Scaling Data Linkage Generation with Domain-Independent Candidate Selection

Dezhao Song and Jeff Heflin

Department of Computer Science and Engineering, Lehigh University,
19 Memorial Drive West, Bethlehem, PA 18015, USA
{des308,heflin}@cse.lehigh.edu

Abstract. We propose a candidate selection algorithm for scalably detecting coreferent instance pairs from heterogeneous Semantic Web data sources. Our algorithm selects candidate pairs by computing a character-level similarity on disambiguating literal values that are chosen using domain-independent unsupervised learning. We index the instances on such values to efficiently look up similar instances. Our algorithm is evaluated on three instance categories in two RDF datasets.

Keywords: Linked Data, Entity Coreference, Scalability, Candidate Selection, Domain-Independence

1 Introduction

We propose a candidate selection algorithm for scalably detecting *owl:sameAs* links between ontology instances with the following key features: 1) Through domain-independent unsupervised learning, we learn a set of datatype properties as candidate selection key that both discriminate and cover the instances well; 2) We index the instances on the object values of the learned predicates for efficient look-up of similar instances; 3) We adopt a character level n-gram based string similarity measure to select candidate pairs; 4) We propose to apply an actual entity coreference system to the selected pairs to measure the F1-score of the coreference results and the runtime of the entire process. Extending our research paper [2], we present the runtime for both candidate selection and coreference, analyze the impact of algorithm parameters, and discuss the limitations of our approach. Due to space limit, please refer to our research paper for full details.

2 Algorithm

We learn candidate selection key, a set of datatype predicates, whose object values are sufficiently discriminating such that the vast majority of instances in a dataset use at least one of such predicates. The algorithm starts with an RDF graph G (a set of triples) and extracts the datatype predicates (*key_set*) and the instances (I_C) of certain categories (C) from G. Then, for each predicate $key \in key\_set$, the algorithm retrieves all the object values of *key* for instances in I_C and computes $key's$ discriminability (Eq. 1), coverage (Eq. 2) and their F1-score: $F_L = \frac{2*dis*cov}{dis+cov}$, where i and o are the subject and object of a triple.

$$dis(key, I_C, G) = \frac{|\{o|t =< i, key, o >\in G \wedge i \in I_C\}|}{|\{t|t =< i, key, o >\in G \wedge i \in I_C\}|} \tag{1}$$

$$cov(key, I_C, G) = \frac{|\{i|t =< i, key, o >\in G \wedge i \in I_C\}|}{|I_C|} \tag{2}$$

We remove predicates whose discriminability is lower than β. If any predicate has a F_L higher than a given threshold α, the predicate with the highest F_L (max_{key}) is chosen; otherwise, the algorithm combines max_{key} with every other predicate to add $|key\_set|$-1 virtual predicates to key_set and remove old ones. For a new key, we concatenate the object values of its predicates to form new triples that are also added to G. The same procedure can then be applied iteratively.

Next, tuples, in the form of t=<i,p,v>, are formed for instances in a given dataset with each of the learned predicate. Instances are indexed on the object values of the learned predicates to enable efficient look-up. For each tuple t, we issue a Boolean query, the disjunction of its tokenized values, to the index to search for tuples with similar values on all predicates comparable to that of t by doing exact match on each query token. The instance of a returned tuple t' is retained if the similarity score between the values of t' and t is higher than the given threshold θ as shown in Equation 3:

$$\frac{|gram\_set(n, t.v) \bigcap gram\_set(n, t'.v)|}{\min(|gram\_set(n, t.v)|, |gram\_set(n, t'.v)|)} > \theta \tag{3}$$

where $gram\_set(n, t.v)$ extracts the character level n-grams from a string. We try to achieve a good coverage on true matches returned by the Boolean query by examining the n-grams while still effectively reducing the candidate set size by setting appropriate thresholds.

3 Evaluation

We evaluate on RKB Person, RKB Publication and SWAT Person instances from RKB[1] and SWAT[2] with metrics Pairwise Completeness, Reduction Ratio and their F1-score [3, 4]. 100K instances for each category were chosen and split into 10 non-overlapping and equal-sized subsets, and their average performance is reported. We also apply an actual entity coreference system [1] to the candidate sets to measure the F1-score of the coreference results and the overall runtime.

We first analyze the impact of α and β. For RKB Publication, the *title* predicate itself has the highest F_L of 0.96; the results will not be affected unless β is high enough such that *title* even gets removed. For SWAT Person, *FOAF:Name* and *CiteSeer:Name* are the only datatype properties used by person instances

[1] http://www.rkbexplorer.com/data/
[2] http://swat.cse.lehigh.edu/resources/data/

with discriminability higher than 0.9, and they have a F_L score of 0.85 and 0.05 respectively; thus setting α lower than 0.85 will not select any candidate pairs. Varying β on this dataset from 0.1-0.9 do not affect the results. Table 1 shows the impact of α and β on RKB Person (θ=0.8) where the *full_name* predicate has the highest F_L of 0.695. Assume β is low enough such that *full_name* is kept, setting α lower than 0.695 will make *full_name* the candidate selection key.

Table 1. Parameter Analysis: α and β on RKB Person; $|Pairs|$: candidate set size; RR: Reduction Ratio; PC: Pairwise Completeness; F_{cs}: the F1-score for RR and PC; *F-Score*: the F1-Score of coreference results; *Time*: the runtime for candidate selection, coreference and the entire process

β	α	*Candidate Selection*				*Coref*	*Time* (s)		
		$\lvert Pairs \rvert$	RR(%)	PC(%)	F_{cs}(%)	*F-score*(%)	CS	*Coref*	*Total*
0.4	0.7-0.9	14,023	99.97	99.28	99.63	93.46	13.15	12.15	25.30
0.3	0.7-0.9	14,024	99.97	**99.33**	**99.65**	**93.48**	13.32	12.13	25.45
0.2	0.7-0.9	14,037	99.97	99.27	99.62	93.35	13.21	12.28	25.47
0.1	0.8-0.9	14,239	99.97	99.31	99.64	93.15	13.19	12.84	26.03
	0.7	14,224	99.97	99.32	99.64	93.26	13.32	12.75	26.06
0.1-0.4	0.6	**14,022**	99.97	99.31	99.64	**93.48**	13.42	12.15	25.56

We also analyze the impact of θ in Table 2. Applying low thresholds generally leads to better coverage on true matches (better PC); however, such improvement may not compensate the much longer runtime needed for the entire process to finish. θ has little influence on the candidate selection time since the only difference is the number of pairs to keep. Because publication titles are generally longer than person names, the system took longer to select candidate pairs on RKB Publication than on the person datasets. Also, on average, an instance has 31.97, 41.69 and 162.63 paths as its context in RKB Person, SWAT Person and RKB Publication respectively thus the coreference algorithm needed more time to finish on RKB Publication when a comparable number of pairs were selected, e.g., RKB Pub 0.7 vs. RKB Per 0.7 and RKB Pub 0.8 vs. SWAT Per 0.8. The additional pairs selected with low thresholds do not have as much context information as those selected with high thresholds; thus the coreference algorithm exhibits a sublinear runtime complexity.

High values for θ are suggested when runtime is critical: the entire process could run faster with certain gain in the coreference results, though we might not have the best PC. When recall is emphasized, low thresholds should be adopted to ensure better PC with tolerable runtime. As found in [2] and shown in Figure 1, a linear behavior is consistent on up to 1 million instances (only 500K instances exist in SWAT Person) for the candidate selection phase.

One limitation of our algorithm is that it currently targets datasets that are primarily composed of strings, and we adopt the same string similarity measure for numeric values, e.g., telephone number. Given that a lot of telephone numbers could be very similar to each other, counting the shared bigrams between two such numbers might greatly increase candidate set size, particularly when the data is primarily describing instances in the same geographic areas.

Another problem is that our algorithm currently performs exact match on each query token when looking up similar instances with the index. This should

Table 2. Parameter Analysis: θ

Dataset	θ (Eq. 3)	*Candidate Selection*				*Coref*	Time (s)		
		$\|Pairs\|$	$RR(\%)$	$PC(\%)$	$F_{cs}(\%)$	F-score(%)	CS	$Coref$	$Total$
RKB Per	0.6	36,881	99.93	**99.73**	**99.83**	92.90	13.12	19.62	32.74
	0.7	20,143	99.96	99.56	99.76	93.07	13.20	14.26	27.46
	0.8	14,024	**99.97**	99.33	99.65	93.48	13.32	12.13	25.45
	0.9	**13,094**	**99.97**	98.98	99.47	**93.54**	13.20	**11.83**	**25.03**
RKB Pub	0.6	100,475	99.80	**99.97**	99.89	99.64	17.96	117.05	135.00
	0.7	21,053	99.96	**99.97**	99.96	99.68	18.58	31.27	49.84
	0.8	6,831	**99.99**	**99.97**	**99.98**	99.74	18.26	13.47	31.73
	0.9	**4,957**	**99.99**	99.80	99.90	**99.76**	18.13	**8.57**	**26.70**
SWAT Per	0.6	28,443	99.94	**99.49**	**99.72**	**95.07**	13.51	15.18	28.69
	0.7	13,343	99.97	99.24	99.60	95.06	13.51	9.68	23.19
	0.8	7,129	**99.99**	98.72	99.35	**95.07**	13.46	7.75	21.21
	0.9	**6,243**	**99.99**	97.96	98.96	**95.07**	13.40	**7.36**	**20.77**

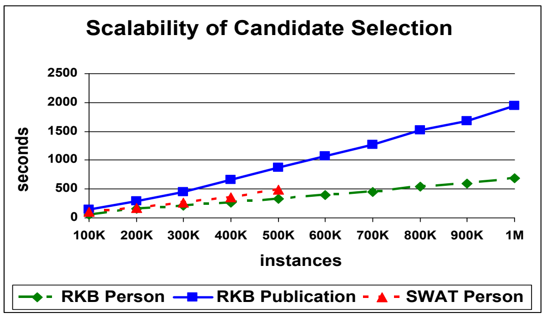

Fig. 1. Scalability of the Proposed Candidate Selection Algorithm

work well on datasets with decent data quality; however, when there are a lot of errors (e.g., misspellings, missing characters or tokens, etc.), our algorithm may not even be able to retrieve all coreferent instances for a given instance. One possible solution to this problem is to adopt fuzzy matching. We could compute the Soundex code for each token and tokens with the same code are treated *similar*. For a given token, we then query the index with all its similar tokens.

References

1. Song, D., Heflin, J.: Domain-independent entity coreference in RDF graphs. In: Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM). pp. 1821–1824 (2010)
2. Song, D., Heflin, J.: Automatically generating data linkages using a domain-independent candidate selection approach. In: 10th International Semantic Web Conference (ISWC). p. Accepted (2011)
3. Yan, S., Lee, D., Kan, M.Y., Giles, C.L.: Adaptive sorted neighborhood methods for efficient record linkage. In: ACM/IEEE Joint Conference on Digital Libraries (JCDL). pp. 185–194 (2007)
4. Elfeky, M.G., Elmagarmid, A.K., Verykios, V.S.: Tailor: A record linkage tool box. In: Proceedings of the 18th International Conference on Data Engineering (ICDE). pp. 17–28 (2002)