rear node of every path of instanceB with a comparable sequence of predicates and choose the highest similarity score, denoted as path_scoreA. Also, we need to determine the weight of this path_score, denoted as path_weight, using the average of the weight

Algorithm 1 Compare(N_a, N_b), N_a is the context N(G,a) and N_b is N(G,b)

1. $total\_score \leftarrow 0, total\_weight \leftarrow 0$
2. **for all** *paths* $m \in N_a$ **do**
3. **if** $\exists path\ n \in N_b$, $Comp(m,n)$ **then**
4. $path\_score \leftarrow 0, path\_weight \leftarrow 0$
5. **if** $E(m)$ *is literal* **then**
6. $path\_score$
7. $\leftarrow \max_{n' \in N_b, Comp(m,n')} JW(E(m), E(n'))$
8. /*path n' has the highest JW score with m*/
9. $path\_weight \leftarrow (W_m + W_{n'})/2$
10. **else if** $E(m)$ *is URI* **then**
11. **if** $\exists path\ n' \in N_b$, Comp(m, n'), E(m)=E(n') **then**
12. $path\_score \leftarrow 1$
13. /*path n' has identical rear node with m*/
14. $path\_weight \leftarrow (W_m + W_{n'})/2$
15. **end if**
16. **end if**
17. $total\_score \leftarrow total\_score + path\_score * path\_weight$
18. $total\_weight \leftarrow total\_weight + path\_weight$
19. **end if**
20. **end for**
21. **return** $total\_score/total\_weight$

(weightA) of pathA and that (weightB) of the path of instanceB with which pathA has the highest similarity score. We need to repeat the process for every path of instanceA. With the pairs of (path_score, path_weight) for a pair of instances, we calculate their weighted average as the final similarity measure between them. The same process is repeated for all pairs of ontology instances of comparable categories, i.e., person-to-person and publication-to-publication. Algorithm 1 only shows the process to compare two instances; this comparison is repeated for each pair of instances in a given instance set.

As we described, we will compare instanceA's paths with paths of instanceB, which raises the question of how we know that the predicate sequences of two paths are comparable. In some situations, the comparability of predicates is not very clear, such as predicates "author-on" and "edit-on". For a publication, a person that did some edits on it does not necessarily have to be listed as an author of it. We say two predicates are comparable if the knowledge base (KB) entails that one is the subproperty of another (obviously, this means equivalent properties are also comparable). For our experiments, we created these mapping axioms manually. For example, the following two predicates are comparable "full-name" and "pretty-name". Furthermore, in our current implementation, we only check the comparability of the last predicate (treated as a composition predicate) from two paths to determine path comparability while ignoring the others. Note, we also use entailment to determine if two classes are comparable.

Another challenge is that we cannot make a closed-world assumption, some information can be missing from the original RDF graph. We address this problem by not applying penalties to URI mismatches. This means that if the URI rear node of pathA of instanceA doesn't match any rear node of comparable paths of instanceB, we do not add any weight to the total_weight. Second, we do not apply any penalties on missing information. If there isn't any path of instanceB that is comparable to pathA of instanceA, still we do not apply any penalties. We compare every path present in the context and apply appropriate penalties; in the meanwhile, any mismatches that might be caused by information incompleteness cannot simply be treated as real mismatches.

4. EVALUATION

We evaluate our algorithm on person and publication instances from the RKB dataset[2] and on person instances from the SWAT dataset[3]. To form our RKB dataset, we picked eight subsets from the entire RKB dataset: ACM, DBLP, CiteSeer, EPrints, IEEE, LAAS-CNRS, Newcastle and ECS. As for our SWAT dataset, the data is parsed from XML dumps of CiteSeer and DBLP. The transformation from the original XML files into RDF was done by the SWAT research lab at Lehigh University. Although the two datasets share some information, the main differences are: (1) they use different ontologies with different predicates. (2) their coverage could be different. (3) some information may be ignored from the original XML files for the SWAT dataset during transformation. The owl:sameAs statements are removed from each dataset before evaluation, and used when evaluating the results of the algorithms.

To increase the ambiguity of our test sets, we randomly picked 1,601 person and 2102 publication instances from the RKB dataset and 1010 person instances from the SWAT dataset through a filtering process. If the names/titles of two instances have a similarity score higher than 0.5 but are still said not to be coreferent based upon the groundtruth, we will put this pair of instances into an instance pool. Then we apply our algorithm on every pair of instances in each of the three test sets. In our evaluations, we use the standard measures, precision, recall and F1-score.

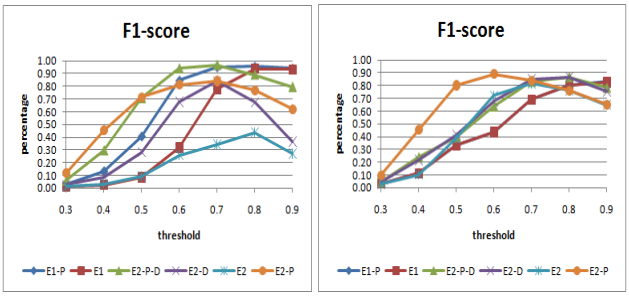
There are a few things to note. First, RKB provides coreference groundtruth between instances in the form of owl:sameAs statements. We verified such groundtruth by manually checking 300 coreferent pairs of person and publication instances respectively. Furthermore, to ensure the completeness of RKB groundtruth, we manually checked the "wrongly" detected pairs from a comparison system (to be described later) that obtains the lowest precision and were able to add 295 coreferent pairs to the RKB person groundtruth. For the SWAT dataset, the authors hand-labeled the groundtruth. Additionally, we materialize all coreferent pairs that can be achieved through transitivity reasoning since the owl:sameAs predicate is transitive.

We compare our proposed algorithm to some comparison systems with a subset of the features we have presented. The features include: expansion (E#) (# represents the depth of expansion), triple discriminability (P), discount (D) which is implemented with the factor. The comparison systems are as follows: E1, E1-P, E2, E2-P, E2-D, E2-P-D. For example, our proposed algorithm (E2-P-D) uses depth 2 expansion and includes discount and predicate discriminability to form path weight. We adapt Equation 2 to compute different path weights for other systems.

Figures 2(a), 2(b) and 3 show the F1-scores of RKB publication, RKB person and SWAT person instances respectively. Although our entity coreference algorithm (E2-P-D) is often bested at lower thresholds, it can achieve the best or nearly the best performance at higher thresholds. On the

[2]http://www.rkbexplorer.com/data/

[3]http://swat.cse.lehigh.edu/resources/data/index.html



(a) RKB Publication (b) RKB Person

Figure 2: RKB Results

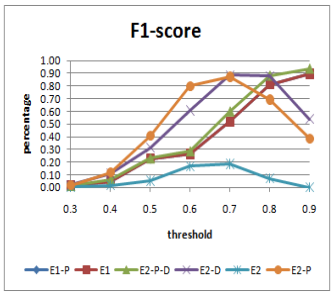


Figure 3: SWAT Results

RKB person instances, its best score (at threshold 0.8) is 2% lower than E2-P's best F1-score (at threshold 0.6), but its best F1-score is better in the other two test sets. The F1-score that our algorithm achieves for RKB publication, RKB person and SWAT person is 96.7%, 86.7% and 93.5% at threshold 0.7, 0.8 and 0.9 respectively. Compared to E1 where no proposed feature is used, E2 finds a larger neighborhood graph. But without discounts and discriminability, it is clearly worse than E1 for RKB publication and SWAT person; however, interestingly for RKB person, E2 performs better than E1 from 0.3 to 0.7 but is surpassed by E1 at 0.8 and 0.9. Furthermore, E1-P is better than or as good as E1 and E2-P is better than E2 for RKB publication and SWAT person at all thresholds, showing the effectiveness of predicate discriminability. For RKB person, E2-P is better than E2 at low thresholds from 0.3 to 0.6 but they start to have relatively the same performance from 0.7 to 0.9. Note that sometimes the curve for E1-P or E1 is not clear because they are overlapping. Additionally, the difference between E2 and E2-D shows that by only applying factor discounts can also give us a significant improvement particularly for RKB publication instances and SWAT person instances. There is still some improvement for RKB person instances but not as significant as shown in the other two test sets. Last, our proposed algorithm E2-P-D, compared to E2-D, shows better results for RKB publications at all thresholds and for SWAT person at 0.9. This verifies the effectiveness of using predicate discriminability. When comparing to E2-P, E2-P-D shows significant improvement for RKB publication from 0.6 to 0.9 and SWAT person instances at 0.8 and 0.9. For full details of the experiment, please refer to our technical report [7].

The results show certain advantages of our approach; however, there are a few points to discuss. First, as we apply higher thresholds, recall generally goes down for all systems (due to space limitations, recall graphs are not shown). One possible solution is an iterative approach that, at each step, merges the context of instances that are very similar. Second, currently, we do not apply penalties for URI mismatches or missing information. We don't want to sacrifice recall while still having a good control on precision by exploiting appropriate weights and context information. In the future, we will explore appropriate ways to better address this issue. Finally, our bag-of-paths approach can hurt the results in some cases because it ignores some underlying semantics. For example, although comparing only the last predicate of two paths speeds up the process, the fact is that if intermediate predicates or nodes are not comparable, then the paths should not be comparable.

5. CONCLUSION AND FUTURE WORK

In this paper, we propose an entity coreference algorithm for Semantic Web instances. Our algorithm finds a neighborhood graph of an instance as its context information. With our discriminability learning scheme and the distance-based discounting approach, we assign a weight to each path in the context. We adopt a bag-of-paths approach to compute the similarity measure between instances pairwisely. Our algorithm is verified with three test sets and it achieves the best performance on two of them. For future work, we plan to try some iterative entity coreference algorithm in order to apply appropriate penalties on URI mismatches. Also, we are interested in exploring some graph-based matching algorithms because, essentially, the context for an ontology instance is not a set of paths but a graph.

6. REFERENCES

[1] N. Aswani, K. Bontcheva, and H. Cunningham. Mining information for instance unification. In *International Semantic Web Conference*, pages 329–342, 2006.

[2] A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In *COLING-ACL*, pages 79–85, 1998.

[3] C. H. Gooi and J. Allan. Cross-document coreference on a large scale corpus. In *HLT-NAACL*, pages 9–16, 2004.

[4] J. Hassell, B. Aleman-Meza, and I. B. Arpinar. Ontology-driven automatic entity disambiguation in unstructured text. In *International Semantic Web Conference*, pages 44–57, 2006.

[5] M. Ley. The DBLP computer science bibliography: Evolution, research issues, perspectives. In *SPIRE*, pages 1–10, 2002.

[6] G. S. Mann and D. Yarowsky. Unsupervised personal name disambiguation. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL*, pages 33–40, 2003.

[7] D. Song and J. Heflin. Domain-independent entity coreference in RDF graphs. Technical report, Lehigh University, LU-CSE-10-004, 2010.