Utilizing Regular Expressions for Instance-Based Schema Matching

Benjamin Zapilko, Matthäus Zloch, and Johann Schaible

GESIS - Leibniz Institute for the Social Sciences
Unter Sachsenhausen 6-8, 50667 Cologne, Germany
{benjamin.zapilko,matthaeus.zloch,johann.schaible}@gesis.org

Abstract. Statistical data consists mostly of numerical values, entries of codelists like country codes or acronyms for gender. Such values are typically described according to specific patterns. In this paper we present a novel approach for instance-based schema matching, where regular expressions are utilized for matching patterns of instance values.

1 Motivation and Background

In various domains, e.g. the social sciences, the matching of statistical data is a typical task. Schema elements of statistical data, e.g. rows or columns of a spreadsheet, are named usually by simple and short labels, sometimes even with abbreviated terms. However, the structure and semantics of their instances (e.g. numerical values, entries of codelists) differ in various aspects from text-heavy data. Instances are often described by a specific syntactical pattern, e.g. dates consist of numerical values divided by periods or slashes or a three-letter code for a geographical area.

For instance-based schema matching [3] states that different domains reveal new challenges like treating new types of information resources, e.g. spatial or temporal information or domain-specific constrains. According to [2] especially domain-specific values, significant occurrences and patterns of values are relevant characteristics to be considered at instance level, as well as integrity constraints for schema elements and their instance values. In [1] the matching process is enhanced by applying a constraint-based matching. Moreover, regular expressions and catchwords are considered for instance-based schema matching in [4]. We focus on statistical data, where the potential of patterns and regular expressions for schema matching can be fully exposed.

2 Schema Matching using Regular Expressions

By utilizing pattern classes our approach considers two schema elements as a match, if their instances can be expressed via at least one regular expression of the same pattern class. We define multiple pattern classes, which correspond to a specific data element, e.g. dates, age groups or geographical codes, and contain various patterns for describing this data element. For a data element "date" different patterns might be e.g. `[0-9]{4}`, `[0-9]{2}-[0-9]{4}` or `[0-9]{2}.[0-9]{2}.[0-9]{4}`. Each pattern is expressed as a regular expression and is assigned a weighting, which states the accuracy of the pattern to compass typical instances of the data element. Inside a pattern class the regular expressions are sorted by their weightings in descending order.

We assume two datasets M and N with their schema elements $S_M \in M$ and $S_N \in N$. The pattern classes C_x with $C_x = \{(regex, \omega) | regex$ matches x, $0 < \omega < 1\}$ contain multiple regular expressions $regex$ describing the statistical data element x of the class. They are accompanied with a weighting ω.

For each pattern class C_x, we compute an average weighting for every schema element S_M and S_N. This average weighting indicates how often instances of the schema element can be expressed by a pattern of the class. Hereby, as soon as an instance can be expressed by a $(regex, \omega) \in C_x$, the value of ω is added to the sum of all weightings, whose regular expressions previously matched another instance from this same schema element, resulting in the final $\sum_0 \omega$. The average is then retrieved by normalizing this sum regarding the total number of instances inside this particular schema element. For each S_M, this is $avg(S_M) = \frac{\sum_0 \omega}{|\text{Instances in } S_M|}$. For S_N the average is calculated analogously. If this average weight is not 0, the schema element is collected among its average weight in a set. We define these sets as M_x and N_x with $M_x = \{(S_M, avg(S_M)\}$ and $N_x = \{(S_N, avg(S_N)\}$.

The Cartesian product of M_x and N_x is computed and added to $Matches_x$, in which a triple $(S_M, S_N, avg(S_M) * avg(S_N))$ defines a match between a S_M and a S_N with the probability of $avg(S_M) * avg(S_N)$. Finally, the result set $Matches_x$ contains all matches between two datasets M and N.

Our approach has been implemented in Java using the JENA API. The source code and an executable jar file are available at https://github.com/mazlo/smurf. In first experiments with real-world statistical data we obtained better results for matching schema elements than other existing matching systems. A detailed evaluation with generic test datasets is currently work-in-progress. We aim to extend our approach to extract patterns from instance values and to generate weightings automatically. Feature extraction from instance values can enhance our approach in computing weightings and in assigning regular expressions to adequate pattern classes.

References

1. Engmann, D.; Maßmann, S. Instance Matching with COMA++. BTW Workshops, 2007, 28-37
2. Halevy, A. Why Your Data Won't Mix Queue, ACM, 2005, 3, 50-58
3. Shvaiko, P.; Euzenat, J. Ontology Matching: State of the Art and Future Challenges. IEEE Transactions on Knowledge and Data Engineering, 2011, 99
4. Zaiss, K.; Schlueter, T.; Conrad, S. Instance-Based Ontology Matching Using Different Kinds of Formalisms. Proceedings of the International Conference on Semantic Web Engineering, Oslo, Norway, July, 2009, 29-31