Efficient Taxonomic Similarity Joins with Adaptive Overlap Constraint

Pengfei Xu and Jiaheng Lu

Department of Computer Science, University of Helsinki, Finland

ABSTRACT

Established similarity join approaches usually deal with synthetic differences like typos and abbreviations, but neglect the semantic relations between words. Such relations, however, are helpful for obtaining high-quality joining results. In this paper, we leverage the taxonomy knowledge (i.e., a set of IS-A hierarchical relations) to define a similarity measure which finds semantic-similar records from two datasets. Based on this measure, we develop a similarity join algorithm with prefix filtering framework to prune away irrelevant pairs effectively. Our technical contribution here is an algorithm that judiciously selects critical parameters in a prefix filter to maximise its filtering power, supported by an estimation technique and Monte Carlo simulation process.

SIMILARITY MEASURE

Let $S : \{s_1, \cdots, s_i\}$ and $T : \{t_1, \cdots, t_j\}$ be two sets of nodes from a hierarchical taxonomy. Let $s \in S$ and $t \in T$ be two nodes.

- Similarity between two nodes is defined based on their lowest common ancestor (LCA):

$$TS(s,t) = \frac{|LCA(s,t)|}{\max(|s|,|t|)}$$

- Based on node similarity, set similarity aggregates all distinct node similarities:

$$GTS(S,T) = \frac{W(S,T)}{\max(|S|,|T|)} = \frac{\max \sum_p \sum_q I_{pq} TS(s_p, t_q)}{\max(|S|,|T|)}$$

  where S (T) contains p (q) nodes, I_{pq} is an indicator variable (i) controlling whether to select the edge (s_p, t_q), and (ii) ensures any of s_p or t_q is used at most once.

Solving for the value of W in GTS is to find the *maximum weight matching in a bipartite graph*. This can be done in polynomial time using *Hungarian algorithm* [1].

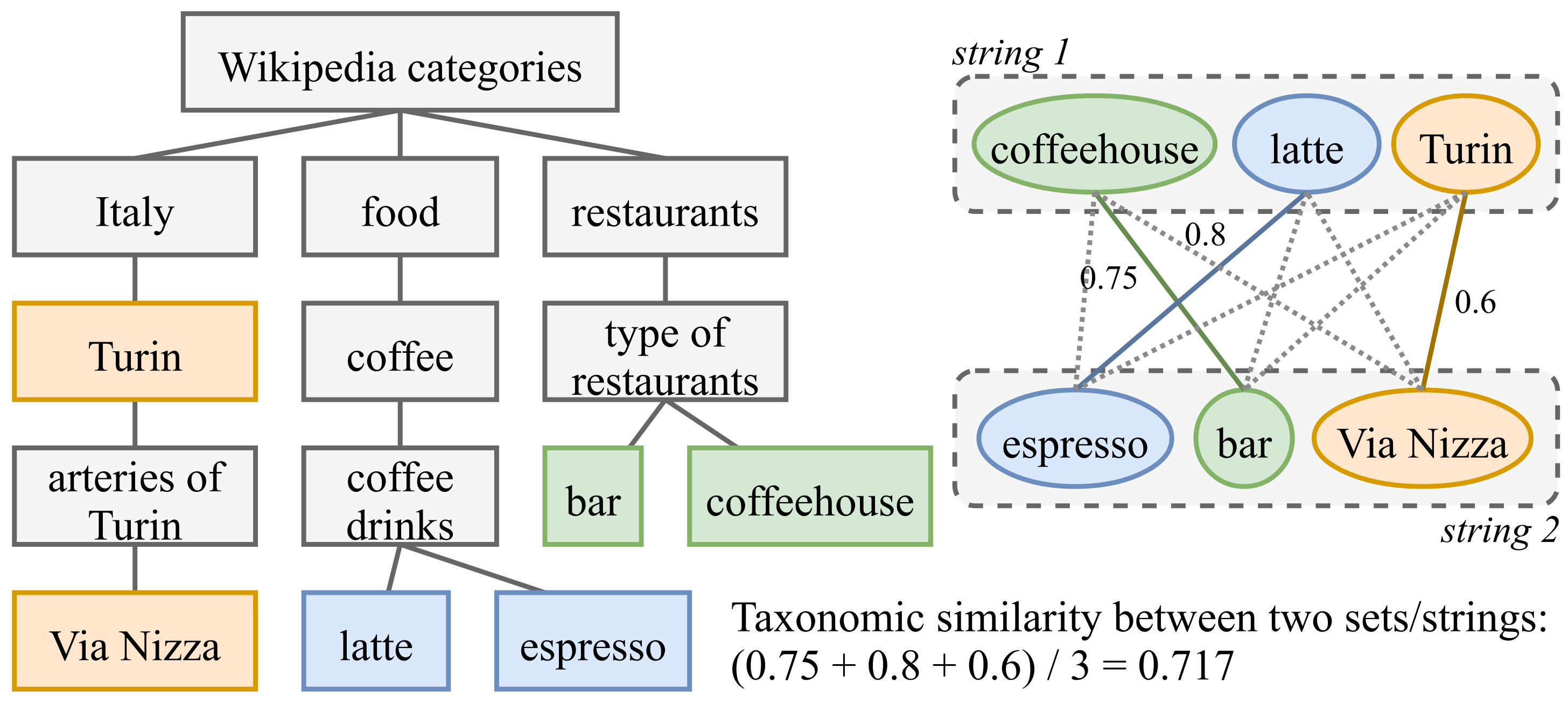

Figure 1: Example of a simplified hierarchical taxonomy and similarity calculation.

Example

Take two strings in Figure 1 as an example. Since the three most-similar node pairs are ("coffeehouse", "bar"), ("latte", "espresso"), and ("Turin", "Via Nizza"), the GTS similarity between two strings becomes 0.717 ($= (0.75 + 0.8 + 0.6)/3$). Note that the distinctness forbid any node from being selected more than once, e.g., selecting both ("latte", "espresso") and ("latte", "Turin") are not allowed.

ACKNOWLEDGEMENT

This work is supported by the Academy of Finland (310321). Contact author and email: jiaheng.lu@helsinki.fi.

ADAPTIVE PREFIX FILTERING

We adopt the popular prefix filtering framework:

- The definition of GTS states that two similar sets must have some similar nodes (say τ).
- Each of τ pairs must have at least $TS(s_i, t_j) = \frac{\theta|T|-\tau+1}{|S|-\tau+1}$ similarity.
- Since TS depends on $|LCA|$, we can get two nodes similarity from their LCA. To find all nodes having a deep-enough LCA, the prefix filtering takes place.
- Each node can only send its deeper-than $\frac{\theta|S|-\tau+1}{|S|-\tau+1}$ ancestors to the index, because having a shallow LCA means not similar enough.
- The prefix filtering pick all strings having τ similar node pairs as candidates.

PARAMETER SELECTION

Different τ leads to various prefix length, the number of candidates, and ultimately, the join time. Testing all τ's on the whole dataset is slow. Hence, we use an estimation algorithm which

- Use independent Bernoulli sampling [3] to get small datasets.
- Run our join algorithm on the sample, with multiple τ's.
- Scale the running time up to full datasets.
- Worst case time grows fast when sample grows due to the Cartesian product. Therefore, our estimator contains multiple stages, each with a small sample size.
- The estimator continuously refine the confidence interval (CI) for each τ, and terminate when the best τ is identified.

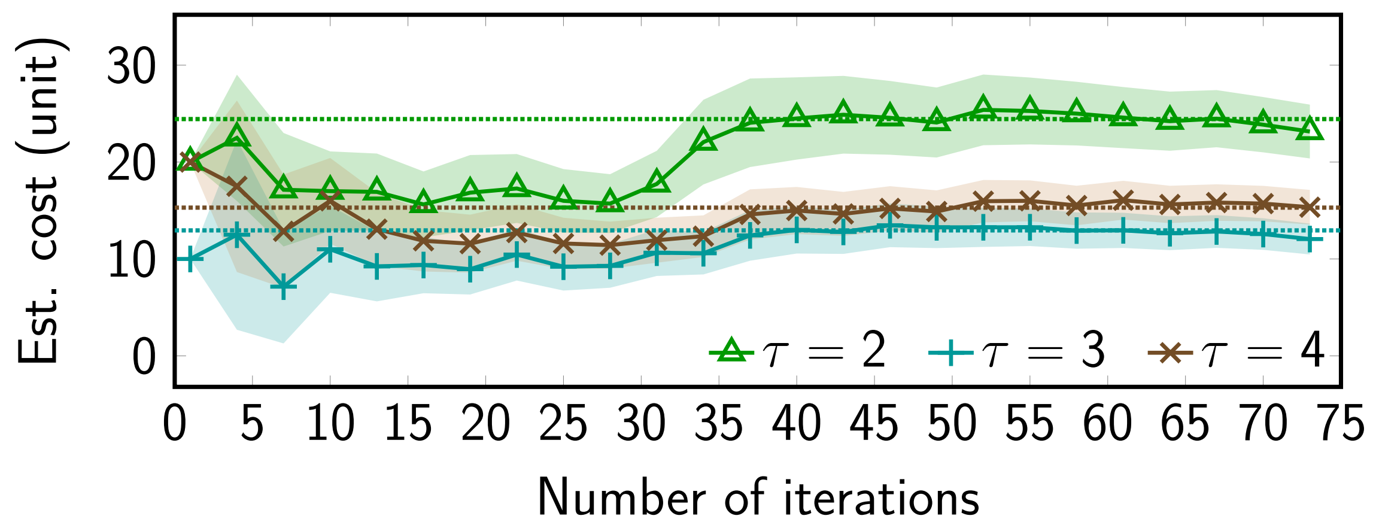

Figure 2: Illustration of the estimation process. Solid lines are estimated means, shaded areas are CI's, dotted lines are empirical real values.

PERFORMANCE

- Performance of our AP-Join v.s. the state-of-art K-Join [2]

Dataset	Algorithm	# Pairs (10^8)			# Candidates (10^6)			Running time (min)		
		0.6	0.7	0.8	0.6	0.7	0.8	0.6	0.7	0.8
Wiki articles	AP-Join	**0.42**	**0.08**	**0.01**	**11.52**	**3.04**	**0.27**	**10.03**	**2.64**	**0.55**
	K-Join	0.87	0.25	0.07	28.42	8.35	1.98	22.28	6.65	1.74
OHSUMED	AP-Join	**1.08**	4.97	1.72	**63.43**	**0.64**	**0.26**	**41.81**	**4.44**	**1.67**
	K-Join	-	**2.13**	**0.86**	-	115.58	38.42	-	80.01	25.33

- Estimation accuracy and speed

Dataset	Accuracy from 128 runs varies θ				Estimation time varies θ (s)			
	0.6	0.7	0.8	0.9	0.6	0.7	0.8	0.9
Wiki articles	92.03%	100%	100%	100%	2.58	1.69	1.74	1.51
OHSUMED	96.09%	99.22%	90.63%	100%	4.63	1.10	2.04	1.42

REFERENCES

J. Munkres, *Algorithms for the Assignment and Transportation Problems*, JSIAM, 5 (1957), pp. 32–38.

Z. Shang, Y. Liu, G. Li, and J. Feng, *K-Join: Knowledge-Aware Similarity Join*, TKDE, 28 (2016), pp. 3293–3308.

D. Vengerov, A. C. Menck, M. Za\"\it, and S. Chakkapen, *Join Size Estimation Subject to Filter Conditions*, PVLDB, 8 (2015), pp. 1530–1541.

Contact: pengfei.xu@helsinki.fi